A Preliminary Analysis of Mesh Refinement for Optimal Control Using Discontinuity Detection via Jump Function Approximations

Alexander T. Miller*
William W. Hager†
Anil V. Rao‡

University of Florida
Gainesville, FL 32611

A mesh refinement method for optimal control which relies on a jump function approximation to detect discontinuities in the solution is described. A Fourier analysis of the solution is used to generate an approximation of a jump function. The peaks of the jump function approximation provide an estimate of possible discontinuity locations. Thresholds on the relative magnitude and decay rate of the jump function approximation are used to verify the existence of discontinuities in the solution, and the mesh is refined by bracketing the locations of the discontinuities. The method is demonstrated using three examples. Results show that solving these types of trajectory optimization problems with the new method require fewer mesh refinement iterations and less computation time when compared with a previously developed method.

I. Introduction

Numerical methods for solving optimal control problems can be categorized into indirect methods and direct methods. In an indirect method, the first-order optimality conditions are formulated using the calculus of variations which leads to a Hamiltonian boundary-value problem (HBVP),[1] and the HBVP is solved numerically. In a direct method, the control and/or the state are parameterized and the optimal control problem is transcribed to a finite-dimensional nonlinear programming problem (NLP). The NLP is then solved numerically using well known sofware.[2,3]

Over the past few decades, the particular class of direct collocation methods have been used extensively to solve optimal control problems numerically. Direct collocation methods are implicit simulation state and control parameterization methods where the constraints in the continuous optimal control problem are enforced at a specially chosen set of values of the independent variable called collocation points. The approximation of an optimal control problem using collocation gives rise to a large sparse NLP,[4] and the NLP is solved using well known software.[2,3]

Traditional direct collocation methods take the form of an h method (for example, Euler or Runge-Kutta methods) where the domain of interest is divided into a mesh and the state is approximated using the same fixed-degree polynomial in each mesh interval. Convergence of an h method is then achieved by increasing the number of mesh points.[4] In contrast to an h method, p methods have been developed in recent years. In a p method the number of intervals is fixed and convergence is achieved by increasing the degree of the polynomial approximation on each interval. In order to achieve maximum effectiveness, p methods have been developed using collocation at *Gaussian quadrature* points.[5–7] For problems whose solutions are

*Ph.D. Student, Department of Mechanical and Aerospace Engineering. E-mail: alexandertmiller@ufl.edu.
†Professor, Department of Mathematics. E-mail: hager@ufl.edu.
‡Associate Professor, Erich Farber Faculty Fellow, and University Term Professor, Department of Mechanical and Aerospace Engineering. E-mail: anilvrao@ufl.edu. Associate Fellow AIAA. Corresponding Author.

smooth and well-behaved, Gaussian quadrature collocation converges at an exponential rate.[8–12] Gauss quadrature collocation methods use Legendre-Gauss[6] (LG), Legendre-Gauss-Radau[7] (LGR), or Legendre-Gauss-Lobatto[5] (LGL) points.

Various h or p direct collocation methods have been developed previously. Reference 13 describes a p method that uses a differentiation matrix to identify potential discontinuities in the solution. Reference 14 develops an h method that uses a density function to generate a sequence of non-decreasing size meshes on which to solve the optimal control problem. Reference 4 develops an error estimate for the state using a low-order h method based on the difference between the integration of the dynamics and the integration of the time derivative of the state.

Although h methods have been used extensively and p methods are useful on certain types of problems, both the h and p approaches have limitations. In the case of an h method, it may be required to use an extremely fine mesh to improve accuracy. However, in a p method it may be required to use an unreasonably large degree polynomial to improve accuracy. In order to significantly reduce the size of the finite-dimensional approximation, and thus improve computational efficiency of solving the NLP, a new class of hp collocation methods has been developed in recent years. 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.

While hp adaptive methods can be developed using classical discretizations (for example, using a Runge-Kutta method where the order of the method in a mesh interval can be varied), employing Gaussian quadrature has advantages over classical approaches. First, exponential convergence can be achieved by increasing the degree of the polynomial approximation in segments where the solution is smooth. Second, Runge phenomenon (where the error at the ends of a mesh interval becomes very large as the polynomial degree increases) is eliminated using Gaussian quadrature. Third, less mesh refinement is necessary when using Gaussian quadrature as compared to a classical method, because the mesh only needs to be refined in segments where smoothness is lost. While hp methods were originally developed as finite-element methods for solving partial differential equations,[15] in the past few years hp methods have been extended to optimal control, and a convergence theory for these methods has been established.[8–12]

Even though hp methods can be effective in problems where the solution is smooth, it is often the case that optimal control problem solutions contain nonsmooth elements. For instance, a jump discontinuity in one or more components of the control may result in one or more state components becoming discontinuous in its derivative. Discontinuous solutions are problematic for h, p, and hp methods, because these methods assume a smooth parameterization of the state and/or control on each mesh interval.

Using a smooth or piecewise smooth parameterization when in fact the solution contains a discontinuity results in an error which must be mitigated via mesh refinement. P methods are generally ineffective in this case, because higher order polynomials remain smooth functions and mesh interval locations are fixed. Employing h or hp methods can be effective, because they allow the number of mesh intervals to vary. However, many of these methods face a major limitation. Namely, they lack the ability to detect and accurately locate discontinuities in the solution which causes mesh intervals to conglomerate around discontinuity locations as mesh refinement continues. Eventually, the discontinuity is contained on a small enough mesh interval that its effect on solution accuracy is negligible. The process of containing a discontinuity in such a way is costly in two ways. First, a large number of mesh refinement iterations may be needed and each new mesh generates an NLP which must be solved. Second, the subsequent NLP may be bigger than necessary due to mesh intervals amassing around the discontinuity location. Significant computation time penalties may result, because most of the computation time arises from solving the NLP on each mesh.

Motivated by current optimal control mesh refinement algorithms' limited ability to accurately locate discontinuities and mitigate their negative effects, the method of this paper describes a new approach where jump discontinuities in the control are detected via an approximation of the control's jump function. The origin of the jump function approximation is a result due to Lukács[16,17] which asserts that a scaled version of the Fourier conjugate sum converges to the jump function as the number of terms in the series approaches infinity. Reference 18 accelerates the convergence rate of this jump function approximation with the introduction of so-called ”concentration factors”. Once located using the control's jump function approximation, each discontinuity is confined to a much smaller interval on the new mesh. The mesh points confining a particular discontinuity are reused and their locations tightened around the discontinuity location on subsequent mesh refinement iterations. The approach used here confines discontinuities to small mesh intervals rapidly, and does so in a way which does not add unnecessary size to the NLP.

The method described in this paper is applicable to problems where the control solution can be adequately described by a Fourier series. Therefore, it is suitable for a wide range of trajectory optimization problems containing discontinuous solutions for components of the control. Furthermore, the methods discussed herein can be applied to other h and hp mesh refinement methods to help alleviate the obstacles presented by jump discontinuities in the solution to an optimal control problem.

The remainder of this paper is organized as follows. Section II describes the Bolza form of the optimal control problem. Section III provides the basis for the method developed in this paper based on jump function approximations. Section IV reviews the Legendre-Gauss-Radau (LGR) collocation method. Section V describes the new mesh refinement method with discontinuity detection for jumps in the control, and Section VI provides an overview of the mesh refinement algorithm based on the methods described in Section V. Section VII uses three examples from the open literature to compare and discuss the performance of the hp-adaptive method of Ref. 19 with the algorithm of Section VI. Finally, Section VIII provides the conclusions of this research.

