JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 31, No. 5, September–October 2008

Trajectory Optimization Using Multiresolution Techniques

S. Jain* and P. Tsiotras†
Georgia Institute of Technology, Atlanta, Georgia 30332-0150

DOI: 10.2514/1.32220

We present a multi-resolution-based approach for solving trajectory optimization problems. The original optimal control problem is solved using a direct method, thereby being transcribed into a nonlinear programming problem that is solved using standard nonlinear programming codes. The novelty of the proposed approach hinges on the automatic calculation of a suitable nonuniform grid over which the nonlinear programming problem is subsequently solved. This tends to increase numerical efficiency and robustness. Control and/or state constraints are handled with ease and without any additional computational complexity. The proposed algorithm is based on a simple and intuitive method to balance conflicting objectives, such as accuracy of the solution, convergence, and speed of computations. The benefits of the proposed algorithm over uniform grid implementations are demonstrated with the help of several nontrivial examples.

I. Introduction

IT IS well known that the solution of realistic trajectory optimization problems is a challenging task. Analytical solutions are seldom available or even possible. As a result, more often than not, one resorts to numerical techniques [1–6]. Available numerical methods can be broadly divided into direct methods [1,3,7,8] and indirect methods [5,9,10]. Indirect methods solve the necessary optimality conditions, stated in terms of Pontryagin's minimum principle, the adjoint differential equations, and the transversality conditions. Direct methods, on the other hand, are based on discretizing the control and/or state variables at a set of nodes, transforming the optimal control problem into a nonlinear programming (NLP) problem. The solution of the resulting NLP problem can be obtained using standard NLP solvers. A nice survey of available numerical algorithms for solving trajectory optimization problems can be found in [11,12].

In recent years, direct transcription methods have become increasingly popular for solving trajectory optimization problems, the major reason being that direct methods do not need an explicit expression for the necessary conditions, which can be intimidating for complicated nonlinear dynamics. Moreover, incorporating state and control constraints is rather straightforward. Most important, experience has shown that direct methods tend to be more robust with respect to inaccurate initial guesses, thus converging more easily. Indirect methods, on the other hand, result in more accurate overall solutions than direct methods and provide more confidence in the (at least local) optimality of the obtained solution. Algorithms that aim at taking advantage of both direct and indirect methods by combining them into a single algorithm have been also proposed in the literature [10,13]. One of the main themes of current research on numerical trajectory optimization techniques is to develop methods that combine the accuracy of indirect methods with the robustness and good convergence properties of direct methods.

The algorithm proposed in this paper falls under the direct method category. Direct methods can be further broadly classified as either shooting methods [1,2,8,14] or collocation methods [3,7,15–17]. Direct collocation methods discretize the ordinary differential equations (ODEs) using collocation or interpolation schemes [18,19], along with the introduction of collocation conditions as NLP constraints, together with the initial and terminal conditions. Direct collocation methods can be further subdivided into pseudospectral methods [16,20–22] and other collocation methods [3,7,15,17,23]. In a sense, *pseudospectral* is a synonym for *collocation*, but the term pseudospectral is typically applied only when collocation is used in conjunction with a basis of global functions such as Chebyshev or Legendre polynomials. The other difference between pseudospectral and the rest of collocation methods is that pseudospectral methods use differentiation, whereas typical collocation methods are based on integration. In other words, pseudospectral methods rely on the discretization of the tangent bundle (roughly, the left-hand side of the differential equations), whereas most of collocation methods rely on the approximation of the vector field (i.e., the right-hand side of the differential equations).

Regardless of the particular method used, if a highly accurate solution is needed using one of the aforementioned direct methods, one must resort to the use of a high-resolution (dense) grid. This choice may lead to the use of a large amount of computational resources, both in terms of CPU time and memory, especially if the resulting NLP problem is not sparse. Therefore, recent work has focused on suitable sparse representations or on the reduction of the high computational load associated with uniform grid discretizations (see, for instance, the work by Betts et al. [7,15], Ross et al. [24,25], Gong and Ross [22], Binder et al. [8,26], and Schlegel et al. [27]).

In terms of mesh refinement algorithms, we should mention the work of Betts et al. [7,15], which selects the new grid points by solving an integer programming problem that minimizes the maximum discretization error by subdividing the current grid. The pseudospectral knotting method of Ross and Fahroo [25] generalizes the spectral patching method‡ of [21] by exchanging information across the patches in the form of event conditions associated with the optimal control problem, hence removing the restriction of continuity in the solution across the end points of the phases. The phase boundaries, termed as *knots*, can be fixed or free, with the free knots being part of the optimization process. On each phase, the problem is solved using the Legendre or Chebyshev pseudospectral method. To improve the pseudospectral methods, Gong and Ross [22] present an algorithm in which the user specifies the number of nodes to be increased within a particular phase, in case the error of the computed optimal control between two successive iterations is greater than a prescribed threshold. The authors of [22] use the gradient of the control to determine (approximately) the location of the knots. Binder et al. [8,26] use a wavelet-Galerkin approach to

Received 17 May 2007; revision received 21 February 2008; accepted for publication 25 February 2008.

*Ph.D. Candidate, School of Aerospace Engineering; sachin.jain@gatech.edu. Student Member AIAA.

†Professor, School of Aerospace Engineering; tsiotras@gatech.edu. Associate Fellow AIAA.

‡In the spectral patching method, the authors divide the time interval into several nonoverlapping continuous subintervals (or phases), where the continuity of states and controls is assumed at the interface of these intervals.

discretize the optimal control problem into an NLP problem. They use a local error analysis of the states and a wavelet analysis of the control profile to add or remove wavelet basis functions. In [26] the authors use a direct shooting approach in which the optimal control problem is converted into an NLP problem by parameterizing the control profile, combined with a wavelet analysis of the gradients of the Lagrangian function with respect to the parameterizing functions at the optimal points to determine the regions that require refinement. For problems with state and/or control path constraints, Schlegel et al. [27] use wavelet analysis of the control profile to determine the regions that require refinement.

The work of this paper continues the current research activities in the multiresolution trajectory optimization area and is motivated by our willingness to trade numerical accuracy for robustness and complexity for execution speed. For several applications (for instance, for onboard real-time guidance or during emergencies) it is of paramount importance to be able to control the accuracy of the solution (both locally and globally) to *promptly* obtain a good solution that can subsequently be further refined and improved upon as needed. For such cases, the overall result does not hinge upon the accuracy of a particular intermediate iteration step; imposing a high numerical accuracy from the outset is wasteful and may even hinder convergence. From a computational point of view, it is not advisable to use high-accuracy numerical schemes and/or dense integration grids during the initial iterations, because these have only a minor effect on the final solution (assuming, of course, that the solution is maintained within the region of convergence). For similar comments corroborating this point of view, see also [28]. What we propose is a progressive tightening of the tolerances at different levels of resolution. Furthermore, this progressive increase of accuracy should not be done uniformly, but only at the locations that dominate the overall accuracy of the solution. Finally, the algorithm should be simple enough so as not to add to the overall computational overhead.

Motivated by the previous observations in [29,30], we have introduced a novel multi-resolution-based mesh refinement technique for the solution of initial boundary-value problems for evolution equations. The algorithm results in a fewer number of nodes than with similar grid adaptation schemes, while maintaining the same overall accuracy of the final solution. Several challenging examples (namely, Burgers's equation and Euler's equations of gas dynamics) have demonstrated the stability and robustness of the mesh refinement algorithm for the solution of these types of problems in 1-D [30]. In the current paper, we use the ideas introduced in [29,30] to design a novel, fully automated, adaptive multiresolution trajectory optimization technique to solve optimal control problems quickly and accurately. The criterion for deciding the region to refine the mesh is based on simple interpolations, which tends to speed up the whole process. In a single step, the proposed algorithm adds and removes points from the grid, as necessary. Furthermore, all computations are performed on refinable complementary dyadic grids (see Sec. III.A). Working with dyadic grids is essentially equivalent to using interpolating wavelets [31–33] for the analysis of the underlying function. We can thus take advantage of the nice multiresolution properties of wavelets to obtain error estimates about the local smoothness properties of the solution [27,31,34].

Compared with previous similar results in this area [7,8,15,22,24,26,27], the algorithm proposed in this paper has several advantages. First, we avoid the solution of a secondary optimization problem for adding points to the mesh as in [7,15,35]. Only simple interpolations are needed to refine the mesh, which can be done on-the-fly. Furthermore, our algorithm does not involve any integrations, as opposed to the highly accurate integrations (Romberg quadratures) required in the method by Betts et al. [7], which again can be computationally expensive for nonlinear dynamics. Finally, our algorithm is capable of not only adding points to the grid but also removing points from the grid when and where is needed. Moreover, both the operations of adding and removing points can be done in a single step. In the pseudospectral knotting method of Gong and Ross [22] and Ross et al. [24], one needs to know a priori the approximate number and location of singularities in the solution; for most problems, these may not be known beforehand. The number of nodes to be added to a particular phase must be defined by the user before starting the algorithm. In our algorithm, the user need not know a priori the number nor the locations of the irregularities in the solution. The algorithm will automatically detect the regions in the solution that are nonsmooth. Furthermore, the nonuniform grids of pseudospectral methods result in grid distributions that remain fixed for each phase, because the location of the nodes are dictated by the zeros of the first derivative of the Legendre or Chebyshev polynomials, irrespective of the location of the soft knots [24]. Our algorithm uses a grid that is fully adaptive, embracing any form, depending on the nonsmooth characteristics of the solution. This provides more flexibility in capturing any irregularities in the solution.

From all previous references in this area the work of Binder et al. [8,26] and Schlegel et al. [27] are the closest, at least in spirit, to the approach proposed in the current paper. These references use wavelet-based ideas to locate possible singularities in the solution and to refine the grid locally. However, because these references work solely in the wavelet domain, they may lead to an increase of the overall computational overhead, because one needs to transform back and forth between the physical and wavelet domain. We avoid this issue altogether by always working in the physical domain. Nonetheless, by working with dyadic grids we still take advantage of the major benefit of the wavelet-based analysis: that is, multiresolution functional representations [31,36].

The rest of the paper is organized as follows. We first formulate the trajectory optimization problem and discretize the continuous optimal control problem into an NLP problem. Next, we present the multiresolution trajectory optimization algorithm (MTOA), followed by a section underpinning the rationale behind the proposed mesh refinement technique, along with a discussion on error estimates. We then give several nontrivial examples that show the robustness, efficiency, and accuracy of the proposed algorithm. We conclude with a brief summary of our results, along with some remaining related open problems and potential extensions.

II. Problem Statement

We wish to determine the state $\mathbf{x}(\cdot)$ and the control $\mathbf{u}(\cdot)$ that minimize the Bolza cost functional:

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

where

$$e\colon \mathbb{R}^{N_x} \times \mathbb{R}_+ \to \mathbb{R}, \qquad \tau \in [t_0, t_f], \qquad \mathbf{x}\colon [t_0, t_f] \to \mathbb{R}^{N_x},$$
$$\mathbf{u}\colon [t_0, t_f] \to \mathbb{R}^{N_u}, \qquad L\colon \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times [t_0, t_f] \to \mathbb{R}$$

subject to the state dynamics

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \tag{2}$$

the boundary conditions

$$\mathbf{x}(t_0) = \mathbf{x}_0, \qquad \mathbf{e}_f(\mathbf{x}(t_f), t_f) = 0 \tag{3}$$

where $\mathbf{e}_f\colon \mathbb{R}^{N_x} \times \mathbb{R}_+ \to \mathbb{R}^{N_e}$, and the constraints

$$\mathbf{C}_u(\mathbf{u}(t)) \le 0, \qquad \mathbf{C}_x(\mathbf{x}(t)) \le 0, \qquad \mathbf{C}_{xu}(\mathbf{x}(t), \mathbf{u}(t)) \le 0 \tag{4}$$

where

$$\mathbf{C}_u\colon \mathbb{R}^{N_u} \to \mathbb{R}^{N_{C_u}}, \quad \mathbf{C}_x\colon \mathbb{R}^{N_x} \to \mathbb{R}^{N_{C_x}}, \quad \mathbf{C}_{xu}\colon \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \to \mathbb{R}^{N_{C_{xu}}}$$

The initial time t_0 is assumed to be given and the final time t_f can be fixed or free.

