A ph Mesh Refinement Method for Optimal Control

Michael. A. Patterson*
William W. Hager †
Anil V. Rao‡

University of Florida
Gainesville, FL 32611

Abstract

A mesh refinement method is described for solving a continuous-time optimal control problem using collocation at Legendre-Gauss-Radau points. The method allows for changes in both the number of mesh intervals and the degree of the approximating polynomial within a mesh interval. First, a relative error estimate is derived based on the difference between the Lagrange polynomial approximation of the state and a Legendre-Gauss-Radau quadrature integration of the dynamics within a mesh interval. The derived relative error estimate is then used to decide if the degree of the approximating polynomial within a mesh should be increased or if the mesh interval should be divided into sub-intervals. The degree of the approximating polynomial within a mesh interval is increased if the polynomial degree estimated by the method remains below a maximum allowable degree. Otherwise, the mesh interval is divided into sub-intervals. The process of refining the mesh is repeated until a specified relative error tolerance is met. Three examples highlight various features of the method and show that the approach is more computationally efficient and produces significantly smaller mesh sizes for a given accuracy tolerance when compared with fixed-order methods.

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 control are discretized at a set of appropriately chosen points in the time interval of interest. The

*Postdoctoral Teaching Fellow, Department of Mechanical and Aerospace Engineering. E-mail: mpatterson@ufl.edu.

†Professor, Department of Mathematics. E-mail: hager@ufl.edu.

‡Associate Professor, Department of Mechanical and Aerospace Engineering. Corresponding Author. Associate Fellow AIAA. E-mail: anilvrao@ufl.edu.

continuous-time optimal control problem is then transcribed to a finite-dimensional nonlinear 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 as been done in the class of direct *Gaussian quadrature orthogonal collocation* methods.[6–19] 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.[20–22] The most well developed Gaussian quadrature methods are those that employ either Legendre-Gauss (LG) points,[9,13] Legendre-Gauss-Radau (LGR) points,[14,15,17] or Legendre-Gauss-Lobatto (LGL) points.[6]

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 is allowed to vary. Originally, hp methods were developed as finite-element methods for solving partial differential equations.[23–27] In the past few years the problem of developing hp methods for solving optimal control problems has been of interest.[28,29] This recent research has shown that convergence using hp methods can be achieved with a significantly smaller finite-dimensional approximation than would be required when using either an h or a p method.

Motivated by the desire to improve computational efficiency when solving the NLP while providing high accuracy in the discrete approximation of the optimal control problem, in this

paper we describe a new ph Gaussian quadrature collocation method for solving continuous-time nonlinear optimal control problems. Here we have deliberately changed the order from hp to ph since our scheme tries to achieve the prescribed error tolerance by increasing the degree of the polynomials in the approximating space, and if this fails, by increasing the number of mesh intervals. The method is divided into three parts. First, an approach is developed for estimating the relative error in the state within each mesh interval. This relative error estimate is obtained by comparing the original state variable to a higher order approximation of the state. This relative error estimate is used to determine if the degree of the polynomial approximation should be increased or if mesh intervals should be added. The polynomial degree is increased if it is estimated that the ensuing mesh requires a polynomial degree that is less than a maximum allowable degree. Otherwise, the mesh is refined. This process is repeated on a series of meshes until a specified accuracy tolerance is met. The decision to increase the polynomial degree or refine the mesh is based on the ratio of the maximum relative error and the accuracy tolerance and is consistent with the known exponential convergence of a Gaussian quadrature method for a problem whose solution is smooth.

Various mesh refinement methods employing direct collocation methods have been described in recent years.[5,28–30] Reference 30 describes a method that employs a differentiation matrix to attempt to identify switches, kinks, corners, and other discontinuities in the solution, and uses Gaussian quadrature rules to generate a mesh that is dense near the end points of the time interval of interest. Reference 5 employs a density function and attempts to generate a fixed-order mesh on which to solve the problem. References 31 and 32 (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. References 28 and 29 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. 28 and 29 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. 28 and 29 does not take advantage of the exponential convergence rate of a Gaussian quadrature collocation method. 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. On the other hand, in this paper an estimate of the error on a given mesh is obtained by varying the degree of the polynomial in the Gaussian quadrature approximation.

The method of this paper is fundamentally different from any of the previously developed methods due to the fact that it takes advantage of the exponential convergence properties of a Gaussian quadrature. Specifically, in the discretization used in this paper, the state is approximated using a piecewise polynomial and the dynamics are collocated at the Legendre-Gauss-Radau quadrature points in each interval. This leads to a finite dimensional mathematical programming problem that is solved to obtain estimates for the control and the state at the collocation points. A relative error estimate is then derived that uses the difference between an interpolated value of the state and a Legendre-Gauss-Radau quadrature approximation to the integral of the state dynamics. This relative error estimate remains computationally tractable when a high-accuracy solution is desired and reduces significantly the number of collocation points required to meet a specified accuracy tolerance when compared with the the methods of Ref. 28 or 29. Furthermore, different from the error estimate developed in Ref. 3, the error estimate in this research utilizes the Legendre-Gauss-Radau quadrature points in both the interpolation of the state and the integration of the state along the interpolated solution. Consequently, the approach of this research enables the use of a Gaussian quadrature method to estimate the errors, thereby achieving convergence using fewer collocation points than may be necessary using an h method with the error estimate of Ref. 3. It is noted, however, that in the case of an h LGR method (that is, a method whose order is the same in all mesh intervals), the error estimate in this paper is similar to the estimate derived in Ref. 3 with the exception that the approach of this paper still employs a higher-order integration method than the approach of Ref. 3.

While the mesh refinement method presented in this paper is based on only the state error and seems to ignore the error in the costate as well as ignore the control minimum principle, in earlier work (such as Refs. 14–16) a close connection has been established between the first-order optimality conditions for the mathematical program and the continuous first-order optimality conditions (that is, the system dynamics, the costate equation, and the minimum principle).

Specifically, it is observed that at a solution of the mathematical programming problem, the minimum principle holds at the collocation points. Thus by solving the mathematical program, the minimum principle is satisfied exactly. As has been shown in Refs. 14–16, the discretization of the state equation leads to an induced discretization for the costate equation which is expressed in terms of the multipliers of the mathematical programming problem. Hence, when the mathematical programming problem is solved, in essence a two point boundary value problem is solved in which the control has been eliminated through the minimum principle. In this paper we develop a mesh refinement method that is based entirely on the estimation of the error in the state. While the error in the induced costate approximation could be monitored using the same approach as is used to monitor the state error, one finds that the errors connected with the system dynamics and the induced costate equation are tightly coupled in that the induced costate error is large during intervals where the state error is large. As a result, the benefit of monitoring the error in the state and the induced costate is marginal when compared to monitoring the error in only the state. Thus it is more efficient and simpler to focus entirely on the error in the state equation. The effectiveness of our mesh refinement strategy is studied on three examples that have different features in the optimal solution and, thus, exercise different benefits of the new ph approach. It is found that the mesh refinement method developed in this paper is an effective yet simple way to generate meshes and to reduce computation times when compared with fixed-order h methods.

The significance of this research is threefold. First, we develop a systematic way to estimate the error in the Radau discretization of an optimal control problem using the fact that Radau collocation is a Gaussian quadrature integration method. Second, based on the derived estimate of the error, we provide a simple yet effective ph mesh refinement method that allows both the degree of the approximating polynomial and the number of mesh intervals to vary. Third, we show on three non-trivial examples that the ph mesh refinement method developed in this paper is more computationally efficient and produces smaller meshes for a given accuracy tolerance when compared with traditional fixed-order h methods.

This paper is organized as follows. In Section 2 we provide a motivation for our new ph method. In Section 3 we state the continuous-time Bolza optimal control problem. In Section 4, we state the integral form of the ph Legendre-Gauss-Radau integration method[14–16] that is used as the basis for the ph mesh refinement method developed in this paper. In Section 5 we develop

the error estimate and our new ph-adaptive mesh refinement method. In Section 6 we apply the method of Section 5 to three examples that highlight different features of the method. In Section 7 we describe the key features of our approach and compare our method to recently developed hp-adaptive methods. Finally, in Section 8 we provide conclusions on our work.

2 Motivation for New ph-Adaptive Collocation Method

In order to motivate the development of our new ph-adaptive Gaussian quadrature collocation method, consider the following two first-order differential equations on the interval $\tau \in [-1, +1]$:

$$\frac{dy_1}{d\tau} = f_1(\tau) = \pi\cos(\pi\tau), \qquad y_1(-1) = y_{10}, \tag{1}$$

$$\frac{dy_2}{d\tau} = f_2(\tau) = \begin{cases} 0, & -1 \leq \tau < -1/2 \\ \pi\cos(\pi\tau), & -1/2 \leq \tau \leq +1/2, \\ 0, & +1/2 < \tau \leq +1 \end{cases}, \quad y_2(-1) = y_{20}. \tag{2}$$

The solutions to the differential equations of Eqs. (1) and (2) are given, respectively, as

$$y_1(\tau) = y_{10} + \sin(\pi\tau), \tag{3}$$

$$y_2(\tau) = \begin{cases} y_{20}, & -1 \leq \tau < -1/2, \\ y_{20} + 1 + \sin(\pi\tau), & -1/2 \leq \tau \leq +1/2, \\ y_{20} + 2, & +1/2 < \tau \leq +1. \end{cases} \tag{4}$$

Suppose now that it is desired to approximate the solutions to the differential equations of Eqs. (1) and (2) using the following three different methods that employ the Legendre-Gauss-Radau[33] collocation method as described in various forms in Refs. 14–17, 19: (1) a p-method where the state is approximated using an N_k^{th} degree polynomial on $[-1, +1]$ and N_k is allowed to vary; (2) an h-method using K equally spaced mesh intervals where K is allowed to vary and a fixed fourth-degree polynomial is employed within each mesh interval; and (3) a ph method where *both* the number of mesh intervals, K, and the degree of the approximating polynomial, N_k, within each mesh interval are allowed to vary. Using any of the aforementioned approximations (p, h, or ph), within any mesh interval $[T_{k-1}, T_k]$ the functions $y_i(\tau), \ (i = 1, 2)$ are approximated using the following Lagrange polynomial approximations:

$$y_i^{(k)}(\tau) \approx Y_i^{(k)}(\tau) = \sum_{j=1}^{N+1} Y_{ij}^{(k)} \ell_j^{(k)}(\tau), \quad \ell_j^{(k)}(\tau) = \prod_{\substack{l=1 \\ l \neq j}}^{N+1} \frac{\tau - \tau_l^{(k)}}{\tau_j^{(k)} - \tau_l^{(k)}}, \tag{5}$$

where the support points for $\ell_j^{(k)}(\tau),\ j = 1, \ldots, N_k+1$, are the N_k Legendre-Gauss-Radau points[33] $(\tau_1^{(k)}, \ldots, \tau_N^{(k)})$ on $[T_{k-1}, T_k]$ such that $\tau_1^{(k)} = T_{k-1}$ and $\tau_{N+1}^{(k)} = T_k$ is a non-collocated point that defines the end of mesh interval k. Within any particular mesh interval $[T_{k-1}, T_k] \subseteq [-1, +1]$, the approximations of $y_i^{(k)}(\tau),\ i = 1, 2$, are given at the support points $\tau_{j+1}^{(k)},\ j = 1, \ldots, N$, as

$$y_i^{(k)}(\tau_{j+1}^{(k)}) \approx Y_i^{(k)}(\tau_{j+1}^{(k)}) = Y_{i1}^{(k)} + \sum_{l=1}^{N} I_{jl}^{(k)} f_i(\tau_l^{(k)}), \quad (i = 1, 2),\ \ (j = 1, \ldots, N), \tag{6}$$

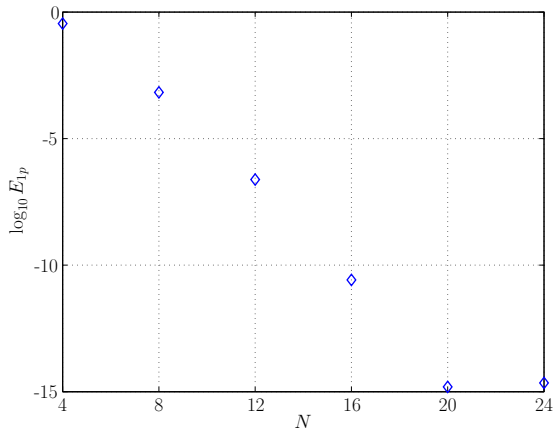
where $Y_{i1}^{(k)}$ is the approximation to $y_i(\tau_1^{(k)})$ at the start of the mesh interval and $I_{jl}^{(k)}$ $(j, l = 1, \ldots, N_k)$ is the $N_k \times N_k$ Legendre-Gauss-Radau integration matrix (see Ref. 14 for details) defined on the mesh interval $[T_{k-1}, T_k]$.