II. Bolza Optimal Control Problem

The following description of a general optimal control problem written in Bolza form follows closely with the Bolza form description of Ref. 19. Without loss of generality, consider the following general optimal control problem in 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

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

subject to the dynamic constraints

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

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

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

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 $-1 = T_0 < T_1 < \ldots < T_K = +1$. 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 control in $\mathcal{S}_k$. The Bolza optimal control problem of Eqs. (1)–(4) can then rewritten as follows. Minimize the cost functional

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

subject to the dynamic constraints

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

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

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

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

III. Motivation for New Mesh Refinement Method

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

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

A. Jump Functions

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

$$f_j(t) = \lim_{\tau \to t^+} f(\tau) - \lim_{\tau \to t^-} f(\tau) \ \ \forall t \in (t_0, t_f) \tag{10}$$

From Eq. (10) it is seen that a jump function, $f_j(t)$, of an underlying function, $f(t)$, is zero everywhere except at locations where the original function, $f(t)$, has jump discontinuities. Moreover, at the jump locations of $f(t)$, $f_j(t)$ takes on the value that equals the amount of the jump discontinuity itself. Using the definition of a jump function, the jump discontinuities can be located by observing where $f_j(t)$ is nonzero.

B. Approximation of Jump Functions

Now while in principle a jump function can be obtained using Eq. (10), in practice the underlying function is not known because the solution is known only on a time series of data where the time points are the collocation points on the mesh for which the approximation of the solution to the optimal control problem was obtained. Thus, Eq. (10) must be approximated using this time series of data. A possible way to approximate a jump function is by using a Fourier series approximation of $f(t)$. If the Fourier series approximation of a $2L$ periodic function, denoted $\hat{f}(t)$, is written as

$$f(t) \approx \hat{f}(t) = a_0 + \sum_{k=1}^{N} \left[a_k \cos\left(\frac{k\pi}{L} t\right) + b_k \sin\left(\frac{k\pi}{L} t\right) \right], \tag{11}$$

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

$$f_j(t) \approx \hat{f}_j(t) = \sum_{k=1}^{N} \sigma(k/N) \left[a_k \sin\left(\frac{k\pi}{L} t\right) - b_k \cos\left(\frac{k\pi}{L} t\right) \right], \tag{12}$$

where a_k and b_k are the same Fourier coefficients of Eq. (11) and $\sigma(k/N)$ are concentration factors.[18] Concentration factors arise from the earlier work of Lukács[16,17] where it was shown that the conjugate Fourier series

$$\tilde{\hat{f}}(t) = \sum_{k=1}^{N} \left[a_k \sin\left(\frac{k\pi}{L} t\right) - b_k \cos\left(\frac{k\pi}{L} t\right) \right] \tag{13}$$

converges to the jump function when multiplied by $-\pi/\log N$ as $N \longrightarrow \infty$. A wider class of so-called "concentration factors" share the same convergence property when applied to the Fourier conjugate sum as written in Eq. (12).[18] Furthermore, these concentration factors accelerate convergence to the actual jump function, thereby making the jump function approximation computationally tractable because fewer terms in the series are required in order to obtain an accurate approximation of the jump function. In this research, the following concentration factor is employed:

$$\sigma_{k,N}^{G} = -\frac{\pi}{\text{Si}(\pi)} \sin\left(\frac{\pi k}{N}\right), \quad \text{Si}(\pi) = \int_0^{\pi} \frac{\sin t}{t} dt \approx 1.85194, \tag{14}$$

where Eq. (14) is called the Gibbs concentration factor.[18]

While obtaining the Fourier coefficients needed in Eq. (12) can be done in many ways, it is important that an even Fourier approximation be used as opposed to a standard or odd Fourier approximation. The reasoning for needing an even Fourier approximation is due to the nature of the periodic behavior the standard, odd, and even Fourier approximations imply. Specifically, consider the numerical approximation of the solution to an optimal control problem on a given mesh where the solution data lies on the time interval $t \in [t_0, t_f]$. A standard Fourier approximation of the form in Eq. (11) will result in the Fourier approximation of the control having a period $T = 2L = t_f - t_0$. Due to this periodicity, artificial jumps may be present at the endpoints as illustrated in Fig. 2. The same issue arises when using the odd Fourier approximation, $\hat{f}_{\text{odd}}(t)$, which is defined as

$$\hat{f}_{\text{odd}}(t) = \sum_{k=1}^{N} b_k \sin\left(\frac{k\pi}{L} t\right), \tag{15}$$

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

$$\hat{f}_{\text{even}}(t) = a_0 + \sum_{k=1}^{N} a_k \cos\left(\frac{k\pi}{L} t\right), \tag{16}$$

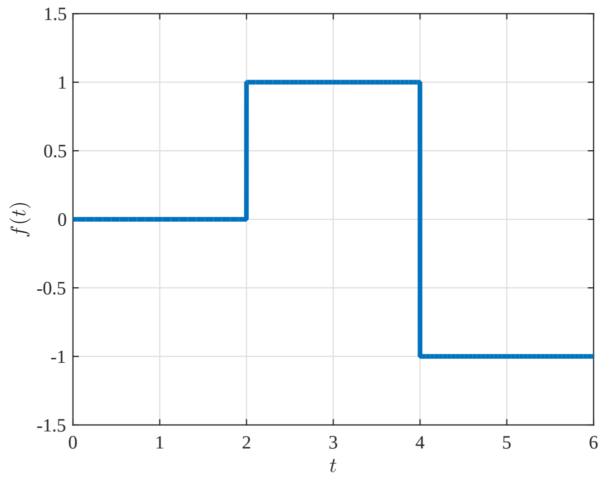
has the same period as the odd approximation but does not pose the risk of creating artificial jumps at t_0 or t_f. An even Fourier series approximation does not produce artificial jumps at the endpoints because the value of the function approximation at the start of any period is equal to the value of the function approximation at the end of the previous period. In order to see the behavior of a standard, odd, and even Fourier series approximation of a jump function, consider the following function:

$$f(t) = \begin{cases} 0 & , \quad 0 \le t < 2, \\ 1 & , \quad 2 \le t < 4, \\ -1 & , \quad 4 \le t \le 6, \end{cases} \tag{17}$$

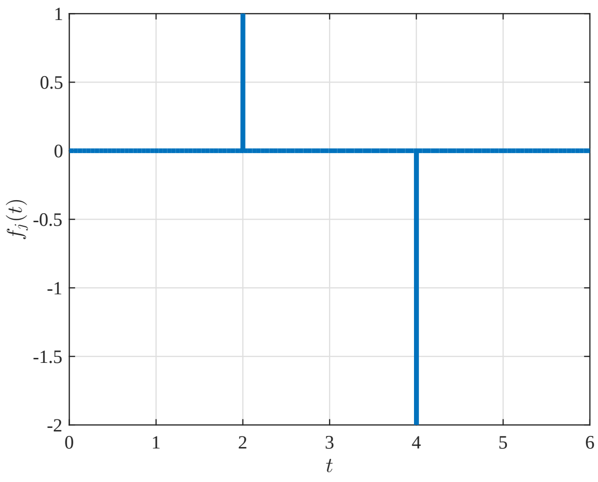
The jump function $f_j(t)$ arising from the function $f(t)$ defined in Eq. (17) is given as

$$f_j(t) = \begin{cases} 1 & , \quad t = 2, \\ -2 & , \quad t = 4, \\ 0 & , \quad \text{otherwise}. \end{cases} \tag{18}$$

Figures 1a and 1b show, respectively, the functions $f(t)$ and $f_j(t)$ defined in Eqs. (17) and (18). Next, Figure 2a shows the standard, odd, and even Fourier series approximations of the function $f(t)$ defined in Eq. (17), while Fig. 2b shows the standard, odd, and even Fourier series approximations of the jump function $f_j(t)$ defined in Eq. (18). It can be seen from Fig. 2a that all three Fourier approximations provide a good approximation of $f(t)$. On the other hand, Fig. 2b shows that the standard and odd Fourier approximations of the jump function $f_j(t)$ defined in Eq. (18) produce artificial jumps at $t = t_0$ and $t = t_f$, while artificial jumps are not produced by the even Fourier approximation of $f_j(t)$. Because the even Fourier series approximation of a jump function does not produce artificial jumps, even Fourier approximations will be employed in the remainder of this paper to obtain the Fourier coefficients needed in the approximation of a jump function. Finally, for convenience, from this point forth the notation $\hat{f}(t)$ will be used to denote an even Fourier series approximation.