Without loss of generality, we will assume that the time interval of interest is the unit interval, that is, $t \in [0, 1] = [t_0, t_f]$. The transformation from $[t_0, t_f]$ to [0, 1] is trivial when t_f is fixed. For a free-final-time problem, one can use an appropriate transformation of

the independent variable and the dynamics to cast the problem as a fixed-final-time problem. Because this is a well-known fact, the details are omitted.

We next transcribe the continuous optimal control problem [Eqs. (1–4)] over the interval [0, 1] into an NLP problem over dyadic grids. This process is described in the next section.

III. NLP Formulation

A. Dyadic Grids

All discretizations of the state dynamics, constraints, and performance index in Eqs. (1–4) will be performed on (nonuniform) grids induced by dyadic grids. A uniform dyadic grid over the unit interval is a collection of points of the form

$$\mathcal{V}_j = \{t_{j,k} \in [0,1]: t_{j,k} = k/2^j, 0 \le k \le 2^j\}, \quad J_{\min} \le j \le J_{\max} \tag{5}$$

where j denotes the resolution level, k is the spatial location, and $J_{\min}$ and $J_{\max}$ are positive integers. We denote by $\mathcal{W}_j$ the set of grid points belonging to $\mathcal{V}_{j+1} \setminus \mathcal{V}_j$; that is,

$$\mathcal{W}_j = \{\hat{t}_{j,k} \in [0,1]: \hat{t}_{j,k} = (2k+1)/2^{j+1}, 0 \le k \le 2^j - 1\}, \quad J_{\min} \le j \le J_{\max} - 1 \tag{6}$$

Hence, $t_{j+1,k} \in \mathcal{V}_{j+1}$ if and only if

$$t_{j+1,k} = \begin{cases} t_{j,k/2}, & \text{if } k \text{ is even} \\ \hat{t}_{j,(k-1)/2}, & \text{otherwise} \end{cases} \tag{7}$$

An example of a dyadic grid with $J_{\min} = 0$ and $J_{\max} = 5$ is shown in Fig. 1.

The subspaces $\mathcal{V}_j$ are nested; that is, $\mathcal{V}_{J_{\min}} \subset \mathcal{V}_{J_{\min}+1} \cdots \subset \mathcal{V}_{J_{\max}}$, with

$$\lim_{J_{\max} \to \infty} \overline{\mathcal{V}_{J_{\max}}} = [0,1] \tag{8}$$

where the overbar denotes set closure. Furthermore, the sequence of subspaces $\mathcal{W}_j$ satisfy the property $\mathcal{W}_j \cap \mathcal{W}_\ell = \varnothing$ for all $j \ne \ell$. Notice that these dyadic grids are constructed by successive subdivisions. Furthermore, the sets $\mathcal{V}_j$ and $\mathcal{W}_k$ for $k \ge j$ are orthogonal to each other, as are the sets $\mathcal{W}_j$ and $\mathcal{W}_\ell$ for $j \ne \ell$ (see Fig. 1). Such subdivision schemes of dyadic grids can be used to construct interpolating wavelets using, for instance, the scheme of Deslauriers and Dubuc [32], independently discovered later by Donoho [33] and Harten [37]. The idea here is that one can keep only the even-indexed points in the grid and generate the odd-indexed grid points (at every level) using polynomial interpolation. The use of subdivision schemes simplifies the computations, because it eliminates the need to constantly transform back and forth between the physical domain and the wavelet/scaling coefficient domain, as was done, for example, in [27].

B. Discretizations on Dyadic Grids

For simplicity, henceforth we denote $\mathbf{x}_{j,k} = \mathbf{x}(t_{j,k})$ and $\mathbf{u}_{j,k} = \mathbf{u}(t_{j,k})$. We convert the optimal control problem [Eqs. (1–4)] into an NLP problem using Runge–Kutta (RK) discretizations. To this end, let a nonuniform grid of the form

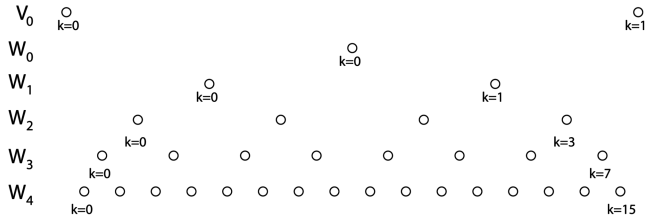


Fig. 1 Example of a dyadic grid.

$$\begin{aligned} \mathbb{G} = \{&t_{j_i,k_i}: t_{j_i,k_i} \in [0,1], 0 \le k_i \le 2^{j_i}, J_{\min} \le j_i \le J_{\max}, \\ &\text{for } i = 0, \ldots, N, \quad \text{and} \quad t_{j_i,k_i} < t_{j_{i+1},k_{i+1}}, \\ &\text{for } i = 0, \ldots, N-1\} \end{aligned} \tag{9}$$

Then a q-stage RK method for discretizing Eq. (2) is given by [11,35]

$$\mathbf{x}_{j_{i+1},k_{i+1}} = \mathbf{x}_{j_i,k_i} + h_{j_i,k_i} \sum_{\ell=1}^{q} \beta^\ell \mathbf{f}^\ell_{j_i,k_i} \tag{10}$$

where

$$\mathbf{f}^{\,\ell}_{j_i,k_i} = \mathbf{f}(\mathbf{y}^\ell_{j_i,k_i}, \mathbf{u}^\ell_{j_i,k_i}, t^\ell_{j_i,k_i})$$

$\mathbf{y}^\ell_{j_i,k_i}$, $\mathbf{u}^\ell_{j_i,k_i}$, and $t^\ell_{j_i,k_i}$ are the intermediate state, control, and time variables on the interval $[t_{j_i,k_i}, t_{j_{i+1},k_{i+1}}]$. These are given by

$$\mathbf{y}^{\,\ell}_{j_i,k_i} = \mathbf{x}_{j_i,k_i} + h_{j_i,k_i} \sum_{m=1}^{q} \alpha^{\ell,m} \mathbf{f}^m_{j_i,k_i} \tag{11}$$

where

$$\begin{aligned} &h_{j_i,k_i} = t_{j_{i+1},k_{i+1}} - t_{j_i,k_i}, \qquad t^\ell_{j_i,k_i} = t_{j_i,k_i} + h_{j_i,k_i}\rho^\ell, \\ &\mathbf{u}^\ell_{j_i,k_i} = \mathbf{u}(t^\ell_{j_i,k_i}), \quad \text{for } 1 \le \ell \le q \end{aligned}$$

The variable q denotes the *stage* of the RK method. In the previous expressions, ρ^ℓ, β^ℓ, and $\alpha^{\ell,m}$ are known constants with $0 \le \rho^1 \le \rho^2 \le \ldots \le 1$. The scheme is explicit if $\alpha^{\ell,m} = 0$ for $m \ge \ell$ and implicit otherwise. Some common examples of q-stage RK methods are the trapezoidal method ($q = 2$), the Hermite–Simpson method ($q = 3$), and the classical fourth-order RK method ($q = 4$) [11,35,38].

Using Eq. (10), the defects of the discretization are given by

$$\zeta_i = \mathbf{x}_{j_{i+1},k_{i+1}} - \mathbf{x}_{j_i,k_i} - h_{j_i,k_i} \sum_{\ell=1}^{q} \beta^\ell \mathbf{f}^\ell_{j_i,k_i} \tag{12}$$

for $i = 0, \ldots, N-1$. To discretize the cost functional (1), we introduce a new state $z(t)$ such that

$$\dot{z}(t) = L(\mathbf{x}(t), \mathbf{u}(t), t), \qquad z(0) = 0 \tag{13}$$

Using a q-stage RK method to discretize Eq. (13) yields

$$z_{j_{i+1},k_{i+1}} = z_{j_i,k_i} + h_{j_i,k_i} \sum_{\ell=1}^{q} \beta^\ell L^\ell_{j_i,k_i} \tag{14}$$

where $L^\ell_{j_i,k_i} = L(\mathbf{y}^\ell_{j_i,k_i}, \mathbf{u}^\ell_{j_i,k_i}, t^\ell_{j_i,k_i})$ for $i = 0, \ldots, N-1$. Hence, we have

$$z_{j_N,k_N} = z_{j_0,k_0} + \sum_{i=0}^{N-1} h_{j_i,k_i} \sum_{\ell=1}^{q} \beta^\ell L^\ell_{j_i,k_i} \tag{15}$$

Because $z(0) = z_{j_0,k_0} = 0$, the cost functional (1) in discretized form can be written as follows:

$$J = e(\mathbf{x}_{j_N,k_N}) + \sum_{i=0}^{N-1} \left(h_{j_i,k_i} \sum_{\ell=1}^{q} \beta^\ell L^\ell_{j_i,k_i} \right) \tag{16}$$

Let us now define the following sets:

$$\begin{aligned} &\mathbf{X} = \{\mathbf{x}_{j_0,k_0}, \ldots, \mathbf{x}_{j_N,k_N}\}, \qquad \mathbf{U} = \{\mathbf{u}_{j_0,k_0}, \ldots, \mathbf{u}_{j_N,k_N}\}, \\ &\tilde{\mathbb{G}} = \{t^\ell_{j_i,k_i} \in [0,1]: t^\ell_{j_i,k_i} \notin \mathbb{G}, 0 \le i < N, 1 \le \ell \le q\}, \\ &\tilde{\mathbf{X}} = \{\mathbf{y}^\ell_{j_i,k_i}: t^\ell_{j_i,k_i} \in \tilde{\mathbb{G}}\}, \qquad \tilde{\mathbf{U}} = \{\mathbf{u}^\ell_{j_i,k_i}: t^\ell_{j_i,k_i} \in \tilde{\mathbb{G}}\} \end{aligned}$$

As a result of the discretization, the optimal control problem reduces to the following NLP problem in terms of the variables $\mathbf{X}$, $\mathbf{U}$, and $\tilde{\mathbf{U}}$.

Minimize

$$J = e(\mathbf{x}_{j_N,k_N}) + \sum_{i=0}^{N-1}\left(h_{j_i,k_i}\sum_{\ell=1}^{q}\beta^{\ell}L^{\ell}_{j_i,k_i}\right) \tag{17}$$

subject to the constraints

$$\zeta_i = 0, \qquad i = 0,\ldots,N-1 \tag{18}$$

$$\mathbf{x}_{j_0,k_0} = \mathbf{x}_0 \tag{19}$$

$$\mathbf{e}_f(\mathbf{x}_{j_N,k_N}) = 0 \tag{20}$$

$$\mathbf{C}_u(\mathbf{U},\tilde{\mathbf{U}}) \le 0 \tag{21}$$

$$\mathbf{C}_x(\mathbf{X},\tilde{\mathbf{X}}) \le 0 \tag{22}$$

$$\mathbf{C}_{xu}(\mathbf{X},\tilde{\mathbf{X}},\mathbf{U},\tilde{\mathbf{U}}) \le 0 \tag{23}$$

Remark 1. It is well known [39,40] that RK discretizations for optimal control problems need to satisfy additional assumptions to obtain consistent approximations. Henceforth, we will therefore assume that the following conditions always hold:

1) If the optimal control problem does not have any constraints, or if the optimal control problem has only pure control constraints, then by RK discretizations we mean RK discretizations that satisfy the conditions in [39].

2) If the optimal control problem has only pure control constraints, the coefficients of the RK scheme satisfy the conditions given in [40] or [39].

3) If the optimal control problem has state or mixed state/control constraints, then by RK discretizations we mean either Euler, trapezoidal, or Hermite–Simpson discretization.