Suppose now that we define the maximum absolute error in the solution of the differential equation as

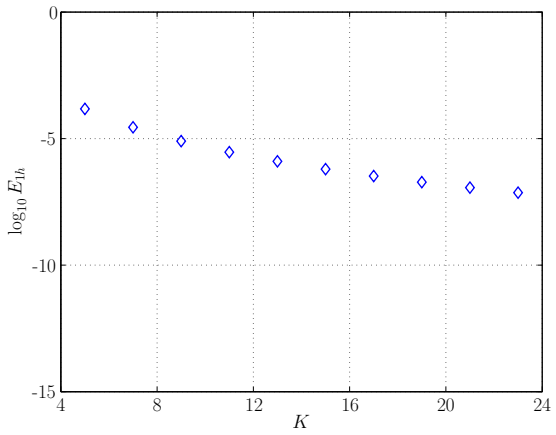
$$E_i = \max_{\substack{j \in [1,\ldots,N_k] \\ k \in [1,\ldots,K]}} \left| Y_{ij}^{(k)} - y_i(\tau_j^{(k)}) \right|, \quad (i = 1, 2).$$

Figures 1a and 1b show the base-10 logarithm of E_1 as a function of N for the p method and as a function of K for the h method. First, it is seen that because $y_1(\tau)$ is a smooth function, the p method converges exponentially as a function of N while the h method converges significantly more slowly as a function of K. Figures 1c and 1d show E_2. Unlike y_1, the function y_2 is continuous but *not smooth*. As a result, the h method converges faster than the p method because no single polynomial (regardless of degree) on $[-1, +1]$ is able to approximate the solution to Eq. (2) as accurately as a piecewise polynomial. However, while the h method converges more quickly than does the p method when approximating the solution of Eq. (2), it is seen that the h method does not converge as quickly as the p method does when approximating the solution to Eq. (1). In fact, when approximating the solution of Eq. (1) it is seen that the h method achieves an error of $\approx 10^{-7}$ for $K = 24$ whereas the p method converges exponentially and achieves an error of $\approx 10^{-15}$ for $N = 20$. As a result, an h method does not provide the fastest possible convergence rate when approximating the solution to a differential equation whose solution is smooth.

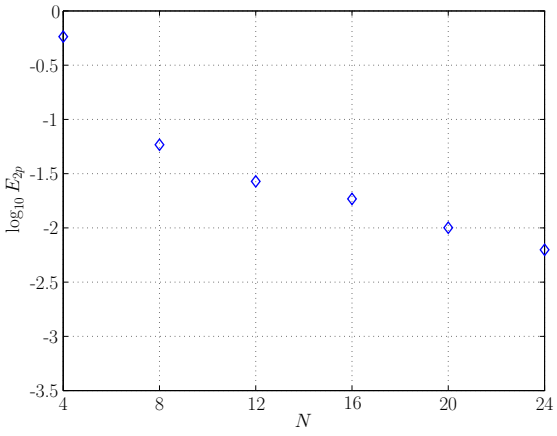
Given the aforementioned p and h analysis, suppose now that the solution to Eq. (2) is approximated using the aforementioned ph Radau method (that is, both the number of mesh intervals and the degree of the approximating polynomial within each mesh interval are allowed to vary). Assume further that the ph method is constructed such that the time interval $[-1, +1]$ is divided into three mesh intervals $[-1, -1/2]$, $[-1/2, +1/2]$, and $[+1/2, +1]$ and Lagrange polyno-



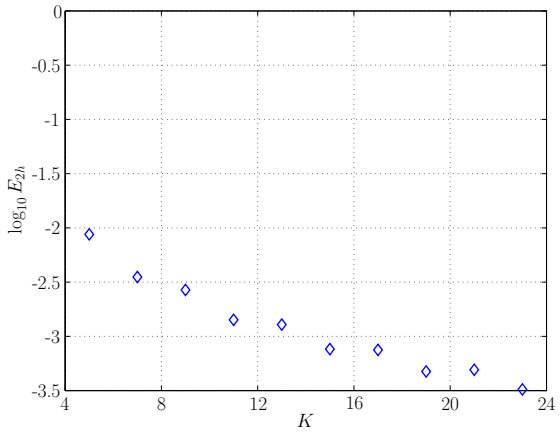
(a) $\log_{10} E_{1p}$ vs. N Using p Radadu Method.



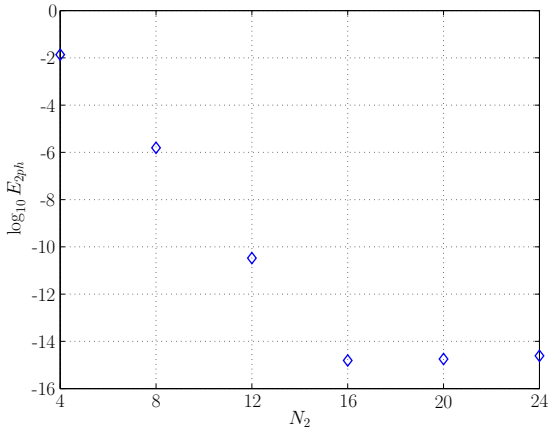
(b) $\log_{10} E_{1h}$ vs. K Using h Radau Method with $N = 4$.



(c) $\log_{10} E_{2p}$ vs. N Using p Radau Method.



(d) $\log_{10} E_{2h}$ vs. K Using h Radau Method with $N = 4$.



(e) $\log_{10} E_{2ph}$ vs. N_2 Using ph Radau Method.

Figure 1: Base-10 Logarithm of Absolute Errors in Solutions of Eqs. (1) and (2) at Lagrange polynomial Support Points Using p, h Methods, and ph Methods.

mial approximations of the form of Eq. (5) of degree N_1, N_2, and N_3, respectively, are used in each mesh interval. Furthermore, suppose N_1, N_2, and N_3 are allowed to vary. Because the solution $y_2(\tau)$ is a constant in the first and third mesh intervals, it is possible to set $N_1 = N_3 = 2$ and vary *only* N_2. Figure 1e shows the error in $y_2(\tau)$, $E_{2ph} = \max|y_2 - Y_2|$ using the aforementioned three interval ph approach. Similar to the results obtained using the p method when approximating the solution of Eq. (1), in this case the error in the solution of Eq. (2) converges exponentially as a function of N_2. Thus, while an h method may outperform a p method on a problem whose solution is not smooth, it is possible improve the convergence rate by using an ph-adaptive method. The foregoing analysis provides a motivation for the development of the ph method described in the remainder of this paper.

3 Bolza Optimal Control Problem

Without loss of generality, consider the following general optimal control problem in Bolza form. Determine the state, $\mathbf{y}(t) \in \mathbb{R}^{n_y}$, the control $\mathbf{u}(t) \in \mathbb{R}^{n_u}$, the initial time, t_0, and the terminal time, t_f, on the time interval $t \in [t_0, t_f]$ that minimize the cost functional

$$\mathcal{J} = \phi(\mathbf{y}(t_0), t_0, y(t_f), t_f) + \int_{t_0}^{t_f} g(\mathbf{y}(t), \mathbf{u}(t), t)\, dt \tag{7}$$

subject to the dynamic constraints

$$\frac{d\mathbf{y}}{dt} = \mathbf{a}(\mathbf{y}(t), \mathbf{u}(t), t), \tag{8}$$

the inequality path constraints

$$\mathbf{c}_{\min} \leq \mathbf{c}(\mathbf{y}(t), \mathbf{u}(t), t) \leq \mathbf{c}_{\max}, \tag{9}$$

and the boundary conditions

$$\mathbf{b}_{\min} \leq \mathbf{b}(\mathbf{y}(t_0), t_0, \mathbf{y}(t_f), t_f) \leq \mathbf{b}_{\max}. \tag{10}$$

The functions ϕ, g, $\mathbf{a}$, $\mathbf{c}$ and $\mathbf{b}$ are defined by the following mappings:

$$\begin{aligned}
\phi &: \mathbb{R}^{n_y} \times \mathbb{R} \times \mathbb{R}^{n_y} \times \mathbb{R} \rightarrow \mathbb{R}, \\
g &: \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \times \mathbb{R} \rightarrow \mathbb{R}, \\
\mathbf{a} &: \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \times \mathbb{R} \rightarrow \mathbb{R}^{n_y}, \\
\mathbf{c} &: \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \times \mathbb{R} \rightarrow \mathbb{R}^{n_c}, \\
\mathbf{b} &: \mathbb{R}^{n_y} \times \mathbb{R} \times \mathbb{R}^{n_y} \times \mathbb{R} \rightarrow \mathbb{R}^{n_b},
\end{aligned}$$

where all vector functions of time are treated as *row* vectors. In this presentation it will be useful to modify the Bolza problem given in Eqs. (7)–(10) as follows. Let $\tau \in [-1,+1]$ be a new independent variable such that

$$t = \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2}. \tag{11}$$

The Bolza optimal control problem of Eqs. (7)–(10) is then defined in terms of the variable τ as follows. Determine the state, $\mathbf{y}(\tau) \in \mathbb{R}^{n_y}$, the control $\mathbf{u}(\tau) \in \mathbb{R}^{n_u}$, the initial time, t_0, and the terminal time t_f on the time interval $\tau \in [-1,+1]$ that minimize the cost functional

$$\mathcal{J} = \phi(\mathbf{y}(-1), t_0, \mathbf{y}(+1), t_f) + \frac{t_f - t_0}{2}\int_{-1}^{+1} g(\mathbf{y}(\tau), \mathbf{u}(\tau), \tau; t_0, t_f)\, d\tau \tag{12}$$

subject to the dynamic constraints

$$\frac{d\mathbf{y}}{d\tau} = \frac{t_f - t_0}{2}\mathbf{a}(\mathbf{y}(\tau), \mathbf{u}(\tau), \tau; t_0, t_f), \tag{13}$$

the inequality path constraints

$$\mathbf{c}_{\min} \leq \mathbf{c}(\mathbf{y}(\tau), \mathbf{u}(\tau), \tau; t_0, t_f) \leq \mathbf{c}_{\max}, \tag{14}$$

and the boundary conditions

$$\mathbf{b}_{\min} \leq \mathbf{b}(\mathbf{y}(-1), t_0, \mathbf{y}(+1), t_f) \leq \mathbf{b}_{\max}. \tag{15}$$

Suppose now that the time interval $\tau \in [-1,+1]$ is divided into a *mesh* consisting of K *mesh intervals* $\mathcal{S}_k = [T_{k-1}, T_k],\ k = 1,\ldots,K$, where $(T_0,\ldots,T_K)$ are the *mesh points*. The mesh intervals $\mathcal{S}_k,\ (k = 1,\ldots,K)$ have the properties that $\bigcup_{k=1}^{K} \mathcal{S}_k = [-1,+1]$ and $\bigcap_{k=1}^{K} \mathcal{S}_k = \emptyset$, while the mesh points have the property that $-1 = T_0 < T_1 < T_2 < \cdots < T_K = +1$. Let $\mathbf{y}^{(k)}(\tau)$ and $\mathbf{u}^{(k)}(\tau)$ be the state and control in $\mathcal{S}_k$. The Bolza optimal control problem of Eqs. (12)–(15) can then rewritten as follows. Minimize the cost functional

$$\mathcal{J} = \phi(\mathbf{y}^{(1)}(-1), t_0, \mathbf{y}^{(K)}(+1), t_f) + \frac{t_f - t_0}{2}\sum_{k=1}^{K}\int_{T_{k-1}}^{T_k} g(\mathbf{y}^{(k)}(\tau), \mathbf{u}^{(k)}(\tau), \tau; t_0, t_f)\, d\tau, \quad (k = 1,\ldots,K), \tag{16}$$

subject to the dynamic constraints

$$\frac{d\mathbf{y}^{(k)}(\tau)}{d\tau} = \frac{t_f - t_0}{2}\mathbf{a}(\mathbf{y}^{(k)}(\tau), \mathbf{u}^{(k)}(\tau), \tau; t_0, t_f), \quad (k = 1,\ldots,K), \tag{17}$$

the path constraints

$$\mathbf{c}_{\min} \leq \mathbf{c}(\mathbf{y}^{(k)}(\tau), \mathbf{u}^{(k)}(\tau), \tau; t_0, t_f) \leq \mathbf{c}_{\max}, \quad (k = 1, \ldots, K), \tag{18}$$

and the boundary conditions

$$\mathbf{b}_{\min} \leq \mathbf{b}(\mathbf{y}^{(1)}(-1), t_0, \mathbf{y}^{(K)}(+1), t_f) \leq \mathbf{b}_{\max}. \tag{19}$$

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 $(T_1, \ldots, T_{K-1})$.