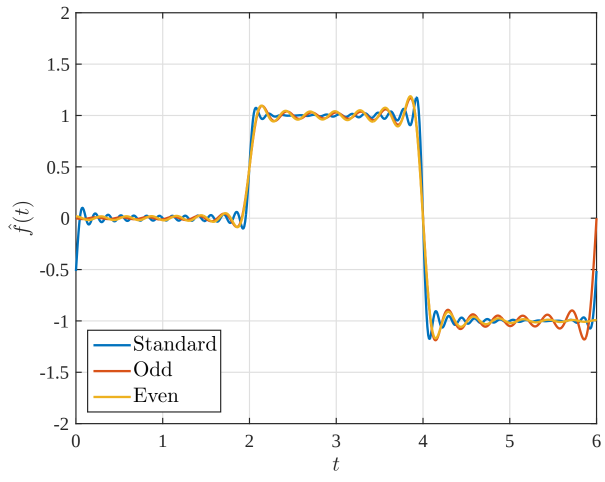


(a) Function, $f(t)$, defined in Eq. (17).

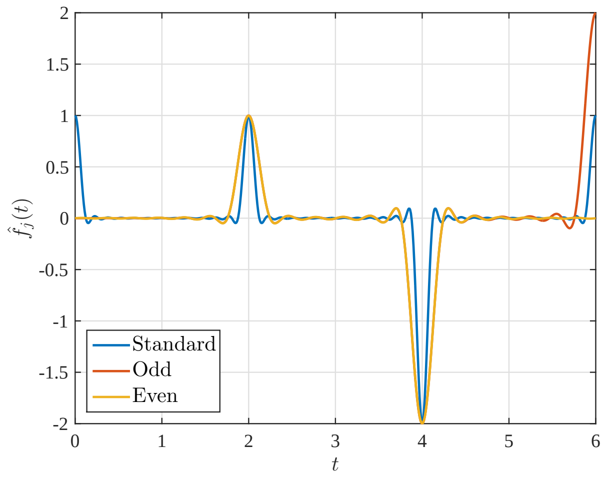


(b) Jump function, $f_j(t)$, defined in Eq. (18).

Figure 1: Example function, $f(t)$, defined in Eq. (17) alongside jump function, $f_j(t)$, defined in Eq. 18.



(a) Fourier approximation of $f(t)$ defined in Eq. (17).



(b) Jump function approximation of $f_j(t)$ defined in Eq. (18).

Figure 2: Standard, odd, and even Fourier approximations of $f(t)$ and their associated jump function approximations. Note that $\hat{f}_j(t)$ has no artificial jumps at the endpoints for the even approximation.

Given that an even Fourier series approximation does not create artificial jumps when approximating a jump function, suppose now the jump function $f_j(t)$ given Eq. (18) is approximated using an even N-term Fourier series, and let this jump function approximation be denoted $\hat{f}_j(t)$. Figure 1 shows the jump function approximation $\hat{f}_j(t)$ for $N = \{10, 20, 40\}$, where it is seen that the jump function approximation approaches the true jump function as N increases. Furthermore, the locations of the maxima and minima of the jump function approximation lie in close proximity, respectively, to the locations of the discontinuities of the actual function $f(t)$, and the values of the jump function approximation at these extremal points are in close proximity to the actual jump in the original function. It is also seen that the extrema in the jump function approximation tend to stay in the same location when a jump discontinuity is present regardless of the value of N. Therefore, locating the maxima and minima in a jump function approximation can be

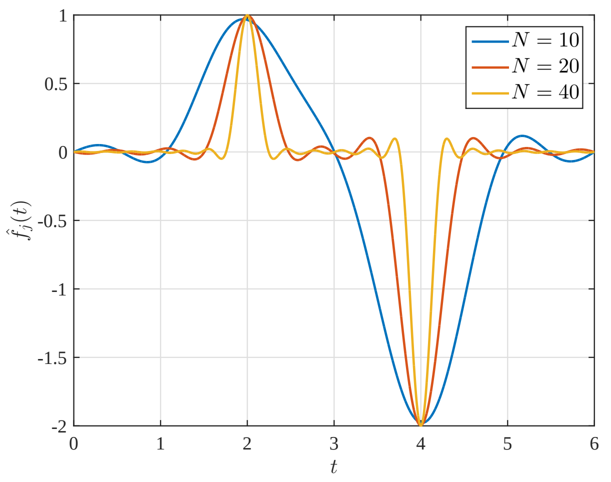


Figure 3: N-term approximation of jump function $f_j(t)$ given in Eq. (18) using $N = \{10, 20, 40\}$. Note that the coefficients used in the jump function approximation correspond to an even Fourier approximation of funciton $f(t)$ defined in Eq. (17).

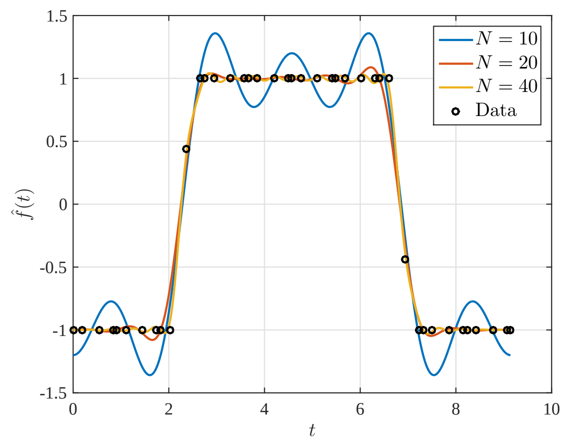
used as a good estimate of the location of jump discontinuities of a function $f(t)$.

C. Even Fourier Series Approximation of Jump Functions Using Unevenly Spaced Data

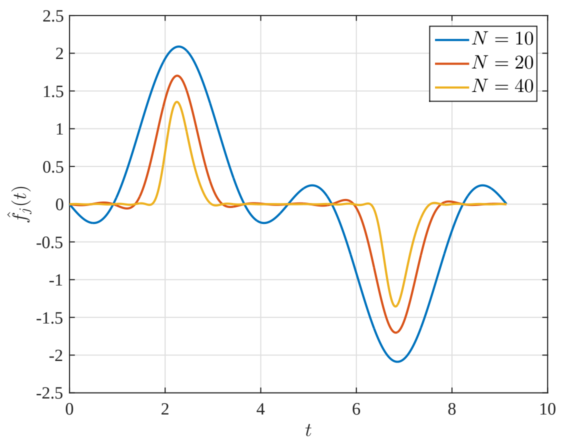
In the example that was studied in Section III.B, the underlying function, $f(t)$, was known. As a result, it was possible to obtain the Fourier coefficients of the jump function, $f_j(t)$, of $f(t)$, analytically. Note, however, that the solution obtained by solving the NLP that arises from the transcription of an optimal control problem via collocation leads to an approximation of the state and control at discrete (sampled) data points. Moreover, this discrete approximation is obtained at points that are not evenly spaced. Thus, any jump function approximation that would be used to determine the locations of discontinuities in the solution must be obtained using this discrete data. In this paper, an even Fourier approximation of unevenly spaced data that lies on $t \in [t_0, t_f]$ is obtained as follows. First, the unevenly spaced data is interpolated to $N + 1$ evenly spaced points on the time interval $t \in [t_0, t_f]$. Second, these interpolated data points are reflected about t_f (excluding the points at t_0 and t_f) to create a sampling of an even function with period $2(t_f - t_0)$. The Fast Fourier Transform (FFT) is then utilized to calculate the first N Fourier coefficients (excluding a_0) of the even Fourier approximation of the data on the period $2(t_f - t_0)$. The choice of N is somewhat arbitrary so long as the resulting Fourier approximation reasonably describes the original set of unevenly spaced data. Figs. 4a and 4b show, respectively, the even Fourier series approximations of unevenly spaced data alongside the corresponding jump function approximations for $N = \{10, 20, 40\}$.