The restriction to the aforementioned schemes stems from the fact that the convergence of these schemes for optimal control problems has been demonstrated in the literature [35,39–42]. Nonetheless, we point out that the proposed mesh refinement approach will work with any RK discretization for which the convergence for the optimal control problems can be shown, using either uniform or nonuniform meshes.

We are now ready to present the proposed multi-resolution-based trajectory optimization algorithm.

IV. Multiresolution Trajectory Optimization

Consider a set of dyadic grids $\mathcal{V}_j$ and $\mathcal{W}_j$ as described in Eqs. (5) and (6). Suppose $g\colon \tilde{\mathcal{D}} \to \mathbb{R}$ is specified on a grid $\mathbb{G}$ [see Eq. (9)], such that

$$\mathcal{U} = \{g_{j,k}\colon t_{j,k} \in \mathbb{G}\} \tag{24}$$

where $g_{j,k} = g(t_{j,k})$. Let $\mathcal{I}^p(t;\mathcal{T}_{\mathbb{G}}(t))$ denote the pth-order essentially nonoscillatory (ENO) interpolation [31] of

$$\mathcal{U} = \{g_{j,k}\colon t_{j,k} \in \mathcal{T}_{\mathbb{G}}(t)\}$$

where

$$\mathcal{T}_{\mathbb{G}}(t) = \{t_{j_m,k_m}\}_{m=i}^{i+p} \subseteq \mathbb{G}, \qquad 0 \le i \le N-p-1$$

The stencil $\mathcal{T}_{\mathbb{G}}(t)$ consists of one neighboring point on the left and one neighboring point on the right of t in the set $\mathbb{G}$, with the remaining $p-1$ points selected from the set $\mathbb{G}$ in a way such that the resulting polynomial is least oscillatory. For more details on ENO interpolations the reader is referred to [31].

To proceed with the algorithm, we first choose the minimum resolution level $J_{\min}$ based on the minimum time step required to achieve the desired accuracy in the regions of the solution where no constraints are active,[§] the threshold ϵ, which should be at least on the order of $h_{J_{\min}}$, where $h_{J_{\min}} = 1/2^{J_{\min}}$ (the significance of ϵ and the reason for such a choice of ϵ will be clear shortly), and pick the maximum resolution level $J_{\max}$. The proposed MTOA involves the following steps. First, we transcribe the continuous trajectory optimization problem into an NLP problem using a q-stage RK discretization, as described in the previous section. We use trapezoidal discretization for the first iteration and switch to a high-order discretization for subsequent iterations. Next, we set iter $= 1$, initialize $\text{grid}_{\text{iter}} = \mathcal{V}_{J_{\min}}$, and choose an initial guess for all NLP variables. Let us denote the set of initial guesses by $\mathcal{X}_{\text{iter}}$. The proposed MTOA then proceeds as follows:

Multiresolution Trajectory Optimization Algorithm:

1) Solve the NLP problem on $\text{grid}_{\text{iter}}$ with the initial guess $\mathcal{X}_{\text{iter}}$. If $\text{grid}_{\text{iter}}$ has points from the level $\mathcal{W}_{J_{\max}-1}$, terminate.

2) Mesh refinement:

a) If the problem has either pure state constraints or mixed constraints on the states and controls, set $\Phi_{\text{iter}} = \{\mathbf{x}_{j,k}, \mathbf{u}_{j,k}\colon t_{j,k} \in \text{grid}_{\text{iter}}\}$ and $N_r = N_x + N_u$.

b) If the optimal control problem does not have any constraints, or if only pure control constraints are present, set $\Phi_{\text{iter}} = \{\mathbf{u}_{j,k}\colon t_{j,k} \in \text{grid}_{\text{iter}}\}$ and $N_r = N_u$.

c) In case no controls are present in the problem, set $\Phi_{\text{iter}} = \{\mathbf{x}_{j,k}\colon t_{j,k} \in \text{grid}_{\text{iter}}\}$.

In the following, let Φ_{iter} denote the set constructed in steps 2a–2c of the algorithm; that is, let

$$\Phi_{\text{iter}} = \{\phi_\ell(t_{j,k})\colon \ell = 1,\ldots,N_r, t_{j,k} \in \text{grid}_{\text{iter}}\}$$

d) Initialize an intermediate grid $\text{grid}_{\text{int}} = \mathcal{V}_{J_{\min}-1}$, with function values

$$\Phi_{\text{int}} = \{\phi_\ell(t_{J_{\min},k}) \in \Phi_{\text{iter}}, 0 \le k \le 2^{J_{\min}}, \ell = 1,\ldots,N_r\} \tag{25}$$

and set $j = J_{\min} - 1$.

e) While $j < J_{\max}$ do

f) Procedure_MR

g) End while

h) The final nonuniform grid is $\text{grid}_{\text{new}} = \text{grid}_{\text{int}}$ and the corresponding function values are in the set $\Phi_{\text{new}} = \Phi_{\text{int}}$.

3) Set iter $=$ iter $+ 1$. If the number of points and the level of resolution remain the same after the mesh refinement procedure, terminate. Otherwise interpolate the NLP solution found in step 1 on the new mesh grid_{new} (which will be the new initial guess $\mathcal{X}_{\text{iter}}$), resassign the set $\text{grid}_{\text{iter}}$ to grid_{new}, and go to step 1.

Next, we give the algorithm Procedure_MR (step 2f of MTOA).

Procedure_MR:

1) Find the points that belong to the intersection of $\mathcal{W}_j$ and $\text{grid}_{\text{iter}}$:

$$\hat{T}_j = \{\hat{t}_{j,k_i}\colon \hat{t}_{j,k_i} \in \mathcal{W}_j \cap \text{grid}_{\text{iter}}, \quad \text{for } i = 1,\ldots,N_{\hat{t}}, 1 \le N_{\hat{t}} \le 2^j - 1\} \tag{26}$$

If $\hat{T}_j$ is empty, set $j = J_{\max}$ and terminate Procedure_MR.

2) Set $i = 1$.

3) While $i \le N_{\hat{t}}$ do

a) Compute the interpolated function values at $\hat{t}_{j,k_i} \in \hat{T}_j$ and

$$\hat{\phi}_\ell(\hat{t}_{j,k_i}) = \mathcal{I}^p(\hat{t}_{j,k_i}, \mathcal{T}_{\text{grid}_{\text{int}}}(\hat{t}_{j,k_i}))$$

where $\hat{\phi}_\ell$ is the ℓth element of $\hat{\phi}$, for $\ell = 1,\ldots,N_r$.

[§]The minimum time step required to achieve a desired accuracy in the regions of the solution where no constraints are active can be calculated using well-known error estimation formulas for RK schemes [39,42–44].

b) Calculate the interpolative error coefficient d_{j,k_i} at the point $\hat{t}_{j,k_i}$[¶]:

$$d_{j,k_i}(\phi) = \max_{\ell=1,\ldots,N_r} d_{j,k_i}(\phi_\ell) = \max_{\ell=1,\ldots,N_r} |\phi_\ell(\hat{t}_{j,k_i}) - \hat{\phi}_\ell(\hat{t}_{j,k_i})| < \epsilon \quad (27)$$

If the value of d_{j,k_i} is below the threshold ϵ, then reject $\hat{t}_{j,k_i}$ and go to step 2e, otherwise add $\hat{t}_{j,k_i}$ to the intermediate grid grid_{int} and move on to the next step.

c) Add to grid_{int} points belonging to the set

$$(\mathcal{V}_{\hat{j}} \cap [t_{j,k_i}, t_{j,k_i+1}]) \setminus \text{grid}_{\text{int}}$$

where $\hat{J} = \min\{j + \hat{j}, J_{\max}\}$, $\hat{j} = 2$ if $\texttt{iter} = 1$, and $\hat{j} \geq 2$ if $\texttt{iter} \geq 1$. Here, $\hat{j}$ is the number of finer levels from which the points are added to the grid for refinement. In particular, we add to the intermediate grid grid_{int} the points

$$\{t_{j,k} : 2^{\hat{J}-j} k_i \leq k \leq 2^{\hat{J}-j}(k_i + 1)\} \setminus \text{grid}_{\text{int}}$$

d) Add the function values at all the newly added points to Φ_{int}. If the function value at any of the newly added points is not known, interpolate the function value at that point from the points in $\text{grid}_{\text{iter}}$ and their function values in $\mathcal{X}_{\text{iter}}$ using $\mathcal{I}^p(\cdot, \mathcal{T}_{\text{grid}_{\text{iter}}}(\cdot))$.

e) Set $i = i + 1$.

4) End while.

5) Set $j = j + 1$.

The order of the interpolating polynomial p can be taken to be one less than the order of the RK discretization of the differential equations. This choice of p is dictated by the error analysis given in the next section, which considers the case with no constraints. It is noted that under the presence of constraints, the order of the RK discretization for optimal control problems may be less than the order of the RK discretization used for the differential equations [39]. The subsequent analysis, albeit heuristic, elucidates the motivation behind the proposed approach and the previous choice of p. Although a more rigorous analysis is required to justify the recommended choice for the order of the interpolating polynomials (hence the order of the RK discretization as well), nonetheless, in all numerical examples we considered, choosing the interpolating polynomial according to the previous criterion turned out to be adequate, irrespective of the presence (or not) of the constraints.

V. Rationale of Proposed Multiresolution Scheme and Error Estimates

In this section, we outline the main idea behind the multiresolution mesh refinement algorithm. In the process, we also provide estimates on the error one expects to obtain by following the proposed approach. To keep the notation as simple as possible, the subsequent discussion will be restricted to the case of a scalar-valued control function u. Furthermore, we will consider a problem without state and control constraints, so that the refinement algorithm is performed based on the (scalar-valued) control histories [case 2b of MTOA with $N_u = N_r = 1$].

The key idea behind the proposed mesh refinement algorithm is based on the fact that the interpolative error coefficient in step 3b of the procedure_MR, and for a sufficiently fine grid, provides a good measure of the local smoothness of the function u. To see why this is so, consider a function u that, at $t = \bar{t}$, has $\nu \geq -1$ continuous derivatives,[**] but that has a jump discontinuity in its $(\nu + 1)$th derivative. Locally around any point $t \neq \bar{t}$, the function u can be approximated accurately by a polynomial (say, $\hat{u}$) of degree ν. Furthermore, in the neighborhood of $\bar{t}$, any interpolating polynomial of degree at least $\nu + 1$ will induce an error that is proportional to the jump discontinuity of $u^{(\nu+1)}$.

The proposed algorithm uses the information of the local interpolation error in Eq. (27) to locally refine the grid, if necessary. In particular, at the locations where the solution is smooth (hence it can be accurately interpolated by neighboring points), no further refinement is performed. At those locations where the function is not smooth, grid points are added to reduce the interpolation error below a certain threshold.

To this end, let the final grid at a certain iteration step be given by the points $\mathsf{G} = \{t_0, t_1, \ldots, t_N\}$. For each point $t_i \in \mathsf{G}$, $(0 \leq i \leq N)$, let

$$\mathcal{T}_{\mathsf{G}}(t_i) = \{\tau_0^i, \tau_1^i, \ldots, \tau_p^i\} \in \mathsf{G} \setminus \{t_i\}$$

where $\tau_0^i < \cdots < \tau_p^i$ is the stencil of $p + 1$ points that are used to interpolate the function u in the interval $[\tau_0^i, \tau_p^i]$, according to the discussion in the previous section. That is, let $\hat{u}$ be the unique polynomial of degree p, such that

$$\hat{u}(\tau_m^i) = u(\tau_m^i), \qquad 0 \leq m \leq p, \qquad 0 \leq i \leq N \quad (28)$$

and

$$u(t) = \hat{u}(t) + u[\tau_0^i, \ldots, \tau_p^i, t] \Pi_{m=0}^p (t - \tau_m^i), \qquad \tau_0^i \leq t \leq \tau_p^i \quad (29)$$

where $u[\tau_0^i, \ldots, \tau_p^i, t]$ is the $(p + 2)$-divided difference of u at the points of the stencil $\mathcal{T}_{\mathsf{G}}(t_i)$ and t. Moreover, if u is sufficiently smooth (i.e., is continuously differentiable at least $\nu \geq p + 1$ times) in the interval $[\tau_0^i, \tau_p^i]$, then [45,46]

$$u[\tau_0^i, \ldots, \tau_p^i, t] = \frac{u^{(p+1)}(\xi)}{(p+1)!}, \qquad \tau_0^i \leq \xi \leq \tau_p^i \quad (30)$$

It then follows from Eq. (29) that

$$d_i(u) = |u(t_i) - \hat{u}(t_i)| \approx |u^{(p+1)}| h_i^{p+1}, \qquad (\nu \geq p + 1) \quad (31)$$

where

$$h_i = \max_{0 \leq m \leq p-1} (\tau_{m+1}^i - \tau_m^i)$$

In Eq. (31) the notation $\approx$ indicates a term of the same order of magnitude. Similarly, the notation $\lesssim$ will be used to indicate a term dominated by an expression of a known order of magnitude.