4 Legendre-Gauss-Radau Collocation Method

The ph form of the continuous-time Bolza optimal control problem in Section 3 is discretized using collocation at Legendre-Gauss-Radau (LGR) points.[14–17,19] 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

$$\mathbf{y}^{(k)}(\tau) \approx \mathbf{y}^{(k)}(\tau) = \sum_{j=1}^{N_k+1} \mathbf{y}_j^{(k)} \ell_j^{(k)}(\tau), \quad \ell_j^{(k)}(\tau) = \prod_{\substack{l=1 \\ l \neq j}}^{N_k+1} \frac{\tau - \tau_l^{(k)}}{\tau_j^{(k)} - \tau_l^{(k)}}, \tag{20}$$

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[33] (LGR) 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. (20) with respect to τ, we obtain

$$\frac{d\mathbf{y}^{(k)}(\tau)}{d\tau} = \sum_{j=1}^{N_k+1} \mathbf{y}_j^{(k)} \frac{d\ell_j^{(k)}(\tau)}{d\tau}. \tag{21}$$

The cost functional of Eq. (16) is then approximated using a multiple-interval LGR quadrature as

$$\mathcal{J} \approx \phi(\mathbf{y}_1^{(1)}, t_0, \mathbf{y}_{N_K+1}^{(K)}, t_f) + \sum_{k=1}^{K} \sum_{j=1}^{N_k} \frac{t_f - t_0}{2} w_j^{(k)} g(\mathbf{y}_j^{(k)}, \mathbf{u}_j^{(k)}, \tau_j^{(k)}; t_0, t_f), \tag{22}$$

where $w_j^{(k)}\ (j = 1, \ldots, N_k)$ are the LGR quadrature weights[33] in $\mathcal{S}_k = [T_{k-1}, T_k]$, $k \in [1, \ldots, K]$, $\mathbf{u}_i^{(k)}$, $i = 1, \ldots, N_k$, are the approximations of the control at the N_k LGR points in mesh interval $k \in [1, \ldots, K]$, $\mathbf{y}_1^{(1)}$ is the approximation of $\mathbf{y}(T_0)$, and $\mathbf{y}_{N_K+1}^{(K)}$ is the approximation of $\mathbf{y}(T_K)$

(where we recall that $T_0 = -1$ and $T_K = +1$). Collocating the dynamics of Eq. (17) at the N_k LGR points using Eq. (21), we have

$$\sum_{j=1}^{N_k+1} D_{ij}^{(k)} \mathbf{y}_j^{(k)} - \frac{t_f - t_0}{2} \mathbf{a}(\mathbf{y}_i^{(k)}, \mathbf{u}_i^{(k)}, \tau_i^{(k)}; t_0, t_f) = \mathbf{0}, \quad (i = 1, \ldots, N_k),$$

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*[14] $\mathbf{D}^{(k)}$ associated with $\mathcal{S}_k,\ k \in [1, \ldots, K]$. While the dynamics can be collocated in differential form, in this paper we choose to collocate the dynamics using the equivalent *implicit integral form* (see Refs. 14–16 for details). The implicit integral form of the Legendre-Gauss-Radau collocation method is given as

$$\mathbf{y}_{i+1}^{(k)} - \mathbf{y}_1^{(k)} - \frac{t_f - t_0}{2} \sum_{j=1}^{N_k} I_{ij}^{(k)} \mathbf{a}(\mathbf{y}_i^{(k)}, \mathbf{u}_i^{(k)}, \tau_i^{(k)}; t_0, t_f) = \mathbf{0}, \qquad (i = 1, \ldots, N_k), \tag{23}$$

where $I_{ij}^{(k)},\ (i = 1, \ldots, N_k, j = 1, \ldots, N_k, k = 1, \ldots, K)$ is the $N_k \times N_k$ *Legendre-Gauss-Radau integration matrix* in mesh interval $k \in [1, \ldots, K]$; it is obtained by inverting a submatrix of the differentiation matrix formed by columns 2 through $N_k + 1$:

$$\mathbf{I}^{(k)} = \left[\mathbf{D}_2^{(k)} \cdots \mathbf{D}_{N_k+1}^{(k)}\right]^{-1},$$

It is noted for completeness that $\mathbf{I}^{(k)}\mathbf{D}_1^{(k)} = -\mathbf{1}$ (see Refs. 14–16), where $\mathbf{1}$ is a column vector of length N_k of all ones.[14–16] Next, the path constraints of Eq. (18) in $\mathcal{S}_k,\ k \in [1, \ldots, K]$ are enforced at the N_k LGR points as

$$\mathbf{c}_{\min} \leq \mathbf{c}(\mathbf{y}_i^{(k)}, \mathbf{u}_i^{(k)}, \tau_i^{(k)}; t_0, t_f) \leq \mathbf{c}_{\max}, \quad (i = 1, \ldots, N_k). \tag{24}$$

The boundary conditions of Eq. (19) are approximated as

$$\mathbf{b}_{\min} \leq \mathbf{b}(\mathbf{y}_1^{(1)}, t_0, \mathbf{y}_{N_K+1}^{(K)}, t_f) \leq \mathbf{b}_{\max}. \tag{25}$$

It is noted that continuity in the state at the interior mesh points $k \in [1, \ldots, K-1]$ is enforced via the condition

$$\mathbf{y}_{N_k+1}^{(k)} = \mathbf{y}_1^{(k+1)}, \quad (k = 1, \ldots, K-1), \tag{26}$$

where the *same* variable is used for both $\mathbf{y}_{N_k+1}^{(k)}$ and $\mathbf{y}_1^{(k+1)}$. Hence, the constraint of Eq. (26) is eliminated from the problem because it is taken into account explicitly. The NLP that arises

from the LGR collocation method is then to minimize the cost function of Eq. (22) subject to the algebraic constraints of Eqs. (23)–(25).

5 ph-Adaptive Mesh Refinement Method

We now develop a ph-adaptive mesh refinement method that using the LGR collocation method described in Section 4. We call our method a ph-method since we first try to adjust the polynomial degree to achieve convergence, and if this fails, we adjust the mesh spacing. The ph-adaptive mesh refinement method developed in this paper is divided into two parts. In Section 5.1, the method for estimating the error in the current solution is derived, and in Section 5.4, the p-then-h strategy is developed for refining the mesh.

5.1 Error Estimate in Each Mesh Interval

In this Section an estimate of the relative error in the solution within a mesh interval is derived. Because the state is the only quantity in the LGR collocation method for which a uniquely defined function approximation is available, we develop an error estimate for the state. The error estimate 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 estimate for the error in the state.

Assume that the NLP of Eqs. (22)–(25) corresponding to the discretized 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 we want to estimate 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 given in Eq. (20) 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 interpolat-

ing polynomial

$$\mathbf{u}^{(k)}(\tau) = \sum_{j=1}^{N_k} \mathbf{u}_j^{(k)} \hat{\ell}_j^{(k)}(\tau), \quad \hat{\ell}_j^{(k)}(\tau) = \prod_{\substack{l=1 \\ l \neq j}}^{N_k} \frac{\tau - \tau_l^{(k)}}{\tau_j^{(k)} - \tau_l^{(k)}}, \tag{27}$$

and let the control approximation at $\hat{\tau}_i^{(k)}$ be denoted $\mathbf{u}(\hat{\tau}_i^{(k)})$, $1 \leq i \leq M_k$. We use the value of the right-hand side of the dynamics at $(\mathbf{y}(\hat{\tau}_i^{(k)}), \mathbf{u}(\hat{\tau}_i^{(k)}), \hat{\tau}_i^{(k)})$ 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 $\hat{\tau}_i^{(k)}$, $1 \leq i \leq M_k$, then we have

$$\hat{\mathbf{y}}^{(k)}(\hat{\tau}_j^{(k)}) = \mathbf{y}^{(k)}(\tau_{k-1}) + \frac{t_f - t_0}{2} \sum_{l=1}^{M_k} \hat{I}_{jl}^{(k)} \mathbf{a}\left(\mathbf{y}^{(k)}(\hat{\tau}_l^{(k)}), \mathbf{u}^{(k)}(\hat{\tau}_l^{(k)}), \hat{\tau}_l^{(k)}\right), \quad j = 2, \ldots, M_k + 1, \tag{28}$$

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 i^{th} component of the state at $(\hat{\tau}_1^{(k)}, \ldots, \hat{\tau}_{M_k+1}^{(k)})$ are then defined, respectively, as

$$\begin{aligned} E_i^{(k)}(\hat{\tau}_l^{(k)}) &= \left|\hat{Y}_i^{(k)}(\hat{\tau}_l^{(k)}) - Y_i^{(k)}(\hat{\tau}_l^{(k)})\right|, \\ e_i^{(k)}(\hat{\tau}_l^{(k)}) &= \frac{E_i^{(k)}(\hat{\tau}_l^{(k)})}{1 + \max\limits_{j \in [1,\ldots,M_k+1]} \left|Y_i^{(k)}(\hat{\tau}_j^{(k)})\right|}, \end{aligned} \qquad \left[\begin{array}{c} l = 1, \ldots, M_k + 1, \\ i = 1, \ldots, n_y, \end{array}\right]. \tag{29}$$

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

$$e_{\max}^{(k)} = \max_{\substack{i \in [1,\ldots,n_y] \\ l \in [1,\ldots,M_k+1]}} e_i^{(k)}(\hat{\tau}_l^{(k)}). \tag{30}$$

5.2 Rationale for Error Estimate

The error estimate derived in Section 5.1 is similar to the error estimate obtained using the modified Euler Runge-Kutta scheme to numerically solve a differential equation $\dot{y}(t) = f(y(t))$. The first-order Euler method is given as

$$y_{j+1} = y_j + hf(y_j), \tag{31}$$

where h is the step-size and y_j is the approximation to $y(t)$ at $t = t_j = jh$. In the second-order modified Euler Runge-Kutta method, the first stage generates the following approximation $\bar{y}$ to $y(t_{j+1/2})$:

$$\bar{y} = y_j + \tfrac{1}{2} hf(y_j).$$

The second stage then uses the dynamics evaluated at $\bar{y}$ to obtain an improved estimate $\hat{y}_{j+1}$ of $y(t_{j+1})$:

$$\hat{y}_{j+1} = y_j + hf(\bar{y}). \tag{32}$$

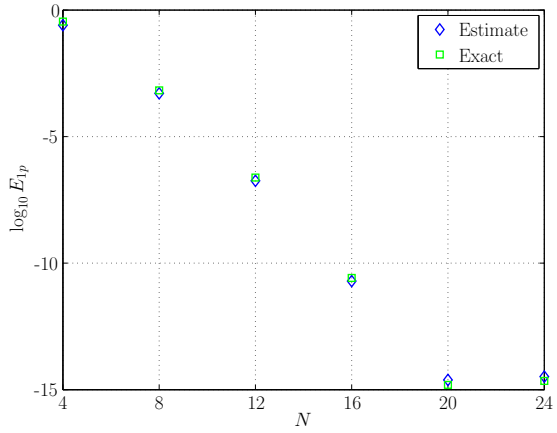
The original Euler scheme starts at y_j and generates y_{j+1}. The first-stage variable $\bar{y}$ is the interpolant of the line (first degree polynomial) connecting (t_j, y_j) and (t_{j+1}, y_{j+1}) evaluated at the new point $t_{j+1/2}$. The second-stage given Eq. (32) uses the dynamics at the interpolant $\bar{y}$ to obtain an improved approximation to $y(t_{j+1})$. Because $\hat{y}_{j+1}$ is a second-order approximation to $y(t_{j+1})$ and y_{j+1} is a first-order approximation to y_{j+1}, the absolute difference $|\hat{y}_{j+1} - y_{j+1}|$ is an estimate of the error in y_{j+1} in a manner similar to the absolute error estimate $E_i^{(k)}(\hat{\tau}_l^{(k)})$ $(l = 1, \ldots, M_k + 1)$ derived in Eq. (29).

The effectiveness of the derived error estimate derived in Section 5.1 can be seen by revisiting the motivating examples of Section 2. Figures 2a and 2b show the p and h error estimates, respectively, E_{1p} and E_{1h}, in the solution to Eq. (1), Figs. 2c and 2d show the p and h error estimates, respectively, E_{2p} and E_{2h}, in the solution to Eq. (2), and Fig. 2e shows the ph error estimates, E_{2ph} in the solution to Eq. (2). It is seen that the error estimates are nearly identical to the actual error. The relative error estimate given in Eq. (30) is used in the next section as the basis for modifying an existing mesh.

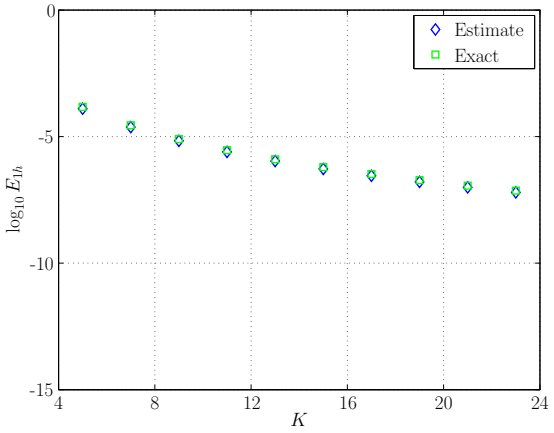
5.3 Estimation of Required Polynomial Degree within a Mesh Interval

Suppose again the LGR collocation NLP of Eqs. (22)–(25) has been solved on a mesh $\mathcal{S}_k,\ k = 1, \ldots, K$. Suppose further that it is desired to meet a relative error accuracy tolerance ϵ in each mesh interval $\mathcal{S}_k,\ k = 1, \ldots, K$. If the tolerance ϵ is not met in at least one mesh interval, then the next step is to refine the current mesh, either by dividing the mesh interval or increasing the degree of the approximating polynomial within the mesh interval.

