JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 33, No. 3, May–June 2010

Enhanced Collocation Method for Dynamical Systems Subject to Finite Set Control

Stuart A. Stanton* and Belinda G. Marchand†
University of Texas at Austin, Austin, Texas 78712-0235

DOI: 10.2514/1.41734

An enhanced optimization method, rooted in direct collocation, is formulated to treat a specific class of continuous-time hybrid control systems; specifically, dynamical systems that depend on continuous state variables and spatially discrete control variables. In this study, a spatially discrete control variable refers to one that is defined over the continuous-time domain yet is constrained to a range of discrete finite values. This subset of hybrid control systems is termed here as finite set control. Traditionally, optimal control for hybrid systems of this class is formulated in the context of mixed-integer nonlinear programming. The numerical efficiency of these methods, however, is adversely affected by the number of independent control variables and the range of the combined set. The methodology presented in this study focuses on this class of problems: multiple independent control variables, each with a unique range and a switching schedule independent from the others. The resulting formulation leads to a more computationally efficient nonlinear program suitable for the identification of optimal control solutions for this class of hybrid systems.

I. Introduction

UNIQUE optimal control applications exist, requiring control solutions that are defined over a continuous-time domain but are constrained to a finite range of values in that domain. This study is focused specifically on systems of this form for which the states are governed by differential equations of motion, and control variables exhibit discontinuities in switching from one operating point to another in the range or set of possible values. Dynamical systems in which the bodies are controlled by on–off actuation schemes fit well into this category. Additionally, continuous control problems in which bang–bang control behavior is assumed a priori may also be treated in this context.

In this framework, the system can be described by a vector of states $\boldsymbol{y} \in \mathbb{R}^{n_y}$ governed by

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(t,\ \boldsymbol{y},\ \boldsymbol{u}) \tag{1}$$

The nonlinear field in Eq. (1) depends on the state $\boldsymbol{y}$ and a control vector $\boldsymbol{u}$, consisting of n_u elements, and may also depend explicitly on time t. In this study, the range of each control variable is limited to a finite set $\mathbb{U}_i$. Therefore, if u_i denotes the ith control variable, for $i = 1,\ \ldots,\ n_u$, then $u_i \in \mathbb{U}_i$ where

$$\mathbb{U}_i = \{\tilde{u}_{i,1},\ \ldots,\ \tilde{u}_{i,m_i}\} \tag{2}$$

That is, each control u_i can take on any one of the m_i possible values in the set $\mathbb{U}_i$. Notice that, in general, the number of elements in each of these n_u finite sets may vary from one control variable to the next. Thus, the range (i.e., the possible values of the elements) and cardinality (i.e., the number of elements in the set) of each finite set are unique for a given control variable. In addition, the total number of control combinations is

$$\bar{m} = \prod_{i=1}^{n_u} m_i \tag{3}$$

Thus, any control problem is simply the determination of which of the $\bar{m}$ control combinations to implement at each instant in time.

Systems described by Eqs. (1) and (2) fall into a class of hybrid control systems [1] due to the presence of both continuous and spatially discrete variables. Although the scope of hybrid control systems is broad, potentially involving continuous and discrete variables in both states and controls, many treatments on the subject are limited to special cases or specific classes. Often, researchers have considered the case of a single switching variable [2–4] for which the discrete variable indicates the system mode or dynamics vector field governing the system at a given time. For example, using different Lyapunov-like functions for each mode, conditions for stability can be obtained, leading to switching strategies based on the Lyapunov-like functions and/or their time derivatives.

This investigation considers an optimal control problem involving the system described by Eq. (1):

$$J = \phi(t_0,\ \boldsymbol{y}_0,\ t_f,\ \boldsymbol{y}_f) + \int_{t_0}^{t_f} L(t,\ \boldsymbol{y},\ \boldsymbol{u})\,\mathrm{d}t \tag{4}$$

with equality constraints formulated as

$$\boldsymbol{0} = \boldsymbol{\psi}_0(t_0,\ \boldsymbol{y}_0), \qquad \boldsymbol{0} = \boldsymbol{\psi}_f(t_f,\ \boldsymbol{y}_f), \qquad \boldsymbol{0} = \boldsymbol{\beta}(t,\ \boldsymbol{y},\ \boldsymbol{u}) \tag{5}$$

Here, $\boldsymbol{\psi}_0 \in \mathbb{R}^{n_{\psi_0}}$ is a vector of initial conditions, $\boldsymbol{\psi}_f \in \mathbb{R}^{n_{\psi_f}}$ denotes the terminal constraints, and $\boldsymbol{\beta} \in \mathbb{R}^{n_\beta}$ represents the path constraints imposed over all $t \in [\,t_0 \quad t_f\,]$. Note that though all the constraints presented here are formulated as equality constraints, that is not a necessary restriction.