If, on the other hand, u has a jump discontinuity in its $(\nu + 1)$ derivative and $\nu < p + 1$, then [37,45]

$$u[\tau_0^i, \ldots, \tau_p^i, t] \approx \frac{[[u^{(\nu+1)}]]}{h_i^{p-\nu}} \quad (32)$$

where $[[u^{(\nu+1)}]]$ denotes the jump at the discontinuity of the $(\nu + 1)$th derivative of u inside the interval $[\tau_0^i, \tau_p^i]$. It follows from Eq. (29) that, in this case, we have the estimate

$$d_i(u) = |u(t_i) - \hat{u}(t_i)| \approx [[u^{(\nu+1)}]] h_i^{\nu+1}, \qquad (\nu < p + 1) \quad (33)$$

It has been shown in [39] that under appropriate smoothness and coercivity hypotheses [39], and assuming that the solution $u^\star$ of the continuous optimal control problem [Eqs. (1–3)] is at least $\nu = p - 1$ continuous differentiable, the following estimate,

$$\max_{0 \leq i \leq N} |\mathbf{x}_i - \mathbf{x}^\star(t_i)| + \max_{0 \leq i \leq N} |u_i - u^\star(t_i)| \lesssim h^{p+1} + h^p \int_0^1 \omega(u^{\star(p)}, [0, 1]; t, h)\, \mathrm{d}t \quad (34)$$

holds for sufficiently small $h = \max_{0 \leq i \leq N}\{t_{i+1} - t_i\}$, and where $\omega(v, [a, b]; t, h)$ denotes the local modulus of continuity of the function v, defined by [47]

$$\omega(v, [a, b]; t, h) = \sup\{|v(\sigma_1) - v(\sigma_2)| : \sigma_1, \sigma_2 \in [t - h/2, t + h/2] \cap [a, b]\} \quad (35)$$

In Eq. (34) it was assumed that the optimal solution $(\mathbf{x}_i, u_i)$ of the discrete problem [Eqs. (18–21)] is computed using a $p + 1$th-order RK scheme, satisfying the Hager [39] conditions.

[¶]Note that $\phi_\ell(\hat{t}_{j,k}) \in \Phi_{\text{iter}}$ for all $\hat{t}_{j,k} \in \hat{T}_j$ and $\ell = 1, \ldots, N_r$.

[**]This notation implies that for $\nu = -1$ the function is discontinuous.

The MTOA estimates the last term in Eq. (34) using the local interpolating error for the control. To see why this is true, rewrite the last term in Eq. (34) as follows:

$$\int_0^1 \omega(u^{\star(p)}, [0,1]; t, h)\,\mathrm{d}t = \sum_{i=0}^{N-1} \int_{t_i}^{t_{i+1}} \omega(u^{\star(p)}, [0,1]; t, h)\,\mathrm{d}t$$

$$= \sum_{i=0}^{N-1} \int_{t_i}^{t_{i+1}} \omega(u^{\star(p)}, [t_i', t_i'']; t, h)\,\mathrm{d}t \tag{36}$$

where $t_i' = 3t_i/2 - t_{i+1}/2$ and $t_i'' = 3t_{i+1}/2 - t_i/2$. Using the definition of the modulus of continuity (35) and the estimate (32), we have

$$\omega(u^{\star(p)}, [t_i', t_i'']; t, h) \le \sup_{\sigma_1,\sigma_2 \in [t_i', t_i'']} |u^{\star(p)}(\sigma_1) - u^{\star(p)}(\sigma_2)| \approx [[u^{\star(p)}]] \tag{37}$$

It follows that

$$\int_{t_i}^{t_{i+1}} \omega(u^{\star(p)}, [t_i', t_i'']; t, h)\,\mathrm{d}t \lesssim h_i^{1-p} d_i(u^\star) \tag{38}$$

Because the MTOA ensures the bound $|d_i(u^\star)| \le \epsilon$, we finally get the estimate

$$h^p \int_0^1 \omega(u^{\star(p)}, [0,1]; t, h)\,\mathrm{d}t \lesssim \epsilon \tag{39}$$

recalling that

$$\sum_{i=0}^{N} h_i \approx 1$$

It follows that

$$\max_{0\le i\le N} |\mathbf{x}_i - \mathbf{x}^\star(t_i)| + \max_{0\le i\le N} |u_i - u^\star(t_i)| \lesssim h^{p+1} + \epsilon \tag{40}$$

Given the general grid in Eq. (9), it follows from Eq. (40) that if we chose $\epsilon \approx h_{J_{\min}}^{p+1}$, where

$$h_{J_{\min}} = t_{J_{\min},k+1} - t_{J_{\min},k} = 1/2^{J_{\min}}, \qquad 0 \le k \le 2^{J_{\min}} - 1$$

we get an estimate of the form

$$\max_{0\le i\le N} |\mathbf{x}_{j_i,k_i} - \mathbf{x}^\star(t_{j_i,k_i})| + \max_{0\le i\le N} |u_{j_i,k_i} - u^\star(t_{j_i,k_i})| \lesssim h_{J_{\min}}^{p+1} \tag{41}$$

Remark 2. As pointed out by Hager [39], the error in the discrete controls u_i may be one or more orders larger that the error obtained if the control were computed by the minimization of the Hamiltonian and by using the discrete state/costate pair instead. Hence, ideally, the approximation order in the right-hand side of Eq. (34) will be one or more orders less that $p+1$, even if a $p+1$th RK order is used. The interested reader may refer to [39] for further details in regard to this observation. Because here we are only interested in rough error estimates, the exact order of convergence for the discrete controls is immaterial for the overall analysis (for example, use a higher order RK scheme if needed) and is well beyond the scope of the paper.

VI. Numerical Examples

In this section, we provide several examples to demonstrate the robustness and efficiency of the proposed approach for the solution of optimal control problems. For all cases, we have used SNOPT [48] to solve the resulting NLP problem [Eqs. (17–23)]. SNOPT is an NLP solver, which is based on sequential quadratic programming. All computations were performed in MATLAB on a Pentium IV machine with a 3-GHz processor and 2 GB of RAM. In all of the following examples, and unless stated otherwise, a linear function was used as an initial guess for the first iteration of MTOA. We also used the implicit Hermite–Simpson scheme for the high-order discretizations in MTOA, the defects of which can be found in [35].

Example 1. We first consider a simple minimum-energy problem with a second-order state variable inequality constraint, taken from [49]. Because the analytic solution for this problem is known, we can infer the *absolute* accuracy of the solution provided by the proposed MTOA.

The problem is to find the control $u(t)$ that minimizes the cost function

$$J = \frac{1}{2}\int_0^1 u^2(t)\,\mathrm{d}t \tag{42}$$

subject to the dynamics

$$\dot{x} = v, \qquad \dot{v} = u \tag{43}$$

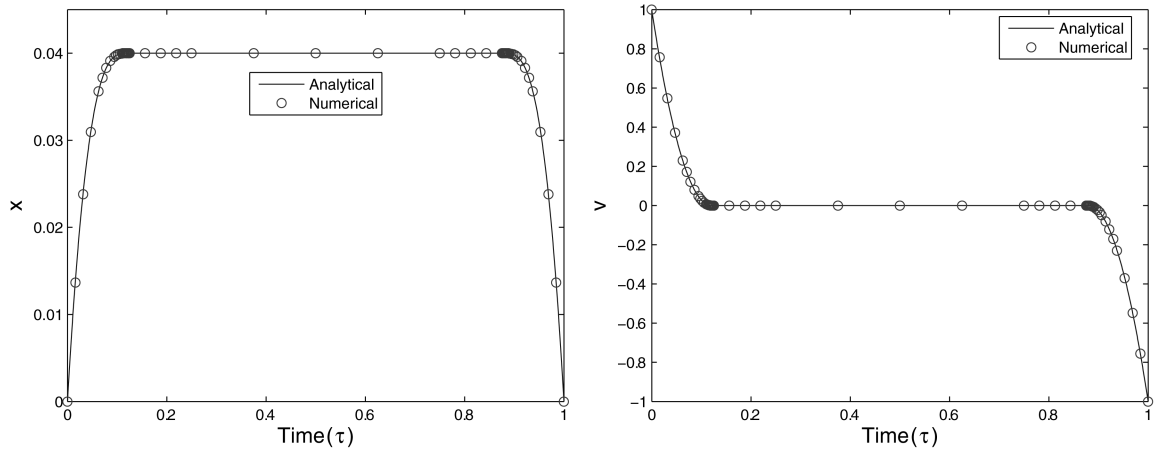
initial and final conditions

$$x(0) = x(1) = 0, \qquad v(0) = -v(1) = 1 \tag{44}$$

and the path constraint

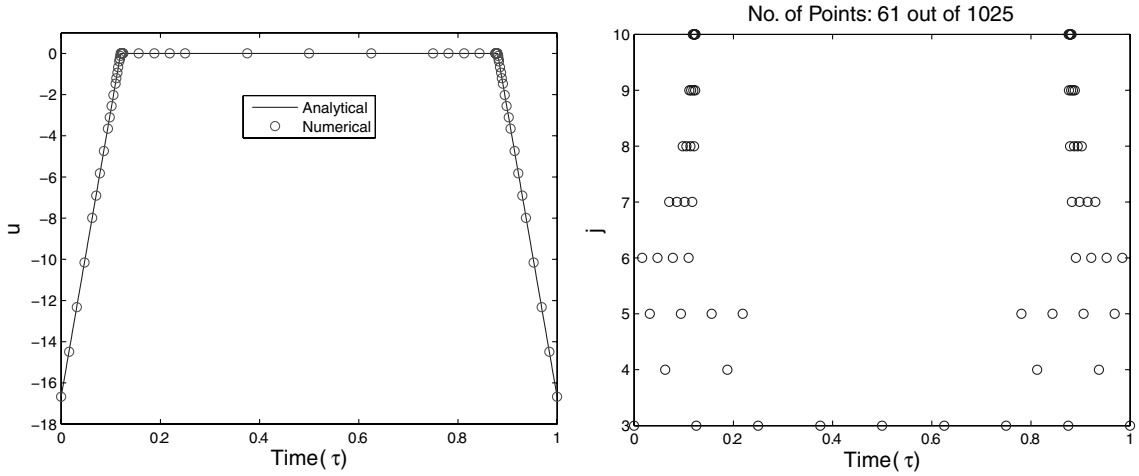
$$x(t) \le 0.04 \tag{45}$$

We solved this problem on a grid with $J_{\min} = 3$ and $J_{\max} = 10$. The threshold used was $\epsilon = 10^{-4}$. The algorithm terminated in 5 iterations. The time histories of the states x and v at the final iteration are shown in Fig. 2. The time history of the control u and the grid point distribution at the final iteration are shown in Fig. 3. It should be noted that the proposed algorithm used only 61 points out of the maximum of 1025 points at the finest resolution grid $\mathcal{V}_{10}$. Because



a) Iteration 8: time history of x **b) Iteration 8: time history of v**

Fig. 2 Example 1: time history of x and v at the final iteration of MTOA.



a) Iteration 8: time history of u **b) Iteration 8: grid point distribution**

Fig. 3 Example 1: time history of u along with the grid point distribution at the final iteration of MTOA.

the analytic solution of this problem is known [49], the absolute error can be computed for all cases. The errors in the computed solution, along with the number of grid points (N_{iter}) used by the algorithm at each iteration, are shown in Table 1. As shown in Table 1, the numerical solution converges to the analytic solution and, with each iteration, the errors are decreasing by roughly an order of magnitude.