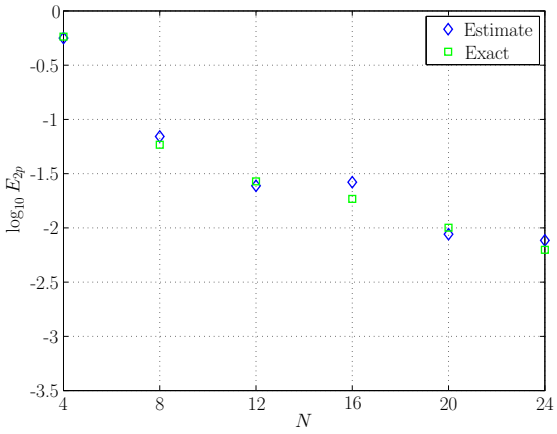
Consider a mesh interval $\mathcal{S}_q,\ q \in [1, \ldots, K]$ where N_q LGR points were used to solve the NLP of Eqs. (22)–(25), and again let ϵ be the desired relative error accuracy tolerance. Suppose further that the estimated maximum relative error, $e_{\max}^{(q)}$, has been computed as described in Section 5.1 and that $e_{\max}^{(q)} > \epsilon$ (that is, the accuracy tolerance ϵ is *not* satisfied in the existing mesh interval). Finally, let $N_{\min}$ and $N_{\max}$ be user-specified minimum and maximum bounds on the number of LGR points within any mesh interval. According to the convergence theory summarized



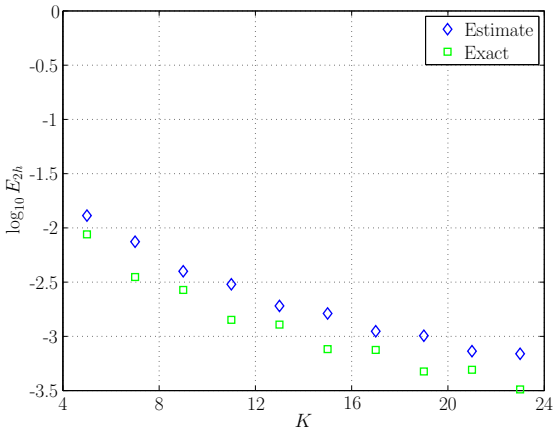
(a) $\log_{10} e_{1p}$ vs. N Using p Radau Method.



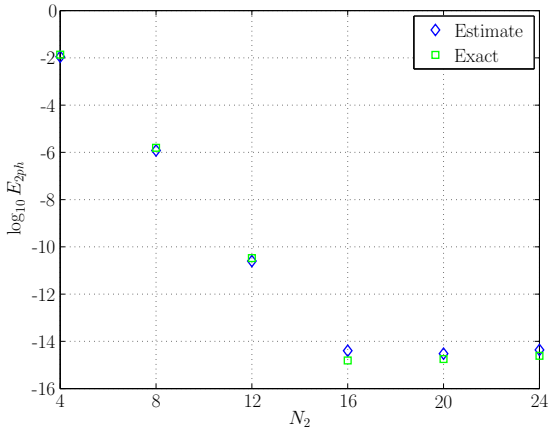
(b) $\log_{10} e_{1h}$ vs. K Using h Radau Method with $N = 4$.



(c) $\log_{10} e_{2p}$ vs. N Using p Radau Method.



(d) $\log_{10} e_{2h}$ vs. K Using h Radau Method with $N = 4$.



(e) $\log_{10} e_{2ph}$ vs. N_2 Using ph Radau Method.

Figure 2: Base-10 Logarithm of Absolute Error Estimates in Solutions of Eqs. (1) and (2) at Points $(\hat{\tau}_2, \ldots, \hat{\tau}_{M_k})$ Using p, h Methods, and ph Methods.

in,[34,35] the error in a global collocation scheme behaves like $O(N^{2.5-k})$ where N is the number of collocation points within a mesh interval and k is the number of continuous derivatives in the solution.[20,21] If the solution is smooth, then we could take $k = N$. Hence, if N was replace by $N + P$, then the error bound decreases by at least the factor N^{-P}.

Based on these considerations, suppose that interval $\mathcal{S}_q$ employs N_q collocation points and has relative error estimate $e_{\max}^{(q)}$ which is larger than the desired relative error tolerance ϵ; to reach the desired error tolerance, the error should be multiplied by the factor $\epsilon/e_{\max}^{(q)}$. This reduction is achieved by increasing N_q by P_q where P_q is chosen so that $N_q^{-P_q} = \epsilon/e_{\max}^{(q)}$, or equivalently,

$$N_q^{P_q} = \frac{e_{\max}^{(q)}}{\epsilon}.$$

This implies that

$$P_q = \log_{N_q}\left(\frac{e_{\max}^{(q)}}{\epsilon}\right). \tag{33}$$

Since the expression on the right side of (33) may not be an integer, we round up to obtain

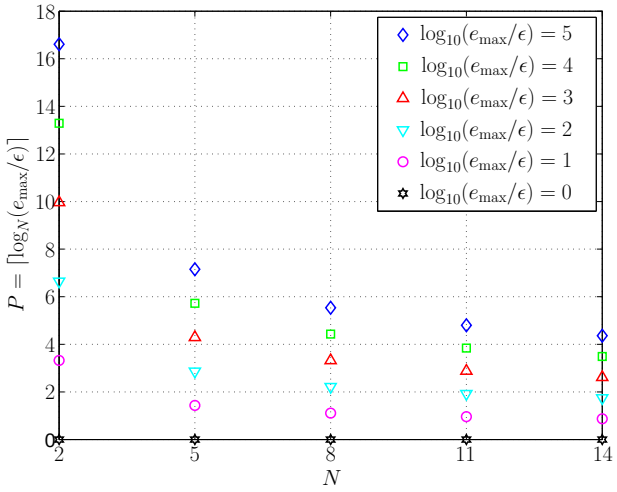
$$P_q = \left\lceil \log_{N_q}\left(\frac{e_{\max}^{(q)}}{\epsilon}\right) \right\rceil. \tag{34}$$

Note that $P_q \geq 0$ since we only use (34) when $e_{\max}^{(q)}$ is greater than the prescribed error tolerance ϵ. The dependence of P_q on N_q is shown in Fig. 3.

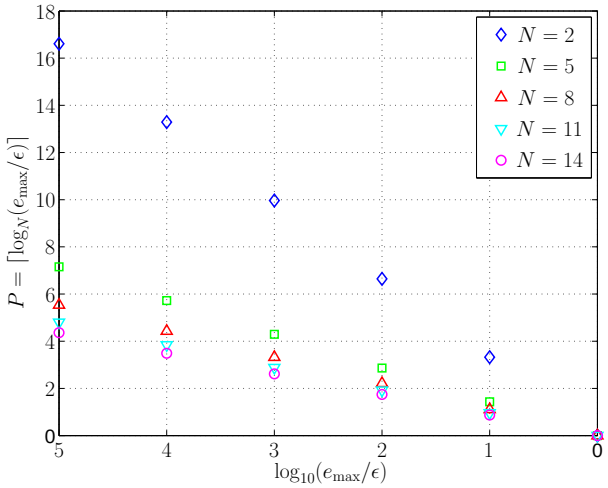
5.4 p-Then-h Strategy for Mesh Refinement

Using Eq. (34), the predicted number of LGR points required in mesh interval $\mathcal{S}_q$ on the ensuing mesh is $\tilde{N}_q = N_q + P_q$, assuming $e_{\max}^{(q)}$ has not reach the specified error tolerance ϵ. The only possibilities are that $\tilde{N}_q \leq N_{\max}$ (that is, $\tilde{N}$ does not exceed the maximum allowable polynomial degree) or that $\tilde{N}_q > N_{\max}$ (that is, $\tilde{N}$ exceeds the maximum allowable polynomial degree). If $\tilde{N}_q \leq N_{\max}$, then N_q is increased to $\tilde{N}_q$ on the ensuing mesh. If, on the other hand, $\tilde{N}_q > N_{\max}$, then $\tilde{N}_q$ *exceeds* the upper limit and the mesh interval $\mathcal{S}_q$ must be divided into sub-intervals.

Our strategy for mesh interval division uses the following approach. First, whenever a mesh interval is divided, the sum of the number of collocation points in the newly created mesh intervals should equal the predicted polynomial degree for the next mesh. Second, each newly created sub-interval should contain the minimum allowable number of collocation points. In other



(a) $P = \lceil \log_N(e_{\max}/\epsilon) \rceil$ vs. N for Fixed Values of $\log_{10}(e_{\max}/\epsilon)$.



(b) $P = \lceil \log_N(e_{\max}/\epsilon) \rceil$ vs. $\log_{10}(e_{\max}/\epsilon)$ for Fixed Values of N.

Figure 3: Function That Relates the Increase in the Degree of the Approximating Polynomial to the Ratio $e_{\max}/\epsilon$ and the Current Polynomial Degree, N.

words, if a mesh interval $\mathcal{S}_q$ is divided into B_q subintervals, then each newly created subinterval will contain $N_{\min}$ collocation points and the sum of the collocation points in these newly created subintervals should be $B_q N_{\min}$. Using this strategy, the number of subintervals, B_q, into which $\mathcal{S}_q$ is divided is computed as

$$B_q = \max\left(\left\lceil \frac{\tilde{N}_q}{N_{\min}} \right\rceil, 2\right), \tag{35}$$

where it is seen in Eq. (35) that $2 \leq B_q \leq \lceil \tilde{N}_q / N_{\min} \rceil$. It is seen that this strategy for mesh interval division ensures that the same total number of collocation points is the same regardless of whether the polynomial degree in a mesh interval is increased or the mesh interval is refined. Second, because the number of LGR points in a newly created mesh interval is started at $N_{\min}$, the method uses the full range of allowable values of N. Because of the hierarchy, the ph method of this paper can be thought of more precisely as a "p-then-h" method where p refinement is exhausted prior to performing any h refinement. In other words, the polynomial degree within a mesh interval is increased until the upper limit $N_{\max}$ is exceeded. The h refinement (mesh interval division) is then performed after which the p refinement is restarted.

It is important to note that the ph method developed in this paper can be employed as a fixed-order h method simply by setting $N_{\min} = N_{\max}$. The h version of the method of this paper is similar to an adaptive step-size fixed-order integration method, such as an adaptive step-size

Runge-Kutta method, in the following respect: In both cases, the mesh is refined, often by step-halving or step-doubling,[36] when the specified error tolerance is not met.

A summary of our adaptive mesh refinement algorithm appears below. Here M denotes the mesh refinement iteration, and in each loop of the algorithm, the mesh number increases by 1. The algorithm terminates in Step 4 when the error tolerance is satisfied or when M reaches a prescribed maximum $M_{\max}$.

ph-Adaptive Mesh Refinement Method

Step 1: Set $M = 0$ and supply initial mesh, $\mathcal{S} = \bigcup_{k}^{K} \mathcal{S}_k = [-1, +1]$, where $\bigcap_{k}^{K} \mathcal{S}_k = \emptyset$.

Step 2: Solve Radau collocation NLP of Eqs. (22)–(25) on current mesh $\mathcal{S}$.

Step 3: Compute scaled error $e_{\max}^{(k)}$ in $\mathcal{S}_k,\ k = 1, \ldots, K$, using method of Section 5.1.

Step 4: If $e_{\max}^{(k)} \leq \epsilon$ for all $k \in [1, \ldots, K]$ or $M > M_{\max}$, then quit. Otherwise, proceed to **Step 5**.

Step 5: Using the method of Section 5.4, modify all mesh intervals $\mathcal{S}_k$ for which $e_{\max}^{(k)} > \epsilon$.

Step 6: Set $M \stackrel{\rightarrow}{=} M + 1$, and return to **Step 2**.

6 Examples

In this Section the ph-adaptive Legendre-Gauss-Radau (LGR) method described in Section 5 is applied to three examples from the open literature. The first example is a variation of the hyper-sensitive optimal control problem originally described in Ref. 37, where the effectiveness of the error estimate derived in Section 5.1 is demonstrated and the improved efficiency of the ph method over various h methods is shown. The second example is a tumor anti-angiogenesis optimal control problem originally described in Ref. 38, where it is seen that the ph method of this paper accurately and efficiently captures a discontinuity in a problem whose optimal control is discontinuous. The third example is the reusable launch vehicle entry problem from Ref. 3, where it is seen that using the ph method of this paper leads to a significantly smaller mesh than

would be obtained using an h method. This third example also shows that allowing $N_{\min}$ to be too small can reduce the effectiveness of the ph method.