Examining Fig. 4, it is seen that the jump function approximation obtained using unevenly spaced data has similar features to the jump function approximation obtained in Fig. 4 where the function $f(t)$ is known. Specifically, the global extrema of the jump function approximation shown in Fig. 4 obtained using unevenly spaced data correspond closely with the locations of the discontinuities in the sampled function as is the case in Fig. 4 where the function $f(t)$ is known. Moreover, these extrema locations do not tend to vary as N is increased. However, it is observed that the values of these extrema tend to decrease as N is increased. The decrease is due to the linear interpolation step when calculating the even Fourier series coefficients using the approach described previously. Despite this new drawback, these extreme points of each jump function approximation remain good estimates for the locations of jumps in the underlying function of the unevenly sampled data. The extrema of a jump function approximation $\hat{f}_j(t)$ is obtained by determining the the zeros of the derivative of $\hat{f}_j(t)$, where the derivative of $\hat{f}_j(t)$ is given as

$$\frac{d\hat{f}_j(t)}{dt} = \sum_{k=1}^{N} \sigma\left(\frac{k}{N}\right) \frac{k\pi}{L} \left[a_k \cos\left(\frac{k\pi}{L} t\right) + b_k \sin\left(\frac{k\pi}{L} t\right)\right]. \tag{19}$$



(a) Even Fourier approximations of unevenly spaced data.



(b) Jump function approximations of the unevenly spaced data in Fig. 4a.

Figure 4: N-term Fourier and jump function approximations of unevenly spaced data for $N = \{10, 20, 40\}$.

IV. Legendre-Gauss-Radau Collocation

Although the ideas of Section III can be applied to many different collocation methods for solving an optimal control problem of the form described in Section II; here we choose Legendre-Gauss-Radau (LGR) collocation as the vehicle for explaining and demonstrating the behavior of the mesh refinement method of this paper. The following description of the LGR collocation method follows closely with the LGR method described in Ref. 19.

The multiple-interval form of the continuous-time Bolza optimal control problem in Section II is discretized using collocation at LGR points.[7,20–23] 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

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

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 (LGR)[24] 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 gives

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

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

$$\sum_{j=1}^{N_k+1} D_{ij}^{(k)} \mathbf{Y}_j^{(k)} = \frac{t_f - t_0}{2} \mathbf{a}(\mathbf{Y}_i^{(k)}, \mathbf{U}_i^{(k)}, t(\tau_i^{(k)}, t_0, t_f)), \quad (i = 1,\ldots,N_k), \tag{22}$$

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*[7] 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

$$\mathcal{J} \approx \mathcal{M}(\mathbf{Y}_1^{(1)}, t_0, \mathbf{Y}_{N_K+1}^{(K)}, t_f) + \sum_{k=1}^{K} \sum_{j=1}^{N_k} \frac{t_f - t_0}{2} w_j^{(k)} \mathcal{L}(\mathbf{Y}_j^{(k)}, \mathbf{U}_j^{(k)}, t(\tau_j^{(k)}, t_0, t_f)) \tag{23}$$

subject to the collocation constraints

$$\sum_{j=1}^{N_k+1} D_{ij}^{(k)} \mathbf{Y}_j^{(k)} - \frac{t_f - t_0}{2} \mathbf{a}(\mathbf{Y}_i^{(k)}, \mathbf{U}_i^{(k)}, t(\tau_i^{(k)}, t_0, t_f)) = \mathbf{0}, \quad (i = 1,\ldots,N_k), \tag{24}$$

the discretized path constraints

$$\mathbf{c}_{\min} \leq \mathbf{c}(\mathbf{Y}_i^{(k)}, \mathbf{U}_i^{(k)}, t(\tau_i^{(k)}, t_0, t_f)) \leq \mathbf{c}_{\max}, \quad (i = 1,\ldots,N_k), \tag{25}$$

and the discretized boundary conditions

$$\mathbf{b}_{\min} \leq \mathbf{b}(\mathbf{Y}_1^{(1)}, t_0, \mathbf{Y}_{N_K+1}^{(K)}, t_f) \leq \mathbf{b}_{\max}. \tag{26}$$

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

$$\mathbf{Y}_{N_k+1}^{(k)} = \mathbf{Y}_1^{(k+1)}, \quad (k = 1,\ldots,K-1). \tag{27}$$

Computationally, the constraint of Eq. (27) is eliminated from the problem by using the *same* variable for both $\mathbf{Y}_{N_k+1}^{(k)}$ and $\mathbf{Y}_1^{(k+1)}$. Finally, we note that

$$N = \sum_{k=1}^{K} N_k \tag{28}$$

is the total number of LGR points on $\tau \in [-1, +1]$.

A. 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. (23)–(26) 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 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

$$\mathbf{U}^{(k)}(\tau) = \sum_{j=1}^{N_k} \mathbf{U}_j^{(k)} \hat{\ell}_j^{(k)}(\tau), \quad \hat{\ell}_j^{(k)}(\tau) = \prod_{\substack{l=1 \\ l \neq j}}^{N_k} \frac{\tau - \tau_l^{(k)}}{\tau_j^{(k)} - \tau_l^{(k)}}, \tag{29}$$

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

$$\hat{\mathbf{Y}}^{(k)}(\hat{\tau}_j^{(k)}) = \mathbf{Y}^{(k)}(\tau_{k-1}) + \frac{t_f - t_0}{2} \sum_{l=1}^{M_k} \hat{I}_{jl}^{(k)} \mathbf{a}\left(\mathbf{Y}^{(k)}(\hat{\tau}_l^{(k)}), \mathbf{U}^{(k)}(\hat{\tau}_l^{(k)}), t(\hat{\tau}_l^{(k)}, t_0, t_f)\right), \qquad j = 2, \ldots, M_k + 1, \tag{30}$$

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

$$\begin{array}{rcl} E_i^{(k)}(\hat{\tau}_l^{(k)}) & = & \left| \hat{Y}_i^{(k)}(\hat{\tau}_l^{(k)}) - Y_i^{(k)}(\hat{\tau}_l^{(k)}) \right|, \\ e_i^{(k)}(\hat{\tau}_l^{(k)}) & = & \dfrac{E_i^{(k)}(\hat{\tau}_l^{(k)})}{1 + \max\limits_{\substack{j \in [1,\ldots,N_k+1] \\ k \in [1,\ldots,K]}} \left| Y_i^{(k)}(\tau_j^{(k)}) \right|}, \end{array} \quad \left[\begin{array}{c} l = 1, \ldots, M_k + 1, \\ i = 1, \ldots, n_y, \end{array} \right]. \tag{31}$$

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

$$e_{\max}^{(k)} = \max_{\substack{i \in [1,\ldots,n_y] \\ l \in [1,\ldots,M_k+1]}} e_i^{(k)}(\hat{\tau}_l^{(k)}). \tag{32}$$

V. Mesh Refinement Method with Discontinuity Detection

The mesh refinement method used in this paper combines the hp-adaptive scheme of Ref. 19 with the approach described in Section III for approximating jump functions. This discussion is restricted to detecting discontinuities and does not provide a discussion of the mesh refinement method of Ref. 19.