The overall CPU time taken by MTOA to solve this problem was 5.6 s. For comparison, we also solved the same problem using a Hermite–Simpson discretization on a uniform grid with the same number of points as in MTOA at the final iteration: that is, on a uniform mesh with 61 nodes. The algorithm terminated in 2.5 s with the errors shown in Table 2. Because the errors in the solution using a uniform mesh with 61 nodes are larger than those achieved using MTOA, we also gradually increased the number of nodes in the uniform mesh and resolved the problem using the same linear initial guess until either the errors were of the same order of magnitude as those obtained using the MTOA or the CPU time taken by the algorithm was approximately equal to the CPU time taken by MTOA. This process ended up in a uniform mesh of 131 nodes. The algorithm terminated in 5.7 s and the final errors are shown in Table 2. These results show a typical trend we observed in all examples we tested and demonstrate the efficacy of the MTOA: higher accuracy for the same CPU time or a smaller number of grid points and CPU time for the same accuracy, compared with uniform grid implementations.

Example 2. Here, we consider a problem derived from the control of a chemical reaction [15,50]. The problem is to maximize the final amount of product y during a two-stage chemical reaction, $x \to y \to z$, by a proper choice of the rate coefficient $u(t)$. The amount of waste product z formed does not influence x and y, and because the magnitude of z is of no interest, we may consider only the reaction rates for x and y, which are given by

$$\dot{x} = -ux \tag{46}$$

$$\dot{y} = ux - \rho u^k y \tag{47}$$

where ρ and k are positive constants. For this example we consider the same parameters as in [15,50]

$$\rho = 2.5, \qquad k = 1.5, \qquad t_f = 2 \tag{48}$$

and initial conditions

$$x(0) = 1, \qquad y(0) = 0.01 \tag{49}$$

The allowable control must lie within the range

$$0.1 \le u(t) \le u_{\max} \tag{50}$$

We solved this problem on a grid with $J_{\min} = 3$ and $J_{\max} = 6$ for three different choices of $u_{\max}$: namely, $u_{\max} = 0.5, 0.4$, and 0.3. The threshold used in the simulations was $\epsilon = 10^{-4}$. The algorithm terminated in four iterations for all cases. The time history of the states x and y and the control u, along with the grid point distribution for different values of $u_{\max}$ at the final iteration of MTOA, are shown in Fig. 4. The final states $x(2)$ and $y(2)$ (rounded off to five decimal places), the overall CPU time taken by MTOA to solve the problem, and the number of nodes used at the final iteration of MTOA (N_f) for the previous three values of $u_{\max}$ are summarized in Table 3. The values of $x(2)$ and $y(2)$ are the same as those reported in [50].

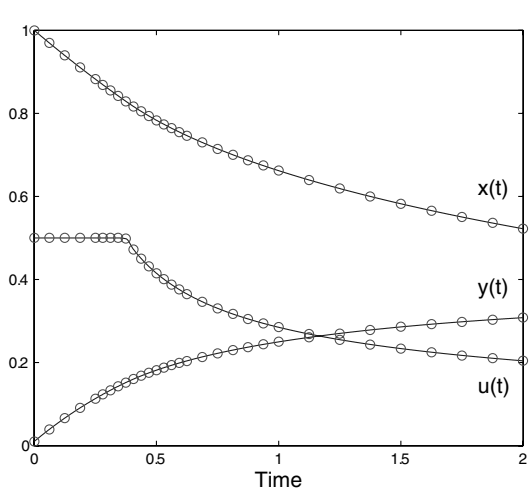
We also solved the same problem using the Hermite–Simpson discretization on a uniform grid with the same number of points as used by MTOA at the final iteration: that is, on a uniform mesh with N_4 nodes. The CPU times t_{CPU} used by the algorithm for all the cases are summarized in Table 4. The values for both $x(2)$ and $y(2)$ were accurate up to five decimal places for the case $u_{\max} = 0.4$. Because neither of the two values $x(2)$ or $y(2)$ was of the same accuracy for the remaining two cases, we resolved the problem for the cases $u_{\max} = 0.5$ and 0.2 with a larger number of nodes in the uniform mesh (again using a linear initial guess). We repeated this process until the values for both the states at the final time coincided to five decimal places to the solution given in Table 3. The result was a uniform mesh of 55 and 20 nodes for the cases $u_{\max} = 0.5$ and 0.3, respectively. These observations, along with the corresponding CPU times, are reported in Table 4. The uniform mesh implementation required more points to obtain the same accuracy. The corresponding CPU times were comparable for this example for both uniform and nonuniform mesh implementations. The reader should be reminded, however, that the

Table 1 Example 1: number of grid points along with the error in the computed optimal cost at each iteration

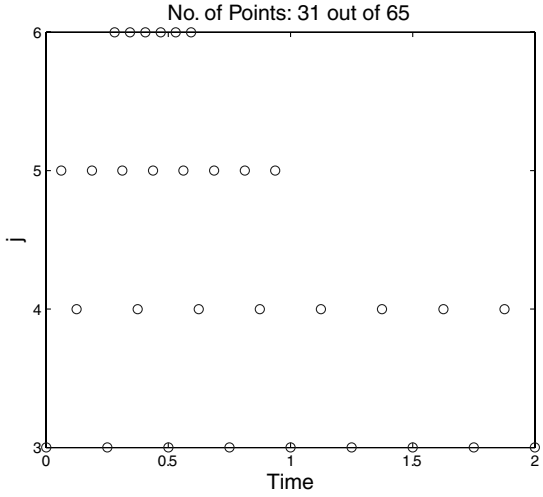
Iteration	N_{iter}	$\|x - x^*\|_{L^\infty}$	$\|v - v^*\|_{L^\infty}$	$\|u - u^*\|_{L^\infty}$	$\|J - J^*\|$
1	9	4.0×10^{-2}	1.5×10^{-1}	1.7×10^{0}	Failed
2	15	1.3×10^{-4}	2.1×10^{-3}	1.3×10^{-1}	5.7×10^{-3}
3	29	3.9×10^{-6}	5.9×10^{-5}	3.0×10^{-3}	4.6×10^{-4}
4	45	3.1×10^{-7}	1.4×10^{-5}	5.2×10^{-4}	6.6×10^{-6}
5	61	3.0×10^{-8}	1.6×10^{-6}	5.6×10^{-5}	3.3×10^{-8}

Table 2 Example 1: number of grid points and error for uniform mesh

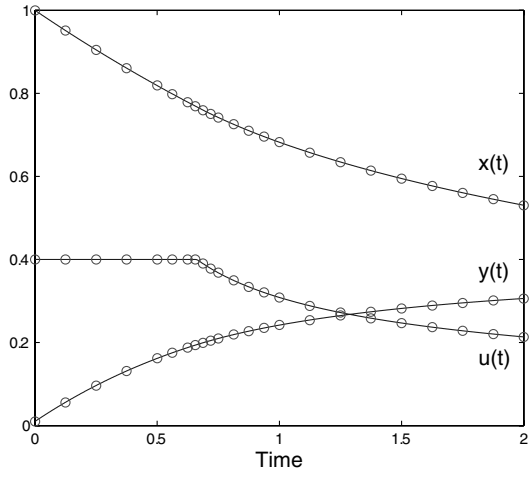
N_1	$\|x - x^*\|_{L^\infty}$	$\|v - v^*\|_{L^\infty}$	$\|u - u^*\|_{L^\infty}$	$\|J - J^*\|$
61	3.7×10^{-6}	1.4×10^{-4}	1.4×10^{-1}	2.7×10^{-4}
131	1.7×10^{-6}	9.5×10^{-5}	2.7×10^{-2}	6.0×10^{-5}



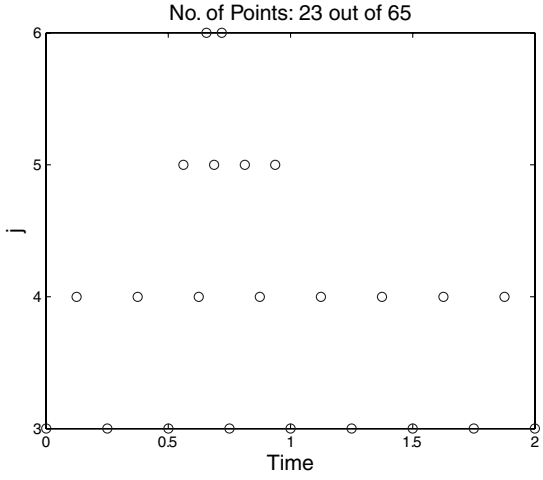
a) Time history of x, y, and u for $u_{\max} = 0.5$



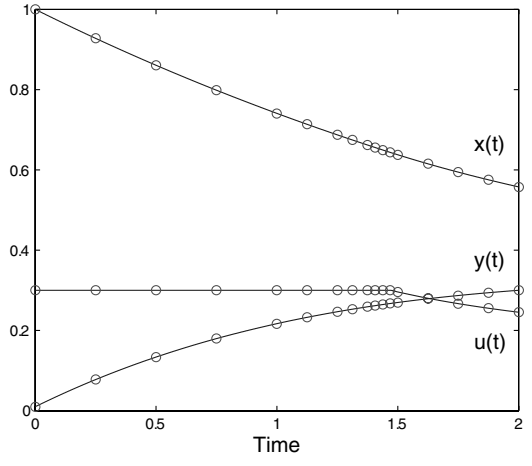
b) Grid point distribution



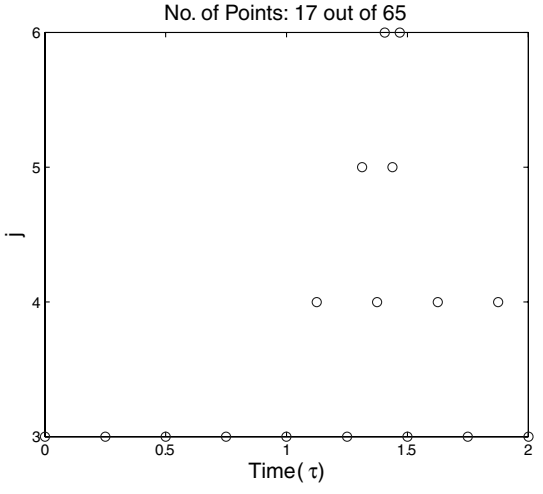
c) Time history of x, y, and u for $u_{\max} = 0.4$



d) Grid point distribution



e) Time history of x, y, and u for $u_{\max} = 0.3$



f) Grid point distribution

Fig. 4 Example 2: time history of states x and y and control u along with the grid point distributions for different $u_{\max}$ at the final iteration of MTOA.

uniform grid solutions were obtained by calling SNOPT only once (assuming convergence was possible). Hence *per iteration*, the CPU time for the MTOA is indeed smaller, as expected.