When using a ph-adaptive method, the terminology ph-$(N_{\min}, N_{\max})$ refers to the ph-adaptive method of this paper where the polynomial degree can vary between $N_{\min}$ and $N_{\max}$, respectively, while an h-N method refers to an h method with a polynomial of fixed degree N. For example, a ph-$(2, 8)$ method is a ph-adaptive method where $N_{\min} = 2$ and $N_{\max} = 8$, while an h-2 method is an h method where $N = 2$. All results were obtained using the optimal control software $\mathbb{GPOPS} - \mathbb{II}$[39] running with the NLP solver IPOPT[40] in second derivative mode with the multifrontal massively parallel sparse direct solver MUMPS,[41] default NLP solver tolerances, and a mesh refinement accuracy tolerance $\epsilon = 10^{-6}$. The initial mesh for a ph-$(N_{\min}, N_{\max})$ or h-$N_{\min}$ method consisted of ten uniformly-spaced mesh intervals with $N_{\min}$ LGR points in each interval, while the initial guess was a straight line between the known initial conditions and known terminal conditions for the problem under consideration with the guess on all other variables being a constant. The required first and second derivatives required by IPOPT were computed using the built-in sparse first and second finite-differencing method in $\mathbb{GPOPS} - \mathbb{II}$ that uses the method of Ref. 19. Finally, all computations were performed on a 2.5 GHz Intel Core i7 MacBook Pro running Mac OS-X Version 10.7.5 (Lion) 16 GB of 1333 MHz DDR3 RAM and MATLAB Version R2012b. The central processing unit (CPU) times reported in this paper are ten-run averages of the execution time.

Example 1: Hyper-Sensitive Problem

Consider the following variation of the *hyper-sensitive* optimal control problem.[37] Minimize the cost functional

$$J = \frac{1}{2}\int_0^{t_f} (x^2 + u^2)dt \tag{36}$$

subject to the dynamic constraint

$$\dot{x} = -x + u \tag{37}$$

and the boundary conditions

$$x(0) = 1.5 \quad , \quad x(t_f) = 1, \tag{38}$$

where t_f is fixed. It is known that for sufficiently large values of t_f that the solution to the hyper-sensitive problem exhibits a so called "take-off", "cruise", and "landing" structure where all of the interesting behavior occurs near the "take-off" and "landing" segments while the solution is essentially constant in the "cruise" segment. Furthermore, the "cruise" segment becomes and increasingly large percentage of the total trajectory time as t_f increases, while "take-off" and "landing" segments have rapid exponential decay and growth, respectively. The analytic optimal state and control for this problem are given as

$$\begin{aligned} x^*(t) &= c_1 \exp(t\sqrt{2}) + c_2 \exp(-t\sqrt{2}), \\ u^*(t) &= \dot{x}^*(t) + x^*(t), \end{aligned} \tag{39}$$

where

$$\left[\begin{array}{c} c_1 \\ c_2 \end{array} \right] = \frac{1}{\exp(-t_f\sqrt{2}) - \exp(t_f\sqrt{2})} \left[\begin{array}{c} 1.5\exp(-t_f\sqrt{2}) - 1 \\ 1 - 1.5\exp(t_f\sqrt{2}) \end{array} \right]. \tag{40}$$

Figures 4a and 4b shows the exact state and control for the hyper-sensitive problem with $t_f = 10000$ and highlight the "take-off", "cruise", and "landing" feature of the optimal solution. Given the structure of the optimal solution, it should be the case that a mesh refinement method place many more collocation and mesh points near the ends of the time interval when t_f is large. Figures 4c shows the evolution of the mesh points T_k while Figure 4d shows the evolution collocation (LGR) points $\tau_j^{(k)}$ on each mesh refinement iteration using the ph-$(3, 14)$ scheme. Two key related features are seen in the mesh refinement. First, Fig. 4c shows that mesh intervals are added on each refinement iteration only in the regions near $t = 0$ and $t = t_f$, while mesh intervals are *not* added in the interior region $t \in [1000, 9000]$. Second, Fig. 4d shows that after the first mesh refinement iteration LGR points are also added only in the regions regions near $t = 0$ and $t = t_f$ and are not added in the interior region $t \in [1000, 9000]$. This behavior of the ph-adaptive method shows that error reduction is achieved by added mesh and collocation points in regions of $t \in [0, t_f]$ where points are needed to capture the changes in the solution. Finally, for comparison with the ph-adaptive method, Figs. 4e and 4f show the solution obtained using an h-2 method. Unlike the ph-$(3, 14)$ method, where mesh points are added only where needed to meet the accuracy tolerance, the h-2 method places many more mesh points over much larger segments at the start and end of the overall time interval. Specifically, it is seen that the mesh is quite dense over time intervals $t \in [0, 3000]$ and $t \in [7000, 10000]$ whereas for the ph-$(3, 14)$ method, the mesh remains dense over the smaller intervals $[0, 1000]$ and $[9000, 10000]$.

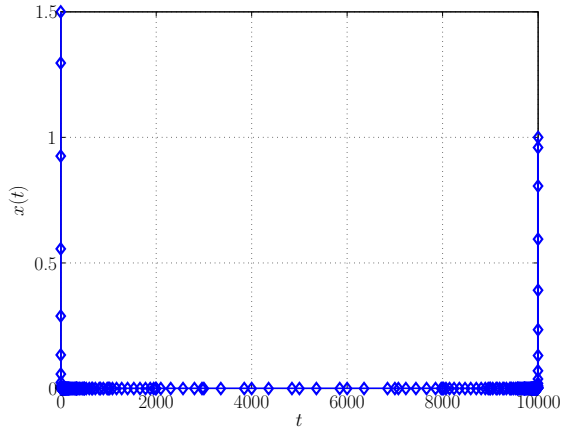
Table 1: Estimated Relative State Error, $e_x^{\max}$, Exact Relative State Error, $e_{x,\text{exact}}^{\max}$, and Exact Relative Control Error, $e_{u,\text{exact}}^{\max}$, for Example 1 with $t_f = 10000$ Using a ph-$(3, 14)$ with an Accuracy Tolerance $\epsilon = 10^{-6}$.

M	$e_x^{\max}$	$e_{x,\text{exact}}^{\max}$	$e_{u,\text{exact}}^{\max}$
1	3.377×10^{0}	5.708×10^{-3}	1.280×10^{0}
2	6.436×10^{-1}	4.009×10^{-2}	1.127×10^{0}
3	9.648×10^{-2}	7.462×10^{-2}	3.369×10^{-1}
4	8.315×10^{-10}	1.016×10^{-9}	1.329×10^{-8}

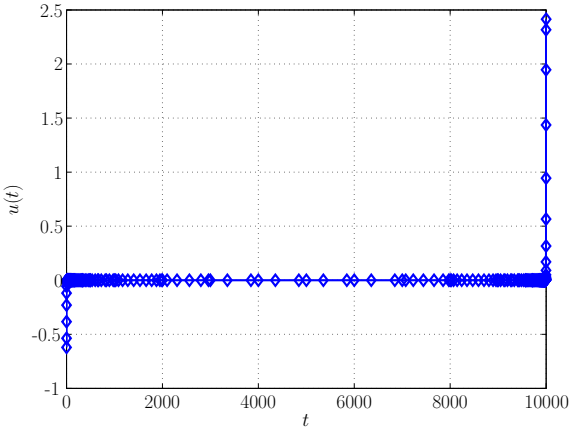
Admittedly, the ph-$(3, 14)$ does add LGR points in the regions $t \in [1000, 3000]$ and $t \in [7000, 9000]$ whereas the h-2 method adds more mesh intervals, but the mesh obtained using the ph-$(3, 14)$ is much smaller (273 collocation points) than the mesh obtained using the h-2 method (672 collocation points). Thus the ph-method exploits the solution smoothness on the intervals $[1000, 3000]$ and $[7000, 9000]$ to achieve more rapid convergence by increasing the degree of the approximating polynomials instead of increasing the number of mesh intervals.

Next, we analyze the quality of the error estimate of Section 5.1 by examining more closely the numerical solution near $t = 0$ and $t = t_f$. Figures 5a and 5b show the state and control in the regions $t \in [0, 15]$ and $t \in [9985, 10000]$ on each mesh refinement iteration alongside the exact solution using the ph-$(3, 14)$ method, while Table 1 shows the estimated and exact relative errors in the state and the exact relative error in the control for each mesh refinement iteration. First, it is seen in Table 1 that the state and control relative error on the final mesh is quite small at $\approx 10^{-9}$ for the state and $\approx 10^{-8}$ for the control. In addition, it is seen from Figs. 5a and 5b that the state and control approximations improve with each mesh refinement iteration. Moreover, the error estimate shown in Table 1 agrees qualitatively with the solutions on the corresponding mesh as shown in Figs. 5a and 5b. It is also interesting to see that the state relative error estimate is approximately the same on each mesh iteration as the exact relative error. The consistency in the relative error approximation and the exact relative error demonstrates the accuracy of the error estimate derived in Section 5.1. Thus, the error estimate derived in this paper reflects correctly the locations where the solution error is large and ph-adaptive method constructs new meshes that reduce the error without making the mesh overly dense.

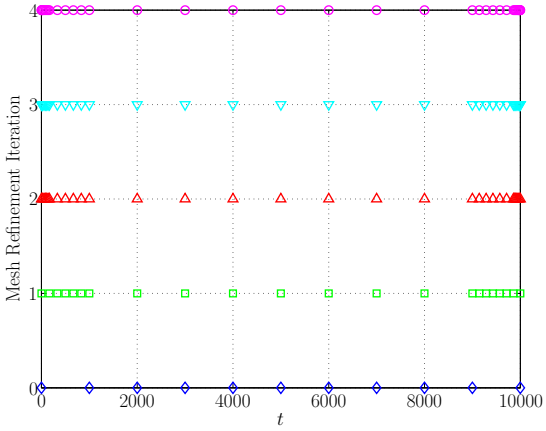
Finally, we provide a comparison of the computational efficiency and mesh sizes obtained



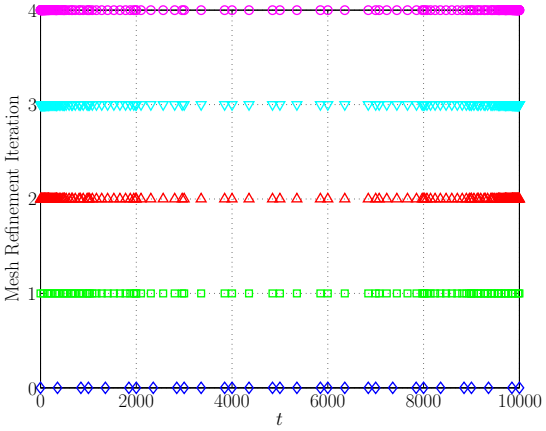
(a) $x(t)$ vs. t for $t_f = 10000$.



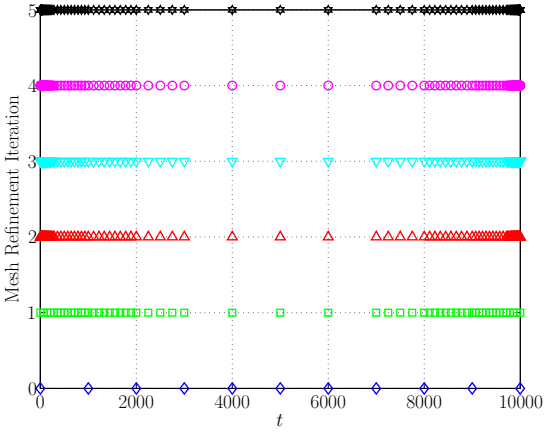
(b) $u(t)$ vs. t.



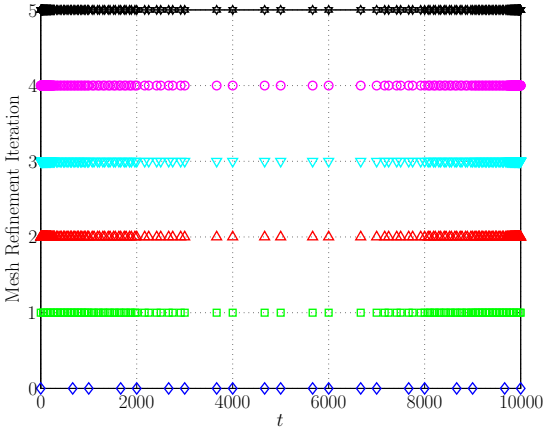
(c) Mesh Point History Using ph-$(3, 14)$ Method.



(d) Collocation Point History Using ph-$(3, 14)$ Method.

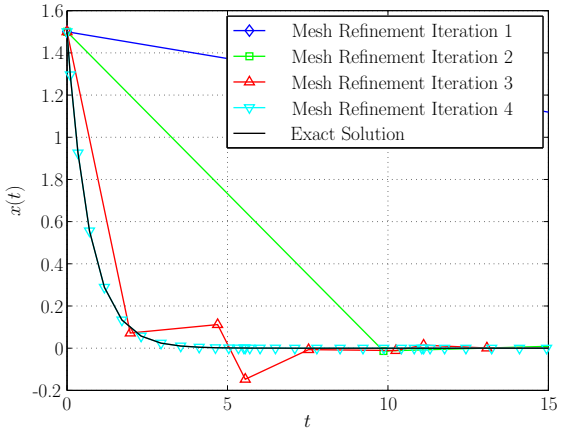


(e) Mesh Point History Using h-2 Method.