Suppose that the NLP of Eqs. (23)-(26) arising from the LGR collocation method is solved solved on on a mesh that has either been supplied (that is, an initial mesh) or a mesh that has been computed using a mesh refinement method (for example, the hp-adaptive method of Ref. 19). After solving the NLP on the initial mesh, the error is approximated using the error approximation method given in Section IV.A. For each mesh interval on the mesh where the maximum relative error tolerance, ϵ, is exceeded, the following discontinuity detection procedure is employed. First, for each component of the control, a jump function approximation is constructed using an even Fourier series approximation as described in Section III. The global extrema of these jump function approximations are determined and are used as the basis for determining the existence of jump discontinuity. When a discontinuity is detected, the current mesh is refined by bracketing the discontinuity with three new mesh points (one at the estimated discontinuity location and two more surrounding the estimated discontinuity location). When no discontinuity is detected, the hp mesh refinement method of Ref. 19 is employed. The result of refining each mesh interval in the aforementioned manner leads to a new mesh. The optimal control problem is then approximated on this new mesh using the LGR collocation method given in Section IV and the NLP of Eqs. (23)-(26) is solved. The process of constructing a new mesh using the hp-adaptive method of Ref. 19 together with the aforementioned discontinuity detection method is repeated until the relative error tolerance ϵ is satisfied on every mesh interval.

A. Method for Obtaining Fourier Coefficients

Assume that at least one mesh interval exists on the current mesh for which the estimated relative error of the solution is larger than the relative error tolerance, ϵ. Fourier coefficients corresponding to an even Fourier series approximation must then be calculated so that the jump function of each control component may be approximated. The necessary Fourier coefficients are generated from the set of estimated values of the control at the collocation points obtained by solving the NLP of Eqs. (23)-(26) on mesh M. The number of sdata points used to approximate the jump function can range from the data on a single mesh interval to the data on the entire mesh. It is desirable to use as many of the collocation points in the Fourier approximation as possible, because computing one set of Fourier coefficients for all K of the mesh intervals will be faster than producing K sets of Fourier coefficients with one set for each interval. Note, however, that the mesh fraction (ratio of the mesh interval time span to the total time span of the mesh) of individual mesh intervals may be widely different. Large differences between the mesh fractions cause the collocation data to be concentrated in some areas of the mesh and sparse in others, and such unevenly spaced data can result in a poor Fourier series approximation.

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

$$\frac{T_k - T_{k-1}}{\max\limits_{j \in \{K_g+1,...,K_g+k_g\}} (T_j - T_{j-1})} \geq \rho_1, \quad \forall k \in \{K_g + 1, ..., K_g + k_g\}, \tag{33}$$

where

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

and $\rho_1 \in [0, 1]$ is a user-defined threshold, k_g are the number of mesh intervals in group $\mathcal{G}_g$, G is the number of groups, $\sum\limits_{i=1}^{G} k_i = K$, and $\bigcup\limits_{g=1}^{G} \mathcal{G}_g = \{S_1, ..., S_K\}$.

Mesh intervals are grouped in the following manner. We start with one group containing all of the mesh intervals in the current mesh ($\mathcal{G}_1 = \{S_1, \ldots, S_K\}$). If one or more of its members does not satisfy Eq. (33), $\mathcal{G}_1$ is split into G new groups, $\mathcal{G}_1 = \{S_1, \ldots, S_{k_1}\}, \mathcal{G}_2 = \{S_{k_1+1}, \ldots, S_{k_1+k_2}\}, \ldots, \mathcal{G}_G = \{S_{K_G+1}, \ldots, S_K\}$. Each new group contains k_g adjacent mesh intervals which either all satisfy or all do not satisfy Eq. (33). Each of the new groups undergo the same division process until Eq. (33) is satisfied by all mesh intervals in each group.

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

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

B. Method for Locating and Verifying Discontinuity Locations

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

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

$$\Delta_r = \frac{u_i^k - u_{i-1}^k}{\left(\max\limits_{k\in\{K_g+1,\ldots,K_g+k_g\}\ ,\ i\in k}\left(u_i^k\right)\right) - \left(\min\limits_{k\in\{K_g+1,\ldots,K_g+k_g\}\ ,\ i\in k}\left(u_i^k\right)\right)}, \tag{34}$$

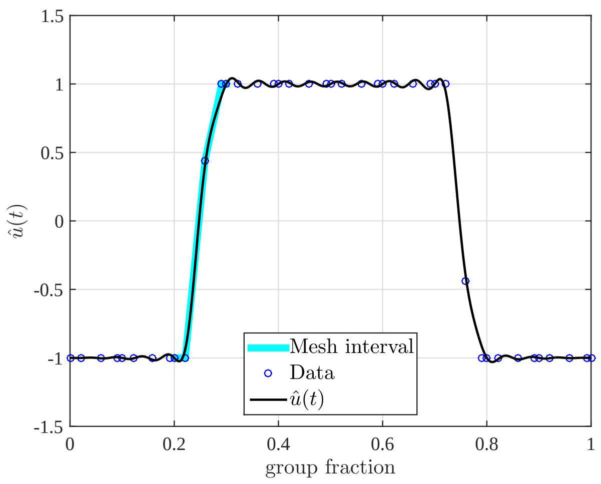
where u_i^k is inclusive of the endpoint of the interval $\mathbf{u}_{N_k+1}^k = \mathbf{u}_1^{k+1}$. Note that division by zero in Eq. (34) is possible only if u_i^k is constant for all $i \in k$, $k \in \{K_g+1, \ldots, K_g+k_g\}$. In such a case, no jump discontinuity is likely. Therefore, we stop searching for jump discontinuities in that particular control component before the division by zero can occur. In all other cases, if the absolute value of Δ_r exceeds a user-set threshold $\rho_2 \in [0, 1]$, then the search for a jump discontinuity in that particular control component is continued.

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

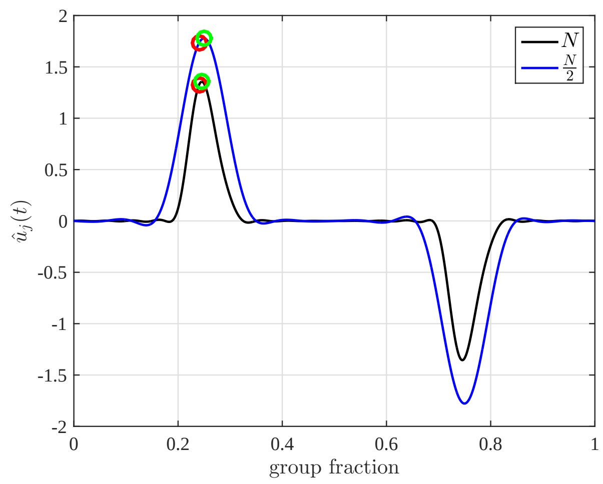
Assuming that $\Delta_r \geq \rho_2$ for a particular control component, the location of the jump discontinuity and the value of the jump must be estimated. Earlier, in Section III, it was shown that the location of the maximum or minimum of the jump function approximation, $\hat{f}_j(t)$, of Eq. (12) can be a good approximation for the location and value of the jump in $f(t)$. We now seek to obtain an extremum of the control component's jump function approximation, denoted as $\hat{u}_j(t)$, within the current mesh interval.