Example 3. In this example, we investigate the performance of MTOA to a hypersensitive problem, taken from [35]. As pointed out in [35,51], this problem is extremely difficult to solve using indirect methods. The problem is to minimize

$$J = \int_0^{10{,}000} (x^2(t) + u^2(t))\,\mathrm{d}t \tag{51}$$

subject to

$$\dot{x} = -x^3 + u \tag{52}$$

Table 3 Example 2: number of nodes used by MTOA at the final iteration, overall CPU time taken by MTOA, and final states for three different values of $u_{\max}$

$u_{\max}$	N_f	t_{CPU}, s	$x(2)$	$y(2)$
0.5	31	6.2	0.52222	0.30813
0.4	23	3.8	0.53051	0.30611
0.3	17	1.7	0.55765	0.30013

Table 4 Example 2: Uniform mesh

$u_{\max}$	N	t_{CPU}, s	Error	N	t_{CPU}, s	Error
0.5	31	3	10^{-5}	55	6.9	10^{-6}
0.4	23	1.7	10^{-6}	——	——	——
0.3	17	1.0	10^{-5}	20	1.3	10^{-6}

and

$$y(0) = 1, \qquad y(10,000) = 1.5 \tag{53}$$

We solved this problem on a grid with $J_{\min} = 4$ and $J_{\max} = 10$. The threshold used was $\epsilon = 10^{-4}$. MTOA terminated in 5 iterations and the overall CPU time taken by MTOA to solve this problem was 17.5 s. The final nonuniform grid (shown in Fig. 5b) included 53 nodes. The time history of the state x and the grid point distribution at the final iteration of MTOA are shown in Fig. 5.

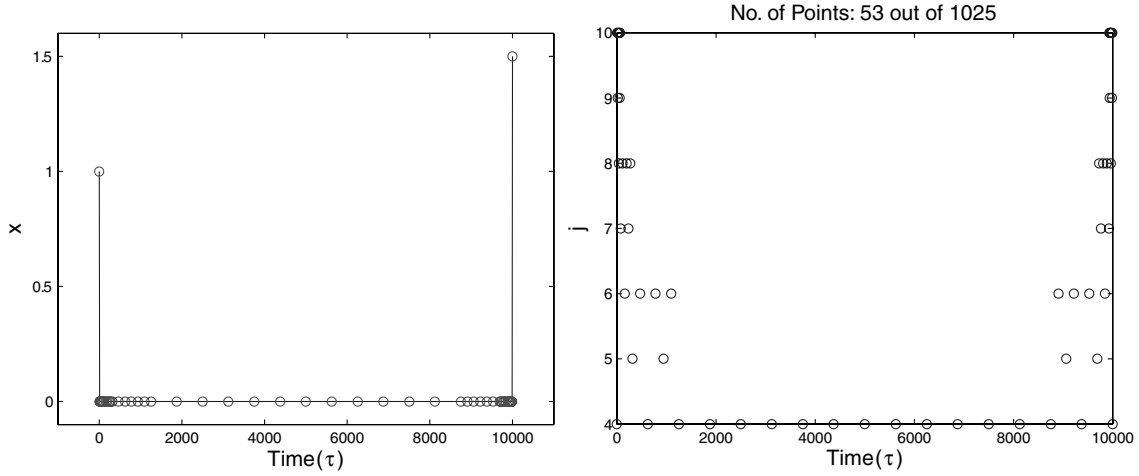
For comparison, we also solved the same problem using a Hermite–Simpson discretization on a uniform grid with the same number of nodes as used by MTOA at the final iteration: that is, on a uniform mesh with 53 nodes. The algorithm terminated after 43.7 s; the value of the optimal cost found was an order of magnitude larger than the optimal cost found using MTOA. These results again show the superiority of the MTOA over uniform grid implementations. For this example, the uniform grid implementation not only took more than twice the CPU time of MTOA, but also returned a solution that was far worse than that obtained from MTOA.

Example 4. As our final example, we consider the realistic problem of optimizing the reentry trajectory of an Apollo-type vehicle [9]. This is a benchmark problem in trajectory optimization that is known to be very challenging, owing to its sensitivity in terms of the initial guesses.

The equations of motion during the flight of the vehicle through the Earth's atmosphere are as follows:

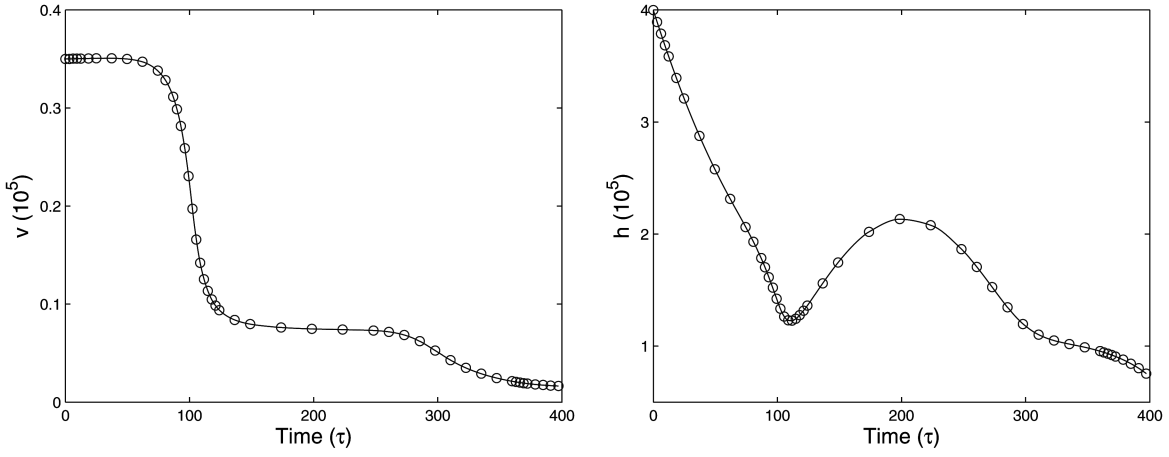
$$\dot{v} = -\frac{S}{2m}\rho v^2 c_D(u) - \frac{g\sin\gamma}{(1+\xi)^2},$$

$$\dot{\gamma} = \frac{S}{2m}\rho v c_L(u) + \frac{v\cos\gamma}{R(1+\xi)} - \frac{g\cos\gamma}{v(1+\xi)^2}, \quad \dot{\xi} = \frac{v}{R}\sin\gamma, \quad \dot{\zeta} = \frac{v}{1+\xi}\cos\gamma$$



a) Iteration 5: time history of x **b) Iteration 5: grid point distribution**

Fig. 5 Example 3: time history of state x along with the grid point distribution at the final iteration of MTOA.



a) Iteration 5: time history of v **b) Iteration 5: time history of h**

Fig. 6 Example 4, problem A: time histories of v and h for $u_{\max} = 180$ at the final iteration of MTOA.

where v is the velocity, γ is the flight-path angle, $\xi = h/R$ is the normalized altitude, h is the altitude above the Earth's surface, R is the Earth's radius, and ζ is the distance on the Earth's surface of a trajectory of an Apollo-type vehicle. The control variable is the angle of attack u. For the lift and drag, the following relations hold:

$$c_D = c_{D_0} + c_{DL}\cos u, \qquad c_{D_0} = 0.88, \qquad c_{DL} = 0.52 \tag{54}$$

$$c_L = c_{L_0}\sin u, \qquad c_{L_0} = -0.505 \tag{55}$$

The air density is assumed to satisfy the relationship $\rho = \rho_0 e^{-\beta R\xi}$. The values of the constants are

$$R = 209.0352(10^5 \text{ ft}), \qquad S/m = 50,000(10^{-5} \text{ ft}^2\,\text{slug}^{-1}),$$
$$\rho_0 = 2.3769 \times 10^{-3} \text{ (slug ft}^{-3}),$$
$$g = 3.2172 \times 10^{-4}(10^5 \text{ ft s}^{-2}), \qquad \beta = 1/0.235(10^{-5} \text{ ft}^{-1})$$

The cost functional to be minimized that describes the total stagnation point convective heating per unit area is given by the integral

$$J = \int_0^{t_f} 10v^3\sqrt{\rho}\,\mathrm{d}t \tag{56}$$

The vehicle is to be maneuvered into an initial position favorable for the final splashdown in the Pacific. Data at the moment of entry are

$$v(0) = 0.35(10^5 \text{ ft s}^{-1}), \qquad \gamma(0) = -5.75 \text{ deg} \tag{57}$$

$$\xi(0) = 4/R(h(0) = 400,000 \text{ ft}), \qquad \zeta(0) = 0(10^5 \text{ ft}) \tag{58}$$

Pesch [9] considered two situations for the given problem: one with constraints on control and the other with constraints on the state. We consider both of these problems in the sequel.

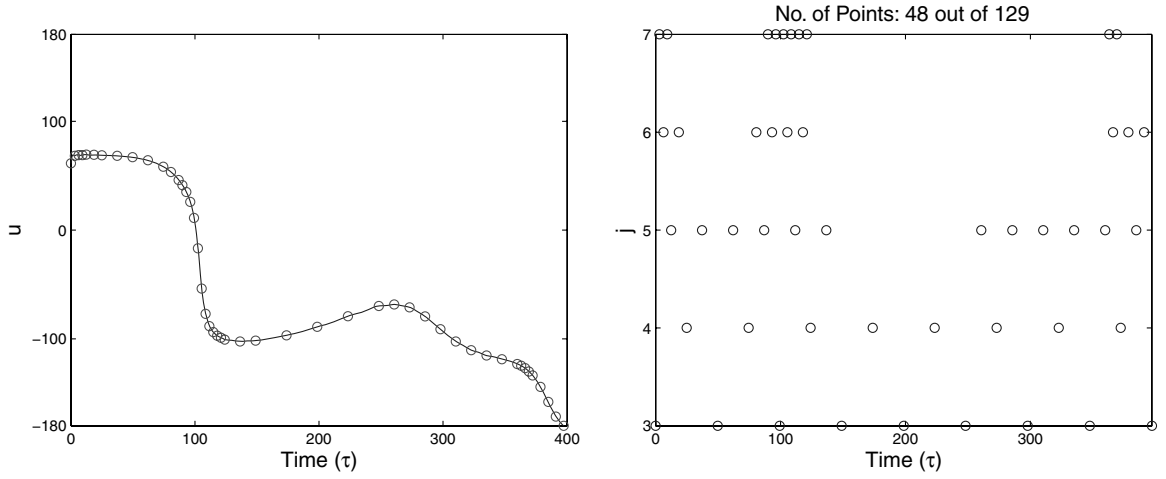
Problem A. Problem A imposes a control inequality constraint that limits the deceleration of the vehicle:

$$|u| \le u_{\max}, \qquad u_{\max} > 0 \tag{59}$$

The data prescribed at the unspecified terminal time t_f for problem A are as follows:

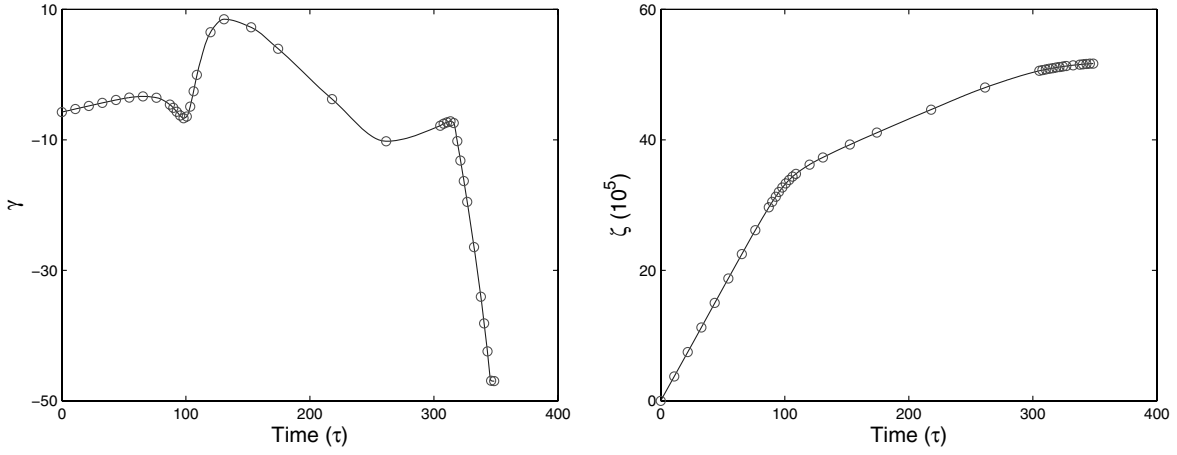
$$v(t_f) = 0.0165(10^5 \text{ ft s}^{-1}), \qquad \gamma(t_f)\text{unspecified} \tag{60}$$

$$\xi(t_f) = 0.75530/R(h(t_f) = 75,530 \text{ ft}),$$
$$\zeta(t_f) = 51.6912(10^5 \text{ ft}) \tag{61}$$