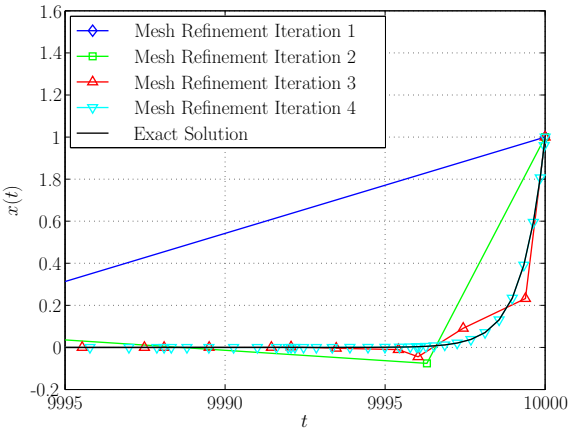


(f) Collocation Point History Using h-2 Method.

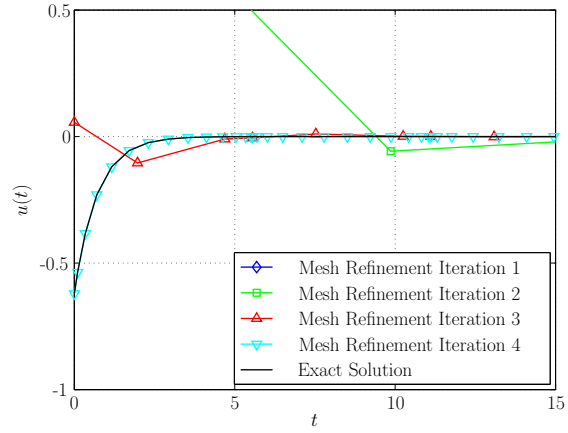
Figure 4: Exact Solution to Example 1 with $t_f = 10000$ and Mesh Refinement History When Using the ph-$(3, 14)$ and h-2 Methods with an Accuracy Tolerance $\epsilon = 10^{-6}$.



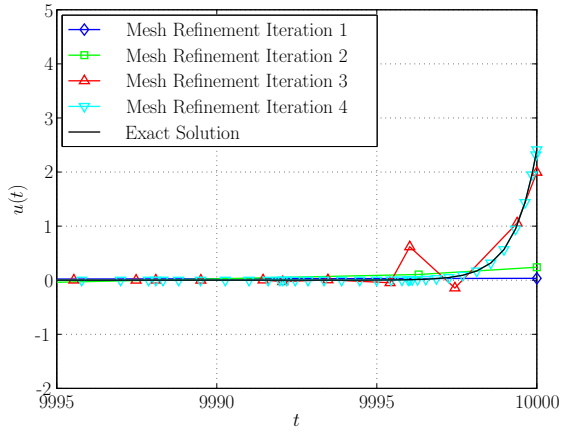
(a) $x(t)$ vs. t Near $t = 0$ for $t_f = 10000$.



(b) $x(t)$ vs. t Near $t = t_f$ for $t_f = 10000$.



(c) $x(t)$ vs. t Near $t = 0$ for $t_f = 10000$.



(d) $x(t)$ vs. t Near $t = t_f$ for $t_f = 10000$.

Figure 5: Solution Near Endpoints of $t \in [0, t_f]$ for Example 1 with $t_f = 10000$ Using the ph-$(3, 14)$ and an Accuracy Tolerance $\epsilon = 10^{-6}$.

by solving Example 1 using the various ph-adaptive and h methods described Section 5. Table 2 shows the CPU times and mesh sizes, where it is seen for this example that the ph-$(3, 14)$ and h-2 methods result in the smallest overall CPU times (with the ph-$(3, 14)$ being slightly more computationally efficient than the h-2 method). Interestingly, while the ph-$(3, 14)$ and h-2 methods have nearly the same computational efficiency, the ph-$(3, 14)$ produces a significantly smaller mesh ($N = 293$ LGR points, nearly the smallest amongst all of the methods) while the h-2 mesh produced a much larger mesh ($N = 672$ LGR points, by far the largest amongst all of the different methods). In fact Table 2 shows for this example that, for any fixed value $N_{\min}$, the ph-$(N_{\min}, N_{\max})$ methods produced smaller mesh sizes than the corresponding h-$N_{\min}$ method. Thus, while an h method may perform well on this example due to the structure of the optimal solution, the ph method produces the solution in the most computationally efficient manner while simultaneously producing a significantly smaller mesh.

Example 2: Tumor Anti-Angiogenesis Optimal Control Problem

Consider the following tumor anti-angiogenesis optimal control problem taken from Ref. 38. The objective is to minimize

$$J = y_1(t_f) \tag{41}$$

subject to the dynamic constraints

$$\begin{aligned} \dot{y}_1(t) &= -\xi y_1(t) \ln\left(\frac{y_1(t)}{y_2(t)}\right), \\ \dot{y}_2(t) &= q(t)\left[b - \mu - d y_1^{2/3}(t) - G u(t)\right], \end{aligned} \tag{42}$$

with the initial conditions

$$\begin{aligned} y_1(0) &= \left[(b-\mu)/d\right]^{3/2}/2, \\ y_2(0) &= \left[(b-\mu)/d\right]^{3/2}/4, \end{aligned} \tag{43}$$

the control constraint

$$0 \leq u \leq u_{\max}, \tag{44}$$

and the integral constraint

$$\int_0^{t_f} u(\tau) d\tau \leq A, \tag{45}$$

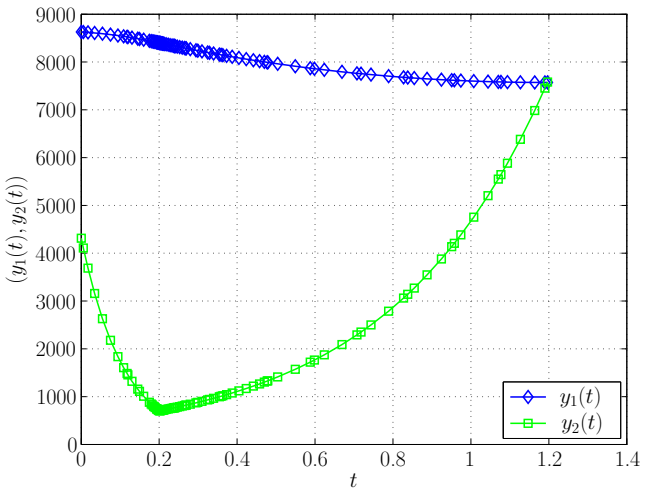
where $G = 0.15$, $b = 5.85$, $d = 0.00873$, $\mu = 0.02$, $u_{\max} = 75$, $A = 15$, and t_f is free. A solution to this optimal control problem is shown using the ph-$(3, 10)$ method is shown in Figs. 6a and 6b

Table 2: Mesh Refinement Results for Example 1 Using Various ph-Adaptive and h Methods.

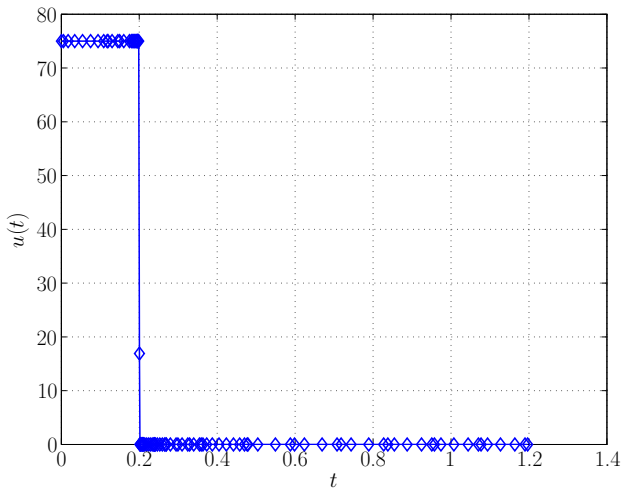
$N_{\min}$	$N_{\max}$	Initial Mesh CPU Time (s)	Mesh Refinement CPU Time (s)	Total CPU Time (s)	N	K	M	Constraint Jacobian Density (%)
2	2	0.34	3.10	3.44	672	336	5	0.629
2	8	0.33	6.93	7.26	633	222	5	0.908
2	10	0.33	6.77	7.10	624	210	5	0.971
2	12	0.33	6.12	6.45	583	170	6	1.271
2	14	0.33	5.17	5.50	520	140	5	1.672
2	16	0.33	6.33	6.66	519	126	4	1.839
3	3	0.33	3.28	3.61	417	139	6	1.229
3	8	0.32	3.89	4.22	369	96	6	1.748
3	10	0.32	3.43	3.75	347	79	5	2.165
3	12	0.33	3.79	4.12	343	64	5	2.606
3	**14**	**0.33**	**3.05**	**3.38**	**293**	**40**	**4**	**3.847**
3	16	0.33	3.49	3.81	290	41	5	3.825
4	4	0.34	3.86	4.20	384	96	7	1.580
4	8	0.33	3.65	3.98	324	66	6	2.269
4	10	0.33	3.37	3.70	311	56	6	2.637
4	12	0.33	3.69	4.02	291	43	5	3.454
4	14	0.33	4.31	4.64	306	39	7	3.599
4	16	0.33	4.95	5.29	320	44	7	3.413

Upon closer examination, it is seen that a key feature in the optimal solution is the fact that optimal control is discontinuous at $t \approx 0.2$. In order to improve the accuracy of the solution in the vicinity of this discontinuity, it is necessary that an increased number of collocation and mesh points are placed near $t = 0.2$. Figure 6b shows the control obtained on the final mesh by the ph-$(3, 10)$ method. Interestingly, it is seen that the ph-$(3, 10)$ method concentrates the collocation and mesh points near $t = 0.2$. Examining the evolution of the mesh refinement, it is seen in Figs. 6c and 6d that the mesh density increases on each successive mesh iteration, but remains unchanged in regions distant from $t \approx 0.2$. The reason that the mesh is modified near $t = 0.2$ is because the accuracy of the state is lowest in the region near the discontinuity in the control. In order to improve solution accuracy, additional collocation and mesh points are required near $t = 0.2$. Thus, the ph-method performs properly when solving this problem as it leaves the mesh untouched in regions where few collocation and mesh points are needed, and it increases the density of the mesh where additional points are required.

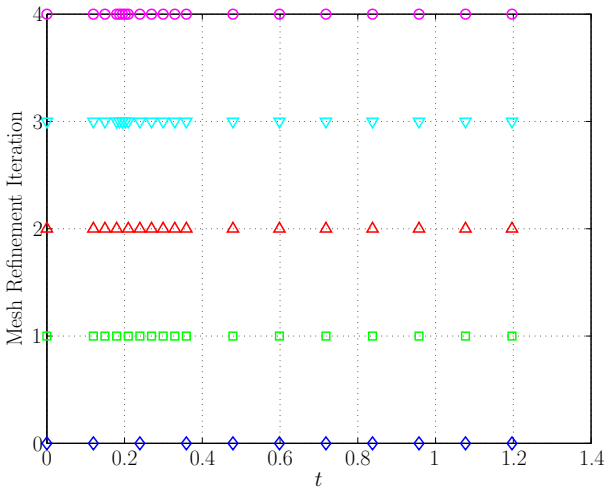
Next, Table 3 summarizes the CPU times and mesh sizes that were obtained by solving Example 2 using the various ph and h methods described Section 5. While for this example the CPU times are quite small, it is still seen that computational efficiency is gained by choosing a ph method over an h method. Specifically, it is seen that the ph-$(3, 10)$ method produces the lowest CPU time with an h-3 method being slightly less efficient than the ph-$(3, 10)$ method. More importantly, Table 3 shows the significant reduction in mesh size when using a ph method. For example, using a ph-$(3, N_{\max})$ or ph-$(4, N_{\max})$, the maximum number of LGR points is $N = 99$ whereas the lowest number of LGR points using either an h-2, h-3, or h-4 method is $N = 116$. Moreover, while the ph-$(3, 14)$ and h-2 methods have nearly the same computational efficiency, the ph-$(3, 14)$ produces a significantly smaller mesh ($N = 293$ LGR points, nearly the smallest amongst all of the methods) while the h-2 mesh produced a much larger mesh ($N = 672$ LGR points, by far the largest amongst all of the different methods). Thus, while an h method may perform well on this example due to the structure of the optimal solution, the ph method produces the solution in the most computationally efficient manner while simultaneously producing the smallest mesh.



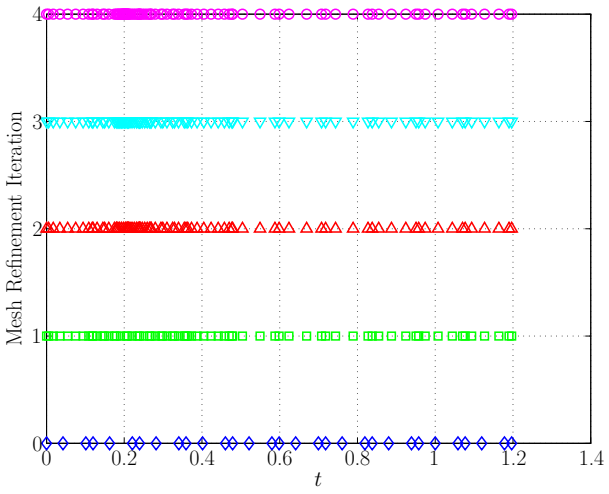
(a) $(p(t), q(t))$ vs. t.