An extremum of $\hat{u}_j(t)$ can be found by implementing Newton's method to find a zero for its derivative $\frac{d}{dt}\hat{u}_j(t)$. We use the midpoint between the times corresponding to u_i^k and u_{i-1}^k (the same points used to calculate Δ_r in Eq. 34) as our initial guess. Once a zero of $\frac{d}{dt}\hat{u}_j(t)$ is obtained, the second derivative of the jump function approximation $\frac{d^2}{dt^2}\hat{u}_j(t)$ is used to verify that the concavity matches the jump, because we should converge to a local max if the jump is positive and a local min if the jump is negative. The process of locating an extremum of $\frac{d}{dt}\hat{u}_j(t)$ is done twice; once using all N Fourier coefficients, and once more using the first $\frac{N}{2}$ (rounded up to the nearest integer) coefficients. The values and locations of each extremum are stored for further analysis. In the case where we are unable to converge to an extremum within S_k and with the correct concavity, the search for a jump discontinuity is ceased for that particular control component.

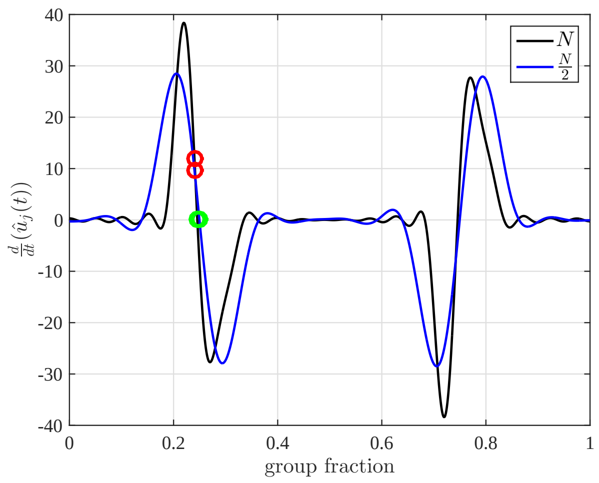
An example of locating an extremum of $\hat{u}_j(t)$ is depicted in Fig. 5 along with visualizations for the first and second derivatives of $\hat{u}_j(t)$. As can be seen, the initial guess is quite close to the extremal point of $\hat{u}_j(t)$ which results in rapid convergence.



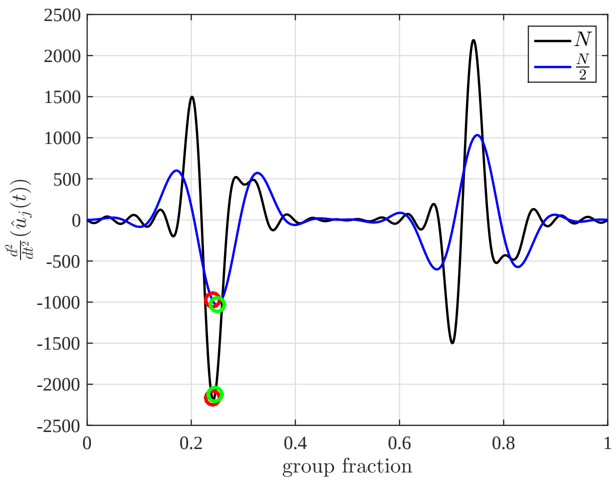
(a) Even Fourier approximation of control collocation data.



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



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



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

Figure 5: Example of the mesh algorithm being implemented to find the maximum of the jump function on the current mesh interval. The red circles correspond to the initial guess and the green circles correspond to the maximum of the jump function. Convergence to the maximum was acheived using Newton's method. Note that the x-axis is labeled "group fraction" which corresponds to the ratio of $T_k - T_{k-1}$ over $T_{k_g} - T_{k_1-1}$.

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

$$\frac{|\hat{u}_j(t^*_N)|}{\left(\max\limits_{k\in\{K_g+1,\ldots,K_g+k_g\}\ ,\ i\in k}\left(u_i^k\right)\right)-\left(\min\limits_{k\in\{K_g+1,\ldots,K_g+k_g\}\ ,\ i\in k}\left(u_i^k\right)\right)} \geq \rho_2, \tag{35}$$

where ρ_2 is the same threshold used previously on Δ_r.

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

$$\frac{|\hat{u}_{j,N}(t_N^*) - \hat{u}_{j,\frac{N}{2}}(t_{\frac{N}{2}}^*)|}{|\hat{u}_{j,N}(t_N^*)|} \leq \rho_3, \tag{36}$$

and

$$\frac{|t_N^* - t_{\frac{N}{2}}^*|}{T_{K_g+k_g} - T_{K_g}} \leq \rho_4, \tag{37}$$

where T_{K_g} and $T_{K_g+k_g}$ are the endpoints of the current group and ρ_3 and ρ_4 are user-defined thresholds. In this research, $\rho_3 = 0.5$ and $\rho_2 = 0.02$. If the criteria of Eq. (35)-(37) are all satisfied, then t_N^* is considered a reliable estimate for the location of a jump discontinuity in the current control component under investigation.

C. Mesh Refinement Actions

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

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

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

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

VI. Mesh Refinement Algorithm

A summary of our mesh refinement algorithm appears below. The hp-adaptive scheme of Ref. 19 provides the shell to which our discontinuity detection algorithm is added. The mesh number is denoted by M and is incremented by one with each loop of the algorithm. M also corresponds to the number of mesh refinement iterations, because M is initialized at 0. This algorithm terminates when either the error tolerance is satisfied in Step 5 or when M reaches a prescribed limit M_{max}.

Mesh Refinement with Discontinuity Detection

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

Step 2: Solve Radau collocation NLP of Eqs. (23)–(26) on mesh M.

Step 3: Group adjacent mesh intervals such that Eq. (33) is satisfied for each group G.

Step 4: Compute scaled error $e_{\max}^{(k)}$ in $\mathcal{S}_k,\ k = 1, \ldots, K$, using Eq. (32).

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

Step 6: For every mesh interval $\mathcal{S}_k, k \in [1, \ldots, K]$,

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

(b) if $e_{\max}^{(k)} < \epsilon$, determine if the mesh size can be reduced using the method of Ref. 19

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

VII. Results and Discussion

The following example problems from the open literature are used to test the performance of the algorithm described in Section VI when compared with the base algorithm of Ref. 19 which does not actively search for jump discontinuity locations. All solutions to the example problems contain jump discontinuities in at least one component of the control. All computation times (CPU times) shown are based on an average time obtained by solving each problem ten times. The computations were performed using a MacBook Pro with a 2.8 GHz Intel Core i7 processor and 16 GB of RAM.

A. Minimum Control Effort Landing on the Moon

Consider the following optimal control problem. Minimize the objective functional

$$\min J = \int_0^{t_f} u(t)dt \tag{38}$$

subject to the dynamic constraints and boundary conditions

$$\begin{array}{rclcl} \dot{h}(t) & = & v(t) & , & (h(0), h(t_f)) = (10, 0), \\ \dot{v}(t) & = & -g + u(t) & , & (v(0), v(t_f)) = (-2, 0), \end{array} \tag{39}$$

and the control inequality constraint

$$0 \leq u(t) \leq 3, \tag{40}$$

where $(h(t), v(t)) \in \mathbb{R}^2$ is the state, $u(t) \in \mathbb{R}$ is control, and g is a constant. The optimal control problem given in Eqs. (38)–(40) was solved using a mesh refinement relative error tolerance $\epsilon = 10^{-6}$. It is known that the solution to the optimal control problem defined in Eqs. (38)–(40) has a bang-bang optimal control with a single control discontinuity at $t \approx 1.41$ where the control switches from its minimum allowable value to its maximum allowable value. The control solution is shown in Fig. 6. The mesh histories using the hp-adaptive method of Ref. 19 without and with the discontinuity method developed in this paper are shown in Fig. 7, while the corresponding final mesh characteristics and CPU times are shown in Table 1.