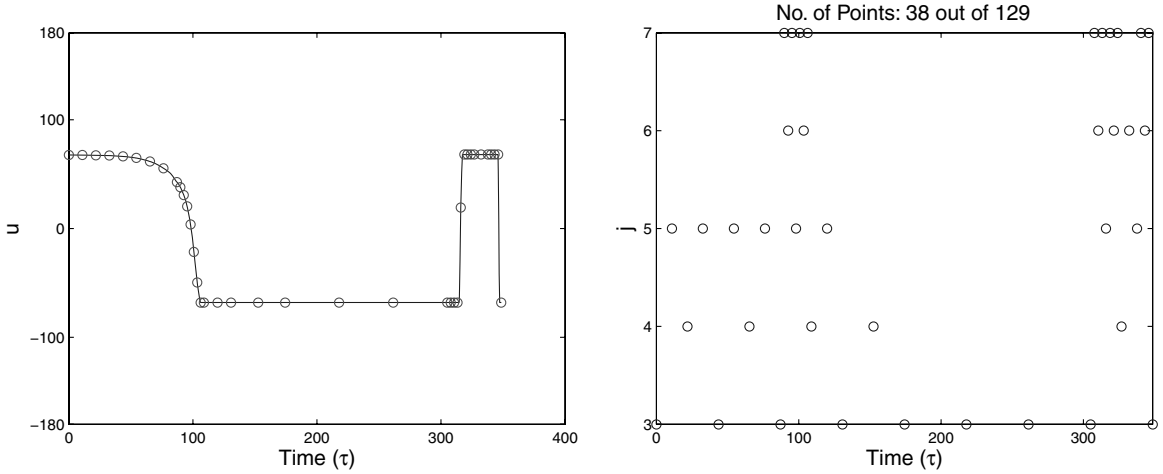
a) Iteration 5: time history of u **b) Iteration 5: grid point distribution**

Fig. 7 Example 4, problem A: time history of u along with the grid point distribution for $u_{\max} = 180$ at the final iteration of MTOA.



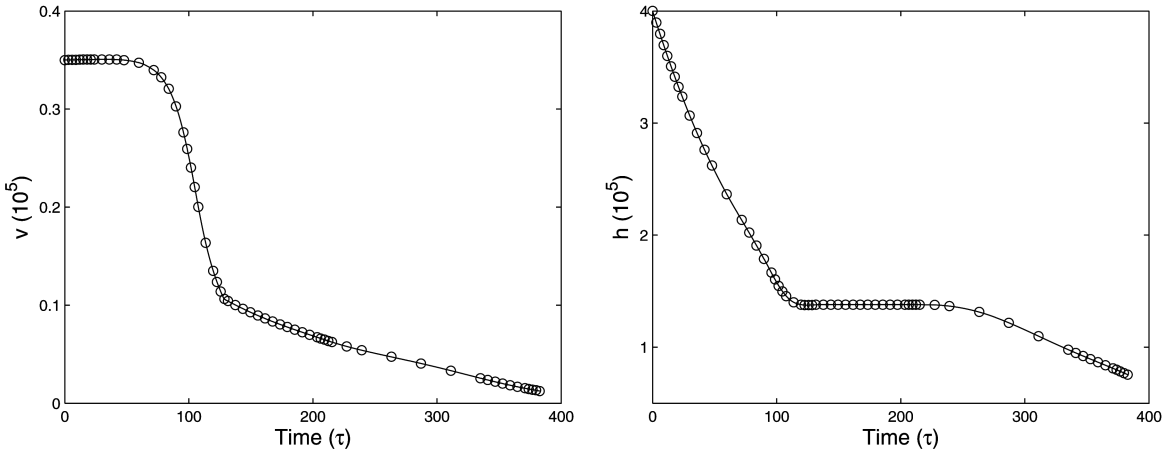
a) Iteration 5: time history of γ **b) Iteration 5: time history of ζ**

Fig. 8 Example 4, problem A: time histories of γ and ζ for $u_{\max} = 68$ at the final iteration of MTOA.



a) Iteration 5: time history of u b) Iteration 5: grid point distribution

Fig. 9 Example 4, problem A: time history of u along with the grid point distribution for $u_{\max} = 68$ at the final iteration of MTOA.



a) Iteration 5: time history of v b) Iteration 5: time history of h

Fig. 10 Example 4, problem B: time histories of v and h for $\xi_{\max} = 0.0066$ at the final iteration of MTOA.

We solved this problem for all the cases considered by Pesch [9], and the results obtained using MTOA vindicate the proposed algorithm. For the sake of brevity, we only give the results for the cases when $u_{\max} = 180$ and 68.

We solved this problem on a grid with $J_{\min} = 3$ and $J_{\max} = 7$. The threshold used for this problem was $\epsilon = 10^{-2}$. The algorithm for both cases terminated after 5 iterations and the overall CPU times taken by MTOA to solve the problem for both cases were 111.2 and 125.6 s, respectively. The time histories of the velocity v and altitude above the Earth's surface (h) at the final iteration of MTOA for $u_{\max} = 180$ are shown in Fig. 6. The time history of the angle of attack u along with the grid point distribution at the final iteration of MTOA for $u_{\max} = 180$ are shown in Fig. 7. The time histories of the flight-path angle γ and the distance on the Earth's surface (ζ) at the final iteration of MTOA for $u_{\max} = 68$ are shown in Fig. 8. The time history of the angle of the angle of attack (u) along with the grid point distribution at the final iteration of MTOA for $u_{\max} = 68$ are shown in Fig. 9.

Table 5 Example 4: Uniform mesh vs MTOA

Problem	Uniform mesh		MTOA	
	N	t_{CPU}, s	N_f	t_{CPU}, s
$A(u_{\max} = 180)$	65	241.1	39	76.6
$A(u_{\max} = 68)$	65	174.9	30	94.2
$B(\xi_{\max} = 0.0066)$	65	265.4	41	60.0

We also solved this problem for both the previous two cases on a grid with $J_{\min} = 3$ and $J_{\max} = 6$, but this time we uniformly refined the mesh after each iteration. The reason for choosing $J_{\max} = 6$ is because this problem could not be solved on a uniform grid finer than $\mathcal{V}_6$ because of hardware limitations. The CPU times taken by the algorithm for both the cases are shown in Table 5. We solved the problem using MTOA with the same parameters as before, but this time with $J_{\max} = 6$. MTOA terminated within four iterations for both cases. The overall CPU times taken by MTOA, along with the number of nodes used by MTOA at the final iteration N_f, are given in Table 5. As shown in Table 5, the MTOA outperformed the standard uniform grid implementation for this problem in terms of CPU time.

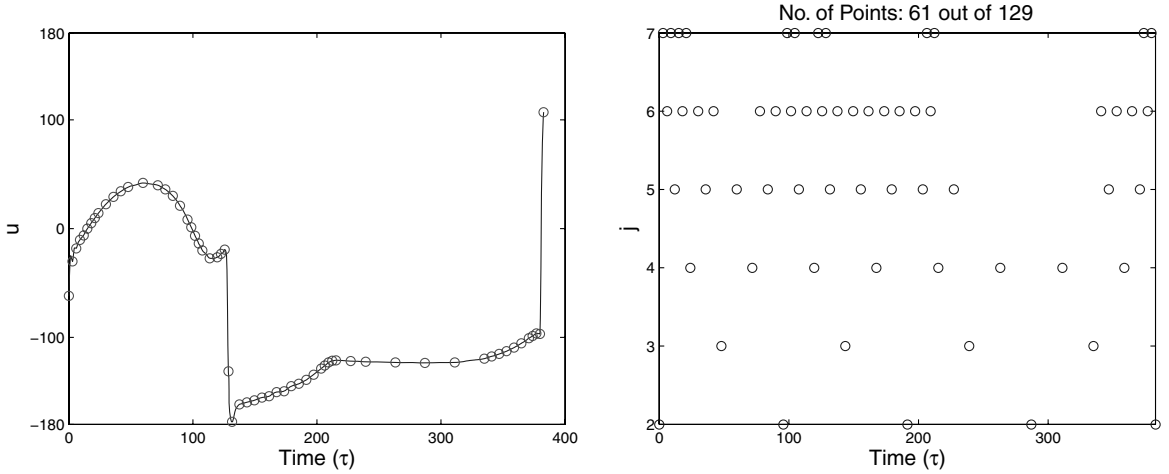
Problem B. For this problem, we impose a constraint to reduce reascent after the first dip into the atmosphere; that is, we have

$$\xi \le \xi_{\max}, \qquad \xi_{\max} > 0 \tag{62}$$

The data prescribed at the unspecified terminal time t_f for this problem are

$$v(t_f) = 0.01239929(10^5 \text{ ft s}^{-1}), \quad \gamma(t_f) = -26.237124 \text{ deg} \tag{63}$$

$$\begin{aligned} \xi(t_f) &= 0.75530/R(h(t_f) = 75,530 \text{ ft}), \\ \zeta(t_f) &= 51.10198(10^5 \text{ ft}) \end{aligned} \tag{64}$$



a) Iteration 5: time history of u **b) Iteration 5: grid point distribution**

Fig. 11 Example 4, problem B: time history of u for $\xi_{\max} = 0.0066$ along with the grid point distribution at the final iteration of MTOA.

Again, we solved this problem for all the cases considered by Pesch [9]. The results obtained using MTOA once again justify the proposed algorithm. For the sake of brevity, we only give the results for the case when $\xi_{\max} = 0.0066$. We solved this problem on a grid with $J_{\min} = 2$ and $J_{\max} = 7$. The threshold used for this problem was $\epsilon = 10^{-2}$. The algorithm terminated after 6 iterations and the overall CPU time taken by MTOA to solve this problem was 235.2 s. The time histories of the velocity v and altitude above the Earth's surface (h) at the final iteration of MTOA are shown in Fig. 10. The time history of the angle of attack u and the final grid point distribution are shown in Fig. 11.

When we attempted to solve the same problem on a uniform mesh with 33 nodes (with the same linear initial guess) using Hermite–Simpson discretization, the algorithm failed to converge. Increasing the number of nodes to 65 nodes and again using the same linear initial guess did not help. We therefore solved this problem again on a grid with $J_{\min} = 2$ and $J_{\max} = 6$, but this time we progressively refined the mesh *uniformly* after each iteration. The value of $J_{\max} = 6$ was chosen because the problem could not be solved on a uniform grid finer than $\mathcal{V}_6$, owing to hardware limitations. The CPU time taken by the algorithm in this case is given in Table 5. Once again, we solved the problem using MTOA with the same parameters as before, but this time with $J_{\max} = 6$. MTOA terminated in 5 iterations. The overall CPU times taken by MTOA, along with the number of nodes used by MTOA at the final iteration N_f, are also given in Table 5. These results again show the benefits of the MTOA in terms of accuracy and speed when compared with uniform grid implementations for this problem.

VII. Conclusions

In this paper, we proposed a novel multi-resolution-based approach for direct trajectory optimization. The algorithm automatically, and with minimal effort, generates a nonuniform grid that reduces the discretization error with each iteration. As a result, one is able to capture the solution accurately and efficiently using a relatively small number of points. All the transition points in the solution (for example, bang–bang subarcs or entry and exit points associated with state or mixed constraints) are captured with high accuracy. The convergence of the algorithm can be enhanced by initializing the algorithm on a coarse grid with a small number of variables. Once a converged solution is attained, the grid can be further refined by increasing the accuracy locally, only at the vicinity of those points that cannot be accurately interpolated by neighboring points in the grid. The methodology thus provides a compromise between robustness with respect to initial guesses, intermediate and final solution accuracy, and execution speed. These observations are supported by several numerical examples of challenging trajectory optimization problems.