The resulting hybrid optimal control problem is often termed in the literature as a mixed-integer optimal control problem [5–8]. Although the feasible control values in $\mathbb{U}_i$ are not necessarily integers, it is generally possible to formulate them as a function of integers (most often as the binary values 0 and 1). This leads to the use of mixed-integer nonlinear programming (MINLP) as a tool for solving these problems. Commonly used algorithms include the branch and bound method [9], the generalized benders decomposition method [10], and the outer approximation method [11,12]. In a basic sense, each of these methods fixes all or some of the integer

Received 21 October 2008; revision received 8 May 2009; accepted for publication 28 May 2009.

*Capt., U.S. Air Force; Ph.D. Candidate, Department of Aerospace Engineering and Engineering Mechanics, 1 University Station, Mail Stop C0600. Student Member AIAA.

†Assistant Professor, Department of Aerospace Engineering and Engineering Mechanics, 1 University Station, Mail Stop C0600. Member AIAA.

variables on each iteration and solves a series of primal subproblems using standard nonlinear programming (NLP) algorithms. Although these methods can be effective, their primary limitation is the adverse impact of increasing the number of integer variables, which subsequently increases the size of the combinatorial problem exponentially.

An overview of the existing research efforts on mixed-integer optimal control problems is provided by Sager et al. [5], discussing the indirect and direct transcription methods in conjunction with MINLP methods. Additionally, Allgor and Barton [13] address several MINLP approaches to decompose the master problem such that either collocation or direct single shooting strategies can be used on the variational subproblems. Stryk and Glocker [6] use a branch and bound procedure in combination with direct collocation for solving robotics problems. Gerdts [7] also uses the branch and bound method on an MINLP, resulting from the discretization of an optimal control problem. Lee et al. [14] use a control parameterization enhancing technique to transform the discrete-valued problem into an ordinary optimal control problem with fixed switching points.

The objective of this investigation is to effectively treat the hybrid optimal control problem using a strictly NLP approach. Wei et al. [15] use a traditional NLP technique on the hybrid system with a single discrete control variable. Gerdts [8] proposes a variable time transformation method using direct shooting to formulate the problem as a nonlinear program. Comparing identical problems using the branch and bound approach, he demonstrates orders of magnitude improvement in computational time, using the strictly NLP approach. A similar method, optimizing switching times (proposed by Sager et al. [5]), has the same effect of reducing the problem to a nonlinear program. However, one drawback to these methods is that the number of stages or segments grows exponentially with the number of control functions and expected switches.

In contrast to classical transcription formulations, the present development features a unique approach that effectively addresses the deficiencies of the methods previously described in dealing with multiple independent spatially discrete control variables. An immediate advantage of the proposed approach is a significant reduction in computational overhead during the numerical solution process. The improved performance allows for a more effective treatment of this class of problems, particularly when the cardinality of each of the associated finite sets (i.e., the discrete range of each control variable) is large.

Solutions derived using the proposed enhanced direct collocation method are characterized as optimal switching schedules. The optimal control history is subsequently deduced directly from these switching schedules and the corresponding values of the control variables in each finite set. The process of transcribing the optimal control problem into a parameter optimization problem is consistent with classical methods [16–18]. The solution is numerically determined using an existing NLP algorithm, such as SNOPT [19]. A unique and complex element of this type of problem, from a numerical perspective, is that switches occur not only along one control axis but also across control variables. This leads to switches in variable dependencies during the solution process. These switching dependencies are explored and addressed in this investigation.

II. Transcription Formulations for the Finite Set Optimal Control Problem

A brief review of traditional collocation for solving optimal control problems, in the presence of continuous control inputs, is presented first to establish the terminology relevant to this study. Subsequently, two concepts for employing a collocation approach on a finite set control problem are demonstrated. The first represents a logical step from the traditional collocation formulation and is addressed by various authors in the literature. The second is an alternative formulation proposed here to reduce the size of the resulting NLP problem when multiple independent finite set control variables are present.