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



(c) Mesh Point History.



(d) Collocation (LGR) Point History.

Figure 6: Solution and Mesh Refinement History for Example 2 Using the ph-$(3, 10)$ Method with an Accuracy Tolerance of 10^{-6}.

Table 3: Mesh Refinement Results for Example 2 Using Various ph-Adaptive and h Methods.

$N_{\min}$	$N_{\max}$	Initial Mesh CPU Time (s)	Mesh Refinement CPU Time (s)	Total CPU Time (s)	N	K	M	Constraint Jacobian Density (%)
2	2	0.11	0.72	0.83	270	135	3	0.979
2	8	0.10	0.88	0.98	245	84	4	1.569
2	10	0.10	1.02	1.12	237	65	5	2.088
2	12	0.10	0.75	0.84	204	40	4	3.354
2	14	0.10	0.74	0.84	204	40	4	3.354
2	16	0.10	0.69	0.78	213	26	4	4.041
3	3	0.17	0.55	0.72	123	41	4	2.746
3	8	0.17	0.71	0.88	104	27	6	4.114
3	**10**	**0.17**	**0.50**	**0.67**	**95**	**19**	**4**	**6.040**
3	12	0.17	0.86	1.03	99	18	7	6.310
3	14	0.17	0.62	0.80	89	15	6	7.527
3	16	0.17	1.56	1.73	99	15	13	7.955
4	4	0.16	0.90	1.05	116	29	6	3.587
4	8	0.16	0.70	0.85	87	18	6	5.898
4	10	0.16	0.96	1.12	85	16	9	6.702
4	12	0.16	0.88	1.04	82	16	10	6.929
4	14	0.16	1.11	1.27	83	16	13	7.047
4	16	0.16	1.01	1.16	87	14	10	8.401

Example 3: Reusable Launch Vehicle Entry

Consider the following optimal control problem from Ref. 3 of maximizing the crossrange during the atmospheric entry of a reusable launch vehicle. Minimize the cost functional

$$J = -\phi(t_f) \tag{46}$$

subject to the dynamic constraints

$$\begin{aligned} &\dot{r} = v\sin\gamma \quad , \quad \dot{\theta} = \frac{v\cos\gamma\sin\psi}{r\cos\phi} \quad , \quad \dot{\phi} = \frac{v\cos\gamma\cos\psi}{r}, \\ &\dot{v} = -\frac{D}{m} - g\sin\gamma \quad , \quad \dot{\gamma} = \frac{L\cos\sigma}{mv} - \left(\frac{g}{v} - \frac{v}{r}\right)\cos\gamma \quad , \quad \dot{\psi} = \frac{L\sin\sigma}{mv\cos\gamma} + \frac{v\cos\gamma\sin\psi\tan\phi}{r}, \end{aligned} \tag{47}$$

and the boundary conditions

$$\begin{aligned} &r(0) = r_0 \quad , \quad r(t_f) = r_f \quad , \quad \theta(0) = \theta_0 \quad , \quad \theta(t_f) = \text{Free}, \\ &\phi(0) = \theta_f \quad , \quad \phi(t_f) = \text{Free} \quad , \quad v(0) = v_0 \quad , \quad v(t_f) = v_f, \\ &\gamma(0) = \gamma_0 \quad , \quad \gamma(t_f) = \gamma_f \quad , \quad \psi(0) = \psi_0 \quad , \quad \psi(t_f) = \text{Free}, \end{aligned} \tag{48}$$

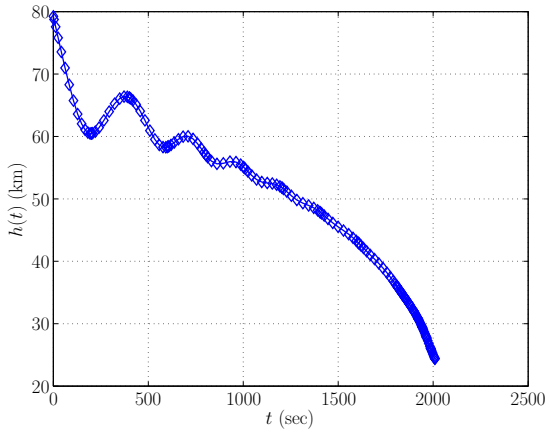
It is noted that the model and the numerical values $(r_0, r_f, \theta_0, \theta_f, v_0, v_f, \gamma_0, \gamma_f, \psi_0)$ are taken from Ref. 3 with the exception that all quantities in Ref. 3 are given in English units while the values used in this example are in SI units. A typical solution of this problem is shown in Figs. 7a–7f using the ph-$(2, 14)$ method.

It is seen that the solution to this example is relatively smooth; although there seems to be a rapid change in the angle of attack in Fig. 7e near $t = 2000$, the total deflection is at most one degree. As a result, one might hypothesize that it is possible to obtain an accurate solution with a relatively small number of collocation and mesh points when compared with an h method. This hypothesis is confirmed in Table 4 where several interesting trends are observed. First, it is seen that the ph-$(2, 14)$ and ph-$(2, 16)$ methods are the most computationally efficient. In particular, the ph-$(2, 14)$ and ph-$(2, 16)$ methods are 30 percent, 44 percent, and 61 percent faster, respectively, than the h-2, h-3, and h-4 methods. Also, it is seen that the ph-$(2, 14)$ and ph-$(2, 16)$ methods produce smaller meshes (with a total of 170 collocation points) when compared with the h-2, h-3, or h-4 methods (where the total numbers of collocation points are 486, 261, and 188, respectively). Next, Figs. 8a and 8b show the evolution of the meshes for the ph-$(2, 14)$ method, where it is seen that the number of mesh intervals increases from that of the initial mesh only at the very end of the trajectory due to the rapid change of the flight path angle near $t = t_f$. As

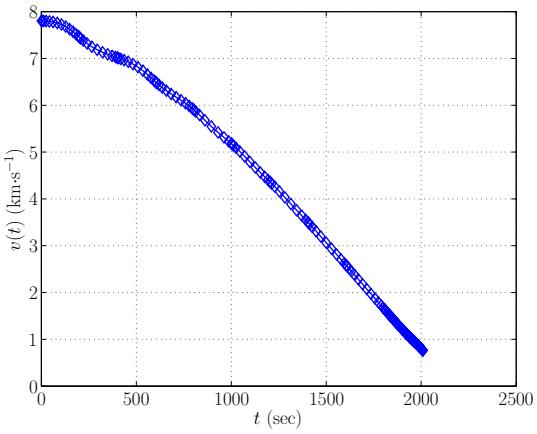
Table 4: Mesh Refinement Results for Example 3 Using Various ph-Adaptive and h Methods.

$N_{\min}$	$N_{\max}$	Initial Mesh CPU Time (s)	Mesh Refinement CPU Time (s)	Total CPU Time (s)	N	K	M	Constraint Jacobian Density (%)
2	2	0.76	1.56	2.33	486	243	3	0.314
2	8	0.76	1.55	2.30	442	86	3	0.871
2	10	0.76	2.22	2.98	404	60	4	1.061
2	12	0.75	1.33	2.08	227	30	3	2.207
2	**14**	**0.75**	**0.83**	**1.58**	**170**	**18**	**2**	**3.560**
2	**16**	**0.75**	**0.83**	**1.58**	**170**	**18**	**2**	**3.560**
3	3	1.31	1.53	2.84	261	87	4	0.795
3	8	1.31	1.09	2.41	195	38	3	1.740
3	10	1.31	1.16	2.47	114	13	4	4.762
3	12	1.31	0.78	2.10	98	10	3	5.965
3	14	1.32	0.78	2.10	98	10	3	5.965
3	16	1.32	0.78	2.10	98	10	3	5.965
4	4	3.19	0.90	4.09	188	47	3	1.396
4	8	3.20	0.80	4.00	127	24	3	2.766
4	10	3.19	0.68	3.87	97	12	3	5.134
4	12	3.20	0.43	3.64	89	10	2	6.025
4	14	3.21	0.44	3.65	89	10	2	6.025
4	16	3.19	0.43	3.63	89	10	2	6.025

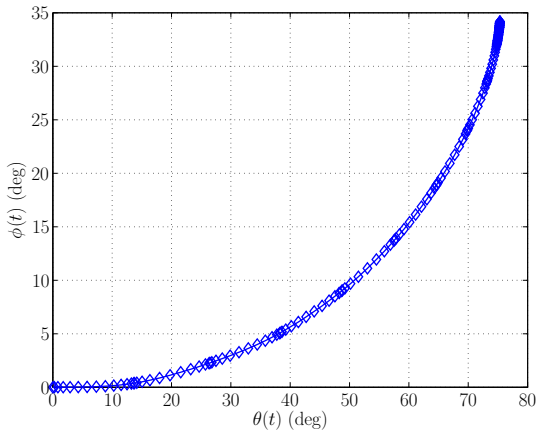
a result, for the vast majority of the solution the largest decrease in error is obtained by using a larger polynomial degree in each mesh interval and using fewer mesh intervals. In this case the fact that the ph-$(2, 14)$ and ph-$(2, 16)$ methods outperform the other methods (in particular, outperform the h methods) is consistent with the fact that the solution to this problem is smooth, having only relatively small oscillations in the altitude and flight path angle. Thus, as stated, the accuracy tolerance can be achieved by using relatively few mesh intervals with a high-degree polynomial approximation in each interval.



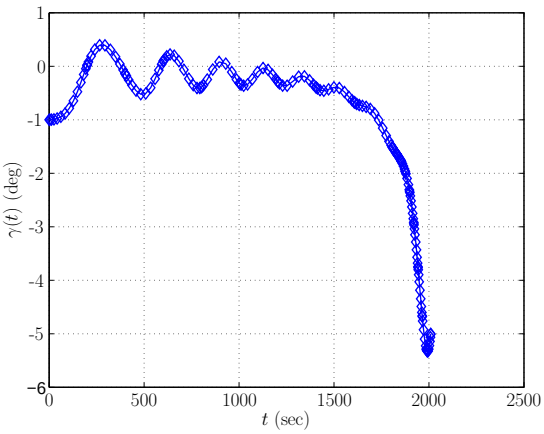
(a) $h(t)$ vs. t.



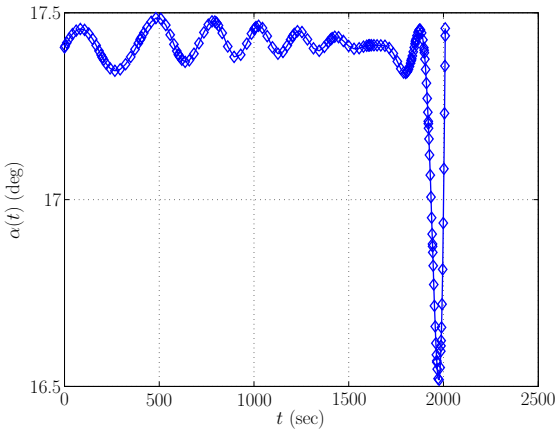
(b) $v(t)$ vs. t.



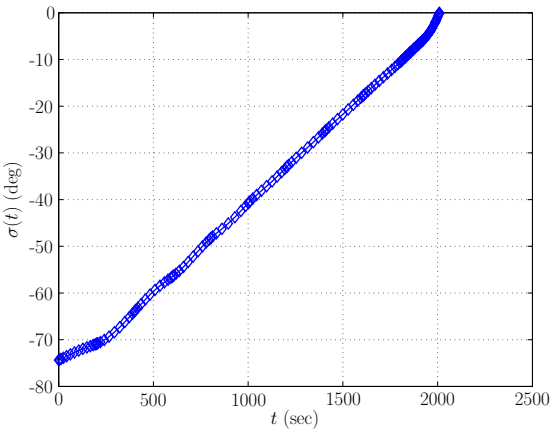
(c) $\phi(t)$ vs. $\theta(t)$.



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

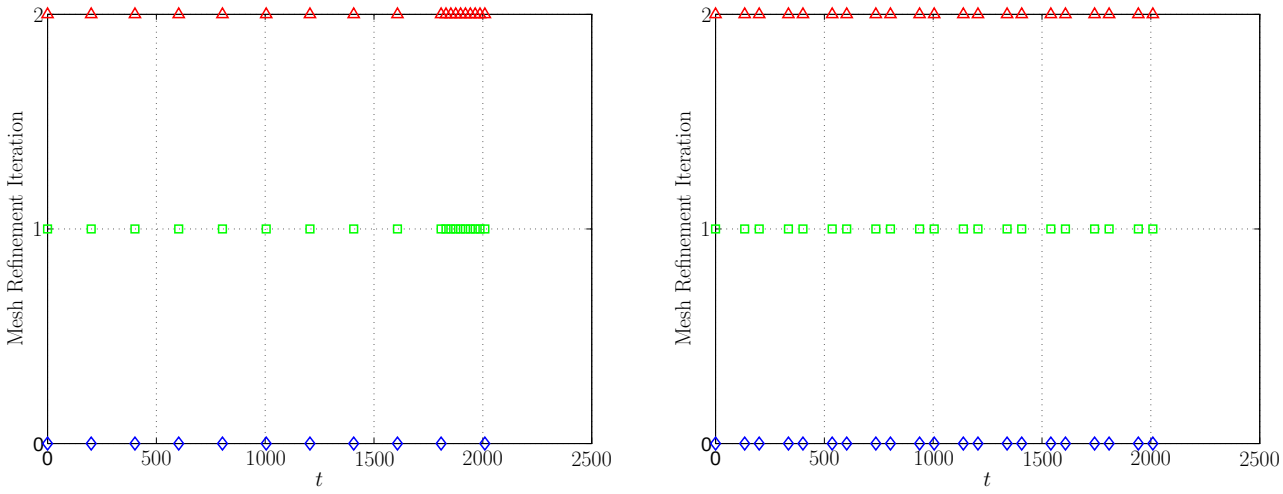


(e) $\alpha(t)$ vs. t.