A preliminary error analysis shows that the effect of the proposed multiresolution scheme is somewhat akin to a local control of the tolerance of the Runge–Kutta integration error. The error analysis also provides guidelines on how certain parameters needed in the algorithm (e.g., the order of the interpolating polynomials, the maximum/minimum time steps, etc.) can be chosen during implementation and to yield consistent approximations. Future work will focus on more quantitative measures for the selection of these parameters as well as on providing explicit error bounds for both the unconstraint case and for more general cases that include path constraints.

Acknowledgments

Partial support for this work was provided by National Science Foundation award no. CMS 0510259 and NASA award no. NNX08AB94A. The authors acknowledge useful discussions with Hao-Min Zhou from the Mathematics Department at Georgia Institute of Technology.

References

[1] Brauer, G. L., Cornick, D. E., and Stevenson, R., "Capabilities and Applications of the Program to Optimize Simulated Trajectories (POST)," NASA CR-2770, Feb. 1977.

[2] Meder, D. S., and McLaughlin, J. R., "A Generalized Trajectory Simulation System," *Summer Computer Simulation Conference*, edited by J. B. Mankin, R. H. Gardner, and H. H. Shugart, AFIPS Press, Montvale, NJ, July 1976, pp. 366–372.

[3] Hargraves, C. R., and Paris, S. W., "Direct Trajectory Optimization Using Nonlinear Programming and Collocation," *Journal of Guidance, Control, and Dynamics*, Vol. 10, No. 4, 1987, pp. 338–342.

[4] Fahroo, F., and Ross, I., "Trajectory Optimization by Indirect Spectral Collocation Methods," *AIAA/AAS Astrodynamics Specialist Conference*, AIAA, Reston, VA, Aug. 2000, pp. 123–129.

[5] Oberle, H. J., and Grimm, W., "BNDSCO-A Program for the Numerical Solution of Optimal Control Problems," Inst. for Flight System Dynamics, DLR, German Aerospace Research Center, TR DLR IB/515-89/22, Oberpfaffenhofen, Germany, 1989.

[6] Bulrish, R., and Kraft, D., "Computational Optimal Control," *International Series of Numerical Mathematics*, Vol. 115, Birkhäser Verlag, Basel, Switzerland, 1994.

[7] Betts, J. T., Biehn, N., Campbell, S. L., and Huffman, W. P., "Compensating for Order Variation in Mesh Refinement for Direct Transcription Methods," *Journal of Computational and Applied Mathematics*, Vol. 125, Nos. 1–2, Dec. 2000, pp. 147–158. doi:10.1016/S0377-0427(00)00465-9

[8] Binder, T., Blank, L., Dahmen, W., and Marquardt, W., "Grid Refinement in Multiscale Dynamic Optimization," RWTH Aachen, Tech. Rep. LPT-2000-11, Aachen, Germany, 2000.

[9] Pesch, H. J., "Real-Time Computation of Feedback Controls for Constrained Optimal Control Problems, Part 2: A Correction Method

Based on Multiple Shooting," *Optimal Control Applications and Methods*, Vol. 10, 1989, pp. 147–171. doi:10.1002/oca.4660100206

[10] Stryk, O. v., and Bulirsch, R., "Direct and Indirect Methods for Trajectory Optimization," *Annals of Operations Research*, Vol. 37, No. 1, Dec.1992, pp. 357–373. doi:10.1007/BF02071065

[11] Betts, J. T., "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, 1998, pp. 193–207.

[12] Polak, E., "An Historical Survey of Computational Methods in Optimal Control," *SIAM Review*, Vol. 15, No. 2, 1973, pp. 553–584. doi:10.1137/1015071

[13] Shen, H., and Tsiotras, P., "Time-Optimal Control of Axi-symmetric Spacecraft," *Journal of Guidance, Control, and Dynamics*, Vol. 22, No. 5, 1999, pp. 682–694.

[14] Bock, H. G., and Plitt, K. J., "A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems," *Proceedings of the Ninth Triennial World Congress of IFAC*, Pergamon, New York, 1984, pp. 242–247.

[15] Betts, J. T., and Huffman, W. P., "Mesh Refinement in Direct Transcription Methods for Optimal Control," *Optimal Control Applications & Methods*, Vol. 19, No. 1, 1998, pp. 1–21. doi:10.1002/(SICI)1099-1514(199801/02)19:1<1::AID-OCA616>3.0.CO;2-Q

[16] 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.

[17] Herman, A. L., and Conway, B. A., "Direct Optimization using Collocation Based on High-Order Gauss-Lobatto Quadrature Rules," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 3, 1996, pp. 592–599.

[18] Russell, R. D., and Shampine, L. F., "A Collocation Method for Boundary Value Problems," *Numerische Mathematik*, Vol. 19, No. 1, 1972, pp. 1–28. doi:10.1007/BF01395926

[19] Weiss, R., "The Application of Implicit Runge–Kutta and Collocation Methods to Boundary Value Problems," *Mathematics of Computation*, Vol. 28, No. 126, 1974, pp. 449–464. doi:10.2307/2005918

[20] Elnagar, G., Kazemi, M. A., 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. doi:10.1109/9.467672

[21] Fahroo, F., and Ross, I., "A Spectral Patching Method for Direct Trajectory Optimization," *Journal of the Astronautical Sciences*, Vol. 48, Nos. 2–3, 2000, pp. 269–286.

[22] Gong, Q., and Ross, I. M., "Autonomous Pseudospectral Knotting Methods for Space Mission Optimization," 16th AAS/AIAA Space Flight Mechanics Meeting, American Astronautical Society Paper 06-151, 2006.

[23] Enright, P. J., and Conway, B. A., "Discrete Approximations to Optimal Trajectories Using Direct Transcription and Nonlinear Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 4, 1992, pp. 994–1002.

[24] Ross, I. M., Fahroo, F., and Strizzi, J., "Adaptive Grids for Trajectory Optimization by Pseudospectral Methods," *AAS/AIAA Spaceflight Mechanics Conference*, AIAA, Reston, VA, 2003, pp. 649–668.

[25] Ross, I. M., and Fahroo, F., "Pseudospectral Knotting Methods for Solving Nonsmooth Optimal Control Problems," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 3, 2004, pp. 397–405. doi:10.2514/1.3426

[26] Binder, T., Cruse, A., Villar, C. A. C., and Marquardt, W., "Dynamic Optimization Using a Wavelet Based Adaptive Control Vector Parametrization Strategy," *Computers and Chemical Engineering*, Vol. 24, July 2000, pp. 1201–1207. doi:10.1016/S0098-1354(00)00357-4

[27] Schlegel, M., Stockmann, K., Binder, T., and Marquardt, W., "Dynamic Optimization using Adaptive Control Vector Parameterization," *Computers and Chemical Engineering*, Vol. 29, No. 8, 2005, pp. 1731–1751. doi:10.1016/j.compchemeng.2005.02.036

[28] Schwartz, A. L., "Theory and Implementation of Numerical Methods Based on Runge–Kutta Integration for Solving Optimal Control Problems," Ph.D. Thesis, Electrical Engineering and Computer Science, Univ. of California, Berkeley, Berkeley, CA, 1996.

[29] Jain, S., Tsiotras, P., and Zhou, H.-M., "Adaptive Multiresolution Mesh Refinement for the Solution of Evolution PDEs," *Proceedings of the 46th IEEE Conference on Decision and Control*, Inst. of Electrical and Electronics Engineers, Piscataway, NJ, 2007, pp. 3525–3530.

[30] Jain, S., Tsiotras, P., and Zhou, H.-M., "A Hierarchical Multiresolution Adaptive Mesh Refinement for the Solution of Evolution PDEs," *SIAM Journal on Scientific Computing* (submitted for publication).

[31] Harten, A., "Multiresolution Representation of Data: A General Framework," *SIAM Journal on Numerical Analysis*, Vol. 33, No. 3, 1996, pp. 1205–1256. doi:10.1137/0733060

[32] Deslauriers, G., and Dubuc, S., "Symmetric Iterative Interpolation Processes," *Constructive Approximation*, Vol. 5, No. 1, 1989, pp. 49–68. doi:10.1007/BF01889598

[33] Donoho, D. L., "Interpolating Wavelet Transforms," Dept. of Statistics, Stanford Univ., Stanford, CA, 1992.

[34] Cohen, A., "Numerical Analysis of Wavelet Methods," *Studies in Mathematics and its Applications*, Vol. 32, edited by D. N. Arnold, P. G. Ciarlet, P. L. Lions, and H. A. van der Vorst, Elsevier Science, Amsterdam, 2003.

[35] Betts, J. T., *Practical Methods for Optimal Control Using Nonlinear Programming*, Society for Industrial and Applied Mathematics, Philadelphia, 2001.

[36] Mallat, S. G., "A Theory for Multiresolution Signal Decomposition: the Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 7, 1989, pp. 674–693. doi:10.1109/34.192463

[37] Harten, A., "Adaptive Multiresolution Schemes for Shock Computations," *Journal of Computational Physics*, Vol. 115, No. 2, 1994, pp. 319–338. doi:10.1006/jcph.1994.1199

[38] Butcher, J. C., "Implicit Runge–Kutta Processes," *Mathematics of Computation*, Vol. 18, No. 85, 1964, pp. 50–64. doi:10.2307/2003405

[39] Hager, W. W., "Runge–Kutta Methods in Optimal Control and the Transformed Adjoint System," *Numerische Mathematik*, Vol. 87, No. 2, 2000, pp. 247–282. doi:10.1007/s002110000178

[40] Dontchev, A. L., Hager, W. W., and Veliov, V. M., "Second-Order Runge–Kutta Approximations in Control Constrained Optimal Control," *SIAM Journal on Numerical Analysis*, Vol. 38, No. 1, 2000, pp. 202–226. doi:10.1137/S0036142999351765

[41] Dontchev, A. L., and Hager, W. W., "The Euler Approximation in State Constrained Optimal Control," *Mathematics of Computation*, Vol. 70, No. 233, 2001, pp. 173–203. doi:10.1090/S0025-5718-00-01184-4

[42] Betts, J. T., Biehn, N., and Campbell, S. L., "Convergence of Nonconvergent IRK Discretizations of Optimal Control Problems with State Inequality Constraints," *SIAM Journal on Scientific Computing*, Vol. 23, No. 6, 2002, pp. 1981–2007. doi:10.1137/S1064827500383044

[43] Hairer, E., Norsett, S. P., and Wanner, G., *Solving Ordinary Differential Equations 1: Nonstiff Problems*, Springer–Verlag, New York, 1987.

[44] Hairer, E., Norsett, S. P., and Wanner, G., *Solving Ordinary Differential Equations 2: Stiff and Differential-Algebraic Problems*, Springer–Verlag, New York, 1991.

[45] Drikakis, D., and Rider, W., *High Resolution Methods for Incompressible and Low-Speed Flows*, Springer, New York, 2004.

[46] De Boor, C., "A Practical Guide to Splines," *Applied Mathematical Sciences*, Vol. 27, Springer–Verlag, New York, 1978.

[47] Sendov, B., and Popov, V. A., *The Averaged Moduli of Smoothness, Pure and Applied Mathematics*, Wiley, Chichester, England, U.K., 1988.

[48] Gill, P. E., Murray, W., and Saunders, M. A., *User's Guide for SNOPT Version 6: A Fortran Package for Large-Scale Nonlinear Programming*, Systems Optimization Lab., Stanford Univ., Stanford, CA, 2002.

[49] Bryson, A. E., and Ho, Y.-C., *Applied Optimal Control: Optimization, Estimation, and Control*, Hemisphere, Washington, D.C., 1975.

[50] Citron, S. J., *Elements of Optimal Control*, Holt, Rinehart, and Winston, New York, 1969.

[51] 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. 20, No. 2, 1999, pp. 59–77. doi:10.1002/(SICI)1099-1514(199903/04)20:2<59::AID-OCA646>3.0.CO;2-8