When discontinuity detection is applied, the problem solves on the second mesh, needing only a single mesh refinement iteration. Without discontinuity detection, however, the interval containing the discontinuity is cut into thirds on the first mesh refinement iteration, then consecutively halved twice on the second and third iterations of mesh refinement before the mesh error tolerance, ϵ, is satisfied. Although the method of Ref. 19 does a good job of dividing only the nonsmooth mesh interval on each mesh refinement iteration, it divides in an evenly spaced manner without regard to the location of the discontinuity. As a result, as opposed to locating the discontinuity and dividing the interval using the method of Section V C, extra mesh refinement iterations are required. Moreover, as seen in Table 1, both the final mesh and the required CPU time without discontinuity detection are larger than the final mesh and the CPU time required with discontinuity detection.

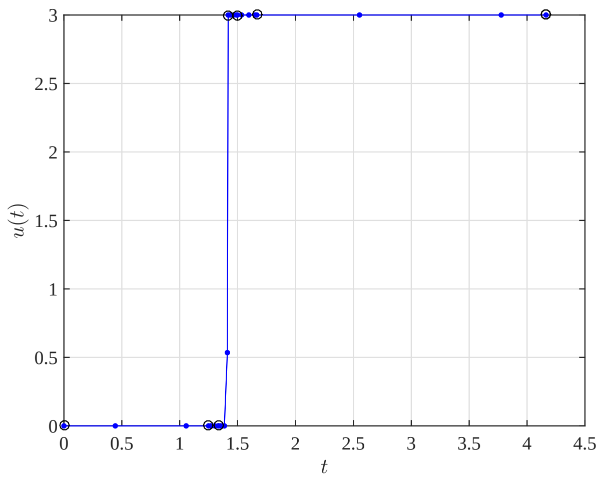
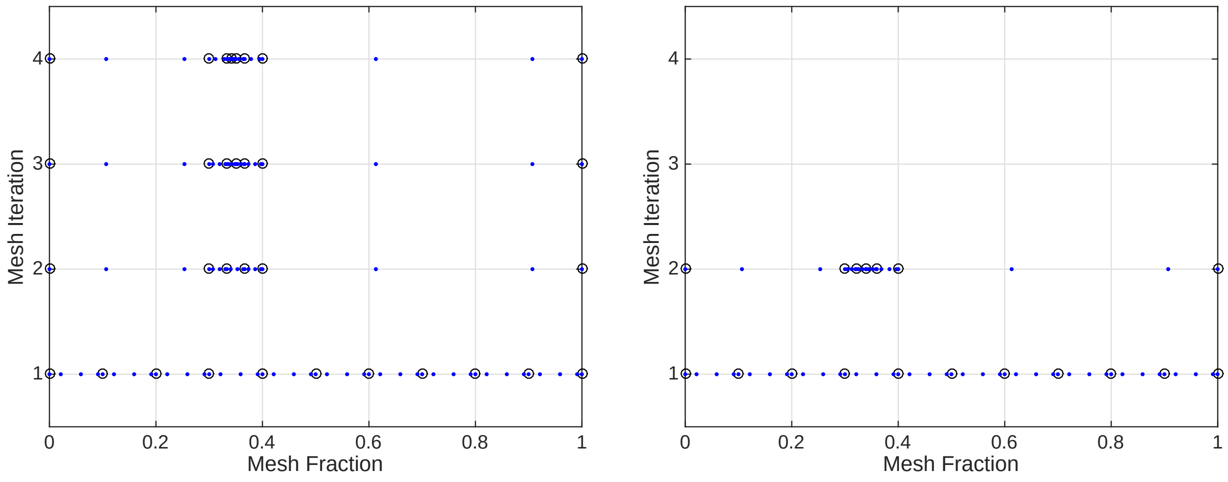


Figure 6: Control solution for Example A.



(a) hp-adaptive method of Ref. 19 without discontinuity detection.

(b) hp-adaptive method of Ref. 19 with discontinuity detection.

Figure 7: Mesh histories using hp-adaptive method of Ref. 19 without and with discontinuity detection for Example A.

Table 1: Final Mesh Characteristics for and Computation Times for Example A.

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

B. Minimum Time Reorientation of a Robot Arm

Consider the following optimal control problem. Minimize the objective functional

$$\min J = t_f \tag{41}$$

subject to the dynamic constraints

$$
\begin{array}{rcllr}
\dot{x}_1(t) & = & x_2(t) & , & (x_1(0), x_1(t_f)) = (9/2, 0), \\
\dot{x}_2(t) & = & u_1(t)/L & , & (x_2(0), x_2(t_f)) = (0, 0), \\
\dot{x}_3(t) & = & x_4(t) & , & (x_3(0), x_3(t_f)) = (0, 2\pi/3), \\
\dot{x}_4(t) & = & u_2(t)/I_\theta & , & (x_4(0), x_4(t_f)) = (0, 0), \\
\dot{x}_5(t) & = & x_6(t) & , & (x_5(0), x_5(t_f)) = (\pi/4, \pi/4), \\
\dot{x}_6(t) & = & u_3(t)/I_\phi & , & (x_6(0), x_6(t_f)) = (0, 0),
\end{array}
\tag{42}
$$

and the control inequality constraints

$$
|u_i(t)| \leq 1, \quad (i = 1, 2, 3). \tag{43}
$$

where $(x_1(t), x_2(t), x_3(t), x_4(t), x_5(t), x_6(t)) \in \mathbb{R}^6$ is the state, $(u_1(t), u_2(t), u_3(t)) \in \mathbb{R}^3$ is the control, $I_\phi = ((L - x_1(t))^3 + x_1^3(t))/3$, and $I_\theta = I_\phi \sin^2(x_5(t))$. The optimal control problem given in Eqs. (41)–(43) was solved using a mesh refinement relative error tolerance $\epsilon = 10^{-8}$. It is known that each component of the optimal control for the example given by Eqs. (41)–(43) has a bang-bang structure with a total of five control discontinuities located at $t \approx \{2.28, 2.80, 4.57, 6.35, 6.86\}$. The control solution for the example given by Eqs. (41)–(43) is shown in Fig. 8. The mesh histories using the hp-adaptive method of Ref. 19 without and with the discontinuity method developed in this paper are shown in Fig. 9, while the corresponding final mesh characteristics and CPU times are shown in Table 2. The final mesh characteristics and computation times respectively, in Tables 2.

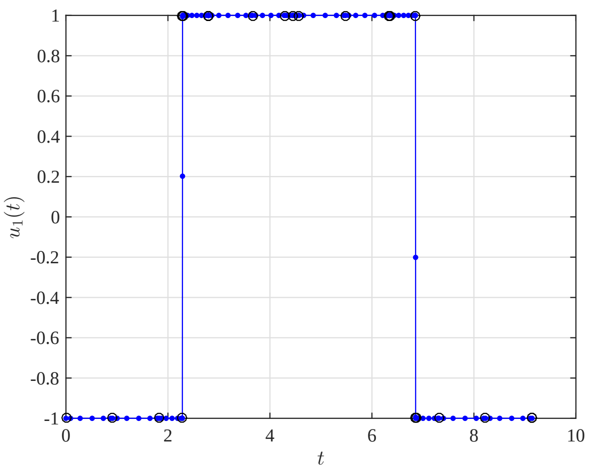
The discontinuity detection algorithm of this paper was able to successfully detect all five jump discontinuities on the first mesh refinement iteration. As can be seen in Fig. 9 (b) the mesh point triplets which bracket each discontinuity are successfully reassigned with each mesh refinement iteration. The reassignment of discontinuity bracketing mesh points has the visual effect of each mesh point triplet shrinking around the discontinuity location with each successive mesh refinement iteration. Reassigning mesh points in this manner has the desirable effect of keeping the number of mesh intervals on each new mesh smaller than it would have been had the mesh points not been reassigned. Table 2 indicate a more efficient use of mesh points and collocation points as well as a faster convergence to the solution when comparing the algorithm of this paper to the algorithm without jump discontinuity detection. In fact, the average computation time is nearly cut in half.