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

Figure 7: Solution to Example 3 Using the ph-$(2, 14)$ Method with an Accuracy Tolerance of 10^{-6}.



(a) Mesh Point History.

(b) Collocation (LGR) Point History.

Figure 8: Mesh Refinement History for Example 3 Using the ph-$(2, 14)$ Method with an Accuracy Tolerance of 10^{-6}.

7 Discussion

Each of the examples illustrates different features of the ph-adaptive mesh refinement method developed in Section 5. The first example shows how the computational efficiency of the ph-adaptive adaptive mesh refinement scheme is similar to the computational efficiency of an h method while generating a much smaller mesh for a given accuracy tolerance than is required when using an h method. This first example also demonstrates the effectiveness of the error estimate derived in Section 5.1. The second example shows how the ph-adaptive method can efficiently capture a discontinuity in the solution by making the mesh more dense near the discontinuity while simultaneously not placing unnecessary mesh and collocation points in regions distant from the discontinuity. Furthermore, similar to the results obtained in the first example, the second example shows the significantly smaller mesh that is generated using the ph-adaptive method when compared with the mesh generated using an h method. Next, the third example demonstrates how the ph-adaptive method does not unnecessarily add mesh intervals when it is only necessary to increase the degree of the polynomial approximation to achieve a desired accuracy tolerance.

Next, the method of this paper takes advantage of the fact that in regions where the solution is smooth it is possible to gain significant accuracy by increasing the degree of the polynomial

approximation. On the other hand, if the estimated polynomial degree exceeds a given threshold (that is, N exceeds the upper limit $N_{\max}$) without satisfying the accuracy tolerance, then a further increase in the polynomial degree will improve the error in the solution only marginally. As a result, in such cases it is more beneficial to refine the mesh. Next, while the parameters $N_{\min}$ and $N_{\max}$ are somewhat arbitrary, it is seen from the three examples that choosing $N_{\min} = 3$ and $N_{\max} = 14$ or $N_{\max} = 16$ generally provides good performance when compared with an h method (for example, see Tables 2, 3, and 4 where it is seen that $N_{\min} = 3$ and $N_{\max} = 14$ or $N_{\max} = 16$ result in either the fastest or nearly the fastest computation times). Finally, the method of this paper has the potential advantage that the nonlinear programming problem (NLP) may be smaller in size when compared with that of an h method, requiring less memory than might be required to achieve the same accuracy using an h method.

It is also observed in each of the three examples that at least one of the ph methods was always more efficient than any of the h methods. Generally, a ph method should be able to outperform an h method since there is another parameter p than can be adjusted. When the solution is smooth, a p method always performs better than an h method since the p method converges exponentially fast. If the solution is piecewise smooth, then the ph method can exploit the smoothness by using a larger p and fewer mesh intervals in the region where the solution is smooth.

It is also important to note that the ph mesh refinement presented in this paper is based on only the estimate of the state error. While incorporating other error estimates (for example, estimates of the errors in the costate or control) is possible, in applications it is often the case that the errors in the costate and the control are comparable to the error in the state. Thus, it is simpler and more efficient to develop a mesh refinement method based only on the error in the state. Moreover, other mesh refinement methods, such as the h mesh refinement methods found in Ref. 3, are based only on an estimate of the state error.

Next, we compare the method developed in this paper with the recently developed hp-adaptive method of Refs. 28 and 29. In the methods of Ref. 28 and 29, the error is estimated from the difference between the time derivative approximation of the state and the right-hand side of the dynamics at points sampled between the Gaussian quadrature collocation points. It was found that both of these previously developed approaches were tractable only if a moderate level of accuracy was required. In the case when a high-accuracy solution is desired, the approaches of Ref. 28 and 29 are unreliable. To see the issue with the approaches of Refs. 28 and

29, consider again Example 1 of this paper. It turns out that when Example 1 is solved using the method of Ref. 29 using an $hp-(3,N_{\max})$ method and an accuracy tolerance $\epsilon \leq 10^{-6}$ that the lowest achievable error estimate is 1.2×10^{-6}. Thus, none of the $hp-(3,N_{\max})$ methods is able to meet any desired tolerance less than 10^{-6} regardless of the number of mesh refinements performed. In fact, the error estimate after the eighth mesh remained at 1.2×10^{-6} despite the fact that the method of Ref. 29 continued to attempt to improve the mesh. This result demonstrates the limited effectiveness of the method of Refs. 28 and 29 when a high-accuracy solution is desired. Contrary to the approach of Ref. 29, it is seen from Table 1 that the method of this paper provides a much more accurate estimate of the actual error and the error estimate approaches the true error as the mesh refinement proceeds.

In addition to the improved reliability of the approach developed in this paper over the previously developed hp-adaptive approaches, the approach of this paper is significantly simpler than the approaches of Refs. 28 and 29. In particular, the approach of Ref. 28 makes the decision to change the polynomial degree or to refine the mesh based on the ratio of the maximum to the mean error in a mesh interval, while the approach of Ref. 29 makes a similar decision based on the ratio of the maximum to the mean curvature in a mesh interval. In addition, the method of Ref. 29 requires that an ad hoc parameter be set that determines the number of newly created mesh intervals in the case where a mesh interval needs to be divided into subintervals. In either of these previously developed methods, the choice of these user-defined parameters is ad hoc. More importantly, the performance of the method on a particular problem changes greatly depending upon on the choice of these parameters. On the other hand, in the method of this paper only the minimum and maximum allowable polynomial degrees need to be chosen. Because for a wide range of problems the allowable polynomial degree will lie between fairly well known limits, setting the minimum and maximum allowable polynomial degrees is much more straightforward than setting the parameters required by the methods of Refs. 28 or 29.

8 Conclusions

A ph-adaptive Legendre-Gauss-Radau collocation method for solving continuous-time optimal control problems has been developed. An estimate of the error was obtained interpolating the current approximation on a finer mesh and then integrating the dynamics evaluated at the inter-

polation points to generate a more accurate approximation to the solution of the state equation. This process is analogous to the 2-stage modified Euler Runge-Kutta scheme where the first stage yields a first-order approximation to the solution of a differential equation, and the second stage interpolates this solution at a new point and then integrates the dynamics at this new point to achieve a second-order approximation to the solution. The difference between the first and second-order approximation is an estimate for the error in the first-order scheme. Using this error estimate, a mesh refinement method was developed that iteratively reduces the error estimate either by increasing the degree of the polynomial approximation in a mesh interval or by increasing the number of mesh intervals. An estimate was made of the polynomial degree required within a mesh interval to achieve a given accuracy tolerance. If the required polynomial degree was estimated to be less than an allowable maximum allowable polynomial degree, then the degree of the polynomial approximation was increased on the ensuing mesh. Otherwise, the mesh interval was divided into subintervals and the minimum allowable polynomial degree was used in each newly created subinterval on the ensuing mesh. This process was repeated until a specified relative error accuracy tolerance was met. The method was applied successfully to three examples that highlight various features of the method and show the merits of the approach relative to a fixed-order method.

Acknowledgments

The authors gratefully acknowledge support for this research from the U.S. Office of Naval Research under Grant N00014-11-1-0068 and from the U.S. Defense Advanced Research Projects Agency under Contract HR0011-12-C-0011.

Disclaimer

The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

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

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

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

[4] Jain, D. and Tsiotras, P., "Trajectory Optimization Using Multiresolution Techniques," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 5, September-October 2008, pp. 1424–1436.

[5] Zhao, Y. and Tsiotras, P., "Density Functions for Mesh Refinement in Numerical Optimal Control," *Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 1, January–February 2011, pp. 271–277.

[6] Elnagar, G., Kazemi, M., and Razzaghi, M., "The Pseudospectral Legendre Method for Discretizing Optimal Control Problems," *IEEE Transactions on Automatic Control*, Vol. 40, No. 10, 1995, pp. 1793–1796.

[7] Elnagar, G. and Razzaghi, M., "A Collocation-Type Method for Linear Quadratic Optimal Control Problems," *Optimal Control Applications and Methods*, Vol. 18, No. 3, 1998, pp. 227–235.

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

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

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

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

[12] Gong, Q., Ross, I. M., Kang, W., and Fahroo, F., "Connections Between the Covector Mapping Theorem and Convergence of Pseudospectral Methods," *Computational Optimization and Applications*, Vol. 41, No. 3, December 2008, pp. 307–335.

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

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

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

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

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

[18] Darby, C. L., Garg, D., and Rao, A. V., "Costate Estimation Using Multiple-Interval Pseudospectral Methods," *Journal of Spacecraft and Rockets*, Vol. 48, No. 5, September-October 2011, pp. 856–866, AIAA Guidance, Navigation, and Control Conference, Portland, OR, AUG 07-13, 2011.

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

[20] Canuto, C., Hussaini, M. Y., Quarteroni, A., and Zang, T. A., *Spectral Methods in Fluid Dynamics*, Spinger-Verlag, Heidelberg, Germany, 1988.

[21] Fornberg, B., *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, 1998.

[22] Trefethen, L. N., *Spectral Methods Using MATLAB*, SIAM Press, Philadelphia, 2000.

[23] Babuska, I. and Suri, M., "The p and hp Version of the Finite Element Method, an Overview," *Computer Methods in Applied Mechanics and Engineering*, Vol. 80, 1990, pp. 5–26.

[24] Babuska, I. and Suri, M., "The p and hp Version of the Finite Element Method, Basic Principles and Properties," *SIAM Review*, Vol. 36, 1994, pp. 578–632.

[25] Gui, W. and Babuska, I., "The h, p, and hp Versions of the Finite Element Method in 1 Dimension. Part I. The Error Analysis of the p Version," *Numerische Mathematik*, Vol. 49, 1986, pp. 577–612.

[26] Gui, W. and Babuska, I., "The h, p, and hp Versions of the Finite Element Method in 1 Dimension. Part II. The Error Analysis of the h and $h-p$ Versions," *Numerische Mathematik*, Vol. 49, 1986, pp. 613–657.

[27] Gui, W. and Babuska, I., "The h, p, and hp Versions of the Finite Element Method in 1 Dimension. Part III. The Adaptive $h-p$ Version," *Numerische Mathematik*, Vol. 49, 1986, pp. 659–683.

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

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

[30] Gong, Q., Fahroo, F., and Ross, I. M., "Spectral Algorithm for Pseudospectral Methods in Optimal Control," *Journal of Guidance, Control and Dynamics*, Vol. 31, No. 3, May–June 2008, pp. 460–471.

[31] Rannacher, R., "Adaptive Finite Element Discretization of Flow Problems for Goal-Oriented Model Reduction," *Computational Fluid Dynamics 2008*, edited by H. Choi, H. Choi, and J. Yoo, Springer Berlin Heidelberg, 2009, pp. 31–45.

[32] Besier, M. and Rannacher, R., "Goal-Oriented Space-Time Adaptivity in the Finite Element Galerkin Method for the Computation of Nonstationary Incompressible Flow," *International Journal for Numerical Methods in Fluids*, Vol. 70, No. 9, November 2012, pp. 1139–1166.

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

[34] Hou, H., Hager, W. W., and Rao, A. V., "Convergence of a Gauss Pseudospectral Transcription for Optimal Control," *2012 AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2012-4452, Minneapolis, MN, August 13-16 2012.

[35] Hou, H., *Convergence Analysis of Orthogonal Collocation Methods for Unconstrained Optimal Control*, Ph.D. thesis, University of Florida, Gainesville, FL, May 2013.

[36] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, Cambridge, UK, 3rd ed., 2007.

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

[38] Ledzewicz, U. and Schättler, H., "Analysis of Optimal Controls for a Mathematical Model of Tumour Anti-Angiogenesis," *Optimal Control Applications and Methods*, Vol. 29, No. 1, January–February 2008, pp. 41–57.

[39] Patterson, M. A. and Rao, A. V., "$\mathbb{GPOPS}-\mathbb{II}$: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp–Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming," *ACM Transactions on Mathematical Software*, Accepted Pending Minor Revision, September 2013.

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

[41] *MUltifrontal Massively Parallel Solver (MUMPS 4.10.0) User's Guide*, May 2011.