A. Traditional Collocation Formulation

The optimal control problem established in Eqs. (1–5) is transcribed using collocation [16–18] into a set of n optimization parameters $\boldsymbol{x}$, an objective function $F(\boldsymbol{x})$, and a series of constraints $\boldsymbol{c}(\boldsymbol{x}) = \boldsymbol{0}$. At a minimum, the vector constraint equation accounts for the initial, final, path, and dynamical constraints and may be decomposed as

$$\boldsymbol{c}(\boldsymbol{x}) = [\, \boldsymbol{c}^T_{\psi_0}(\boldsymbol{x}) \quad \boldsymbol{c}^T_{\psi_f}(\boldsymbol{x}) \quad \boldsymbol{c}^T_{\beta}(\boldsymbol{x}) \quad \boldsymbol{c}^T_{\dot{y}}(\boldsymbol{x}) \,]^T = \boldsymbol{0} \tag{6}$$

In this study, a suitable collocation formulation must account for the presence of discontinuities, either in states or controls. The term knot is used here to denote a point in time at which a discontinuity might occur, and it divides the trajectory into separate subarcs or segments. The knots may be either fixed or free in time. When the knots are free, the times at which they occur are optimized. Let there be n_k knots and n_s segments. Along each segment, the problem is discretized in time at n_n different places called nodes. Figure 1 illustrates the conceptual relation between knots, nodes, and segments at two consecutive iterations. Each segment starts at either t_0 or a knot and ends at either another knot or t_f. Thus, the formulation consists of $n_k = n_s - 1$ knots.

Although, conceptually, each segment may consist of a different number of nodes, the transcription is simplified by assuming an equal number of nodes per segment. In this formulation, the nodes are uniformly spaced within a segment, but they are not necessarily uniformly spaced along the course of the trajectory. That is, each segment may span a different length of time. In most cases, it is desirable for the segment duration (or the location of the knots) to be optimized as well. In Fig. 1, the knot times change between iteration p and iteration $p + 1$ as the NLP algorithm improves the current $\boldsymbol{x}$. Consequently, assigned times for nodes change to maintain uniform spacing per segment.

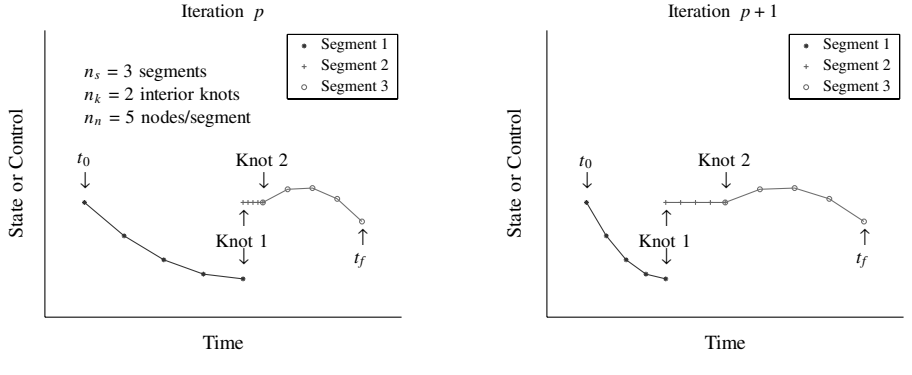

Fig. 1 An example of segments and knots at two consecutive iterations.

A convenient parameterization includes (in the parameter vector) the states and controls at the nodes, the duration of the segments, and the initial and final times. Consider the parameter vector

$$\boldsymbol{x} = [\underbrace{\cdots\ y_{i,j,k}\ \cdots}_{n_y n_n n_s}\ \underbrace{\cdots\ u_{i,j,k}\ \cdots}_{n_u n_n n_s}\ \underbrace{\cdots\ \Delta t_k\ \cdots}_{n_s}\ \underbrace{t_0\ t_f}_{2}\ ]^T \tag{7}$$

In the preceding equation, i identifies members of a vector, j denotes the respective nodes, and k represents the relevant knot or segment. The knot times are defined as

$$t_k = t_0 + \sum_{\kappa=1}^{k} |\Delta t_\kappa|, \qquad k = 1,\ \ldots,\ n_k \tag{8}$$

and the resulting node times are

$$t_{j,k} = t_{k-1} + \frac{j-1}{n_n - 1}(t_k - t_{k-1})$$
$$j = 1,\ \ldots,\ n_n, \qquad k = 1,\ \ldots,\ n_s \tag{9}$$

where $t_{n_s} = t_f$.

A summary of the quantities available from this parameterization are presented in Table 1. The arrays $\boldsymbol{Y}_{n_y,n_n,n_s}$ and $\boldsymbol{U}_{n_u,n_n,n_s}$ contain the states and controls (respectively) at each node. Node times, knot times, and segment durations are contained in $\boldsymbol{T}_{n_n,n_s}$, $\boldsymbol{T}_{n_k}$, and $\Delta\boldsymbol{T}_{n_s}$. Thus, the parameter vector provides sufficient information to adequately represent the optimal control problem as a nonlinear program through appropriate definitions of $F(\boldsymbol{x})$ and $\boldsymbol{c}(\boldsymbol{x})$.