Table 2: Final Mesh Characteristics and Computation Times for Example B.

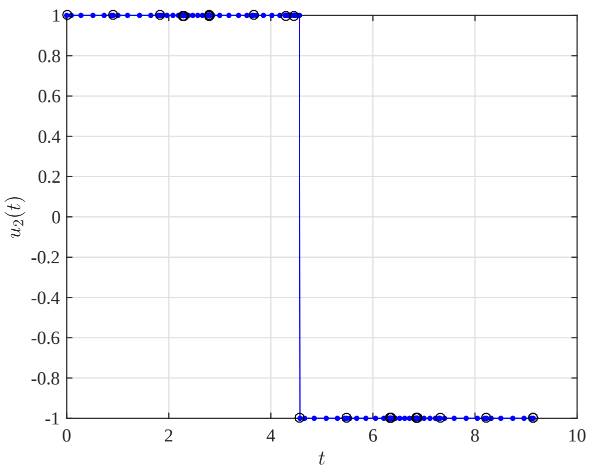
	Without Discontinuity Detection	With Discontinuity Detection
Number of Meshes	7	4
Number of Mesh Intervals	33	22
Number of Collocation Points	182	118
CPU Time	1.780	0.909

VIII. Conclusions

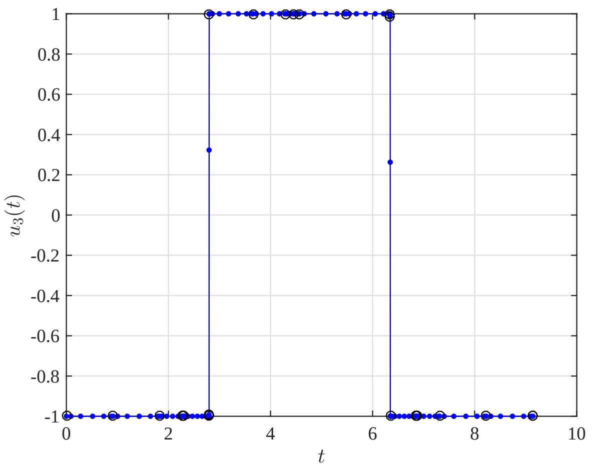
A mesh refinement method for optimal control has been developed that can detect jump discontinuities in control components. A jump function was defined and a method for approximating a jump function using an even Fourier series approximation was developed. The locations of discontinuities in the solution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



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

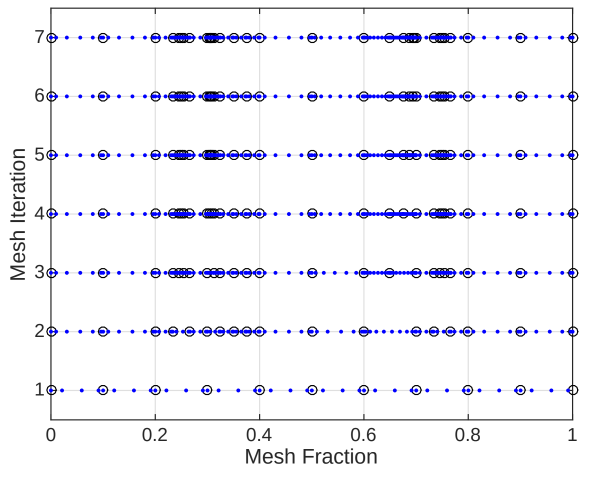


(b) Control component $u_2(t)$.

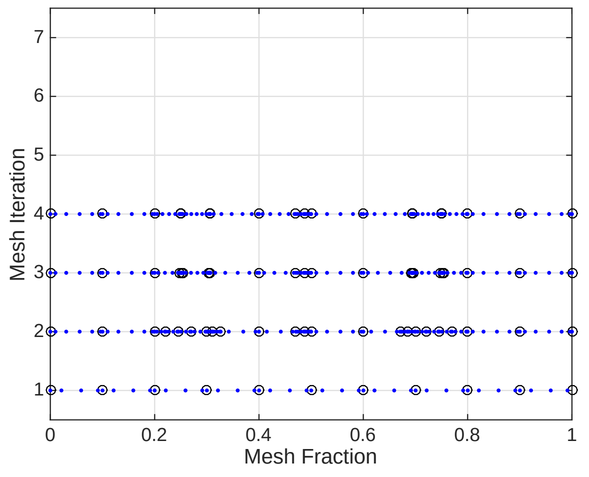


(c) Control component $u_3(t)$.

Figure 8: Control solution for Example B.



(a) Without discontinuity detection.



(b) With discontinuity detection.

Figure 9: Mesh histories using hp-adaptive method of Ref. 19 without and with discontinuity detection for Example B.

Acknowledgments

Support for this research from the U.S. Office of Naval Research under grant N00014-15-1-2048, from the U.S. National Science Foundation under grants CBET-1404767, DMS-1522629, and CMMI-1563225, and from the U.S. Air Force Research Laboratory under contract FA8651-08-D-0108/0054 is gratefully acknowledged.

References

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

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] Hager, W. W., Hou, H., and Rao, A. V., "Lebesgue Constants Arising in a Class of Collocation Methods," *IMA Journal of Numerical Analysis*, 2016, to appear.

[9] Hager, W. W., Hou, H., and Rao, A. V., "Convergence Rate for a Gauss Collocation Method Applied to Unconstrained Optimal Control," *Journal of Optimization Theory and Applications*, Vol. 169, 2016, pp. 801–824.

[10] Hager, W. W., Hou, H., and Rao, A. V., "Convergence Rate for a Radau Collocation Method Applied to Unconstrained Optimal Control," 2015, arXiv.org/abs/1508.03783.

[11] Hager, W. W., Hou, H., Mohapatra, S., and Rao, A. V., "Convergence Rate for an hp-Collocation Method Applied to Unconstrained Optimal Control," 2016, arXiv.org/abs/1605.02121.

[12] Hager, W. W., Mohapatra, S., and Rao, A. V., "Convergence Rate for a Gauss Collocation Method applied to Constrained Optimal Control," 2016, arXiv.org/abs/1607.02798.

[13] 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.

[14] 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.

[15] 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.

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

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

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

[19] Liu, F., Hager, W. W., and Rao, A. V., "Adaptive Mesh Refinement Method for Optimal Control Using Decay Rates of Legendre Polynomial Coefficients," *IEEE Transactions on Control System Technology*, , No. Accepted for Publication, June 2017, DOI: 10.1109/TCST.2017.2702122.

[20] 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.

[21] 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.

[22] 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.

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

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

[25] Meditch, J., "On the Problem of Optimal Thrust Programming for a Soft Lunar Landing," *IEEE Transaction on Automatic Control*, Vol. 9, No. 4, October 1964, pp. 477–484.

[26] Dolan, E., More, J. J., and Munson, T. S., "Benchmarking Optimization Software with CPOPS 3.0," Tech. Rep. ANL/MCS-273, Argonne National Laboratory, Argonne, IL, 2004.