B. One Concept for Optimizing Switching Times

When the control variables are limited to a finite set, the traditional collocation formulation must be modified. Assume that the ith control variable has a discrete range characterized by the finite set $\mathbb{U}_i = \{\tilde{u}_{i,1},\ \ldots,\ \tilde{u}_{i,m}\}$. Note that, in this case, the range of each control variable is limited to m different values. Although it is generally possible for the cardinality of each finite set $\mathbb{U}_i$ to be different, the cardinality of all n_u sets in this example is assumed equinumerous. That is, all finite sets have the same number of elements. This constraint can be expressed as

$$\prod_{\mu=1}^{m}(u_i - \tilde{u}_{i,\mu}) = 0, \qquad i = 1,\ \ldots,\ n_u \tag{10}$$

Then, at each instant in time, the optimal solution indicates which of the m possible values is optimal for each variable.

A direct consequence of this formulation is that the NLP algorithm must determine a control solution over the $\bar{m} = m^{n_u}$ possible control combinations at each instant in time. However, this type of optimization process is rooted in gradient methods, and gradients are not defined across a switch or discontinuity in a variable. Thus, during the solution process, spatially discrete control variables are essentially treated as continuous variables subject to constraints, and the final solution must meet the constraint in Eq. (10). Unfortunately, gradient methods move a point toward a root of the constraint according to the gradient at the current point of $\boldsymbol{x}$, independent of whether it is the right root to choose. Once the constraint is met, it is difficult (if not impossible) for the NLP algorithm to seek another root. Although the constraint is valid, it cannot be implemented with traditional transcription.

Table 1 Nomenclature summary: multiple segment formulation

Array	Description	Dimension	Element
$\boldsymbol{Y}_{n_y,n_n,n_s}$	States by node	$n_y \times n_n \times n_s$	$\boldsymbol{y}_{j,k}$ or $y_{i,j,k}$
$\boldsymbol{U}_{n_u,n_n,n_s}$	Controls by node	$n_u \times n_n \times n_s$	$\boldsymbol{u}_{j,k}$ or $u_{i,j,k}$
$\boldsymbol{T}_{n_n,n_s}$	Node times	$n_n \times n_s$	$t_{j,k}$
$\boldsymbol{T}_{n_k}$	Knot times	n_k	t_k
$\Delta\boldsymbol{T}_{n_s}$	Segment durations	n_s	Δt_k
n_s	Number of segments	$n_k + 1$	

To circumvent this problem, consider the multiple segment formulation of Sec. II.A. Instead of optimizing the control values, assume the control values and optimize the switching times. Similar to the direct shooting implementations presented by Gerdts [8] and Sager et al. [5], a formulation can be devised that assigns each segment one of the $\bar{m}$ options and then requires the NLP algorithm to determine the duration of each segment. With this formulation, the predetermined control values are removed from the parameter vector. This is analogous to having one discrete control variable for which the $\bar{m}$ values represent each of the possible control modes available to the system.

For example, a segment k is designated with control $\boldsymbol{u}(t_{j,k}) = [\tilde{u}_{1,1},\ \ldots,\ \tilde{u}_{n_u,1}]^T$ for all nodes $j = 1,\ \ldots,\ n_n$ along that segment. If this control combination is not desirable in the optimal solution, the NLP algorithm determines that the duration of that segment Δt_k is zero. A user must ensure that each control combination is represented in at least one segment (possibly many). Assuming the existence of nonoptimal control combinations, it is conceivable that the durations of a significant number of segments are reduced to zero.

Let $n_s = \alpha\bar{m}$ segments, where α is an arbitrary integer assigned by the user. Choosing a formulation with enough segments to cycle through each of the $\bar{m}$ control combinations multiple times increases the number of control sequences in the solution space for the problem. Naturally, a user must balance solution flexibility with the size of the underlying NLP problem. Without knowing in advance what the optimal sequence may look like, a user should set α as high as possible. The resulting NLP problem is naturally large, and parameters in $\boldsymbol{x}$ associated with zero-duration segments are essentially wasted. Therefore, the algorithm is unnecessarily slowed down, as a potentially large number of variables take up space and computational time while contributing nothing to the solution.

C. Switching Segments and Time for Multiple Independent Controls

The formulation presented in the previous section assumes the cardinality of each $\mathbb{U}_i$ (for $i = 1,\ \ldots,\ n_u$) is exactly equal to m. Furthermore, only synchronous switches in the control variables are allowed. That is, control variables are not allowed to switch values independent from one another. To address this limitation, an alternative formulation is presented here, one that significantly reduces the number of wasted parameters and improves overall computational efficiency. In particular, the proposed method is most useful for problems with many control variables in which $\bar{m}$ becomes excessively large. The salient feature of this formulation is that each of the control axes is treated independently. This minimizes the number of variables while offering the greatest flexibility in the solution process. Knots and segments are employed, but their definitions are slightly altered.

Let a knot be defined as any interior point for which one element of the control vector switches from one value to another. The fundamental distinction here is that each knot is associated with a control axis. Before, the knot times could be described by $\boldsymbol{T}_{n_k}$, an array with $\boldsymbol{T}_{n_k}$ values. Likewise, the segment durations were $\Delta\boldsymbol{T}_{n_s}$, and there were $n_s = n_k + 1$ segments. Now, the knot times are described as $\boldsymbol{T}_{n_u,n_k}$, a two-dimensional array with $n_u \times n_k$ values. In other words, there are knots for each control axis. The axis durations are also two-dimensional, as $\Delta\boldsymbol{T}_{n_u,n_k+1}$, with $n_u \times (n_k + 1)$ values. These are the time durations between two switches in a given axis. It is important to note that this is not necessarily the duration of the segments. A segment is now defined as the time elapsed between switches in any of the elements of the control vector. For example, the first segment is bounded both by t_0 and the first switch in the control, regardless of the axis in which the switch occurs. With $n_u n_k$ interior knots to separate the segments, the total number of segments becomes $n_s = n_u n_k + 1$. The nomenclature for the multiple independent control formulation is summarized in Table 2.

To illustrate the concept further, consider a sample case with $n_k = 3$ interior knots for each of the $n_u = 3$ control axes. Specifically, let $\mathbb{U}_i = \{-1,\ 0,\ 1\}$ for each axis. Figure 2 demonstrates the concept

Table 2 Nomenclature summary: multiple independent control formulation

Array	Description	Dimension	Element
$\boldsymbol{Y}_{n_y,n_n,n_s}$	States by node	$n_y \times n_n \times n_s$	$\boldsymbol{y}_{j,k}$ or $y_{i,j,k}$
$\boldsymbol{U}^*_{n_u,n_k+1}$	Prespecified controls	$n_u \times (n_k+1)$	$u^*_{i,k}$
$\boldsymbol{U}_{n_u,n_s}$	Controls by segment	$n_u \times n_s$	$u_{i,k}$
$\boldsymbol{U}_{n_u,n_n,n_s}$	Controls by node	$n_u \times n_n \times n_s$	$\boldsymbol{u}_{j,k}$ or $u_{i,j,k}$
$\boldsymbol{T}_{n_n,n_s}$	Node times	$n_n \times n_s$	$t_{j,k}$
$\boldsymbol{T}_{n_u,n_k}$	Knot times	$n_u \times n_k$	$t_{i,k}$
$\Delta\boldsymbol{T}_{n_u,n_k+1}$	Axis durations	$n_u \times (n_k+1)$	$\Delta t_{i,k}$
$\boldsymbol{T}'_{n_s+1}$	Unordered knot times	$0 \ \ldots \ n_s$	t'_k
$\boldsymbol{T}_{n_s+1}$	Ordered knot times	$0 \ \ldots \ n_s$	t_k
n_s	Number of segments	$n_u n_k + 1$	

at two consecutive iterations of the optimization process. The control values are predesignated in a pattern logical for an aerospace application, for which nonzero control values indicate thrusting arcs. In each control axis, the control value begins at 1 (a positive thrusting arc), and a coasting arc follows the first knot. At the second knot, the control value switches to -1 (a negative thrusting arc) until the third knot, where another coasting arc begins. With additional knots, this pattern continues. The control values in each of the n_u control axes are designated similarly. In this example, observe that the total number of segments is $n_s = (3)(3) + 1 = 10$.

Let the time elements of the parameter vector $\boldsymbol{x}$ be identified as $\boldsymbol{x}_t$. The time elements include all of the axis durations $\Delta\boldsymbol{T}_{n_u,n_k+1}$ along with the initial and final time. Here, $\boldsymbol{x}_t$ is defined as

$$\boldsymbol{x}_t = [\cdots \quad \Delta t_{i,k} \quad \cdots \quad t_0 \quad t_f]^T \tag{11}$$

The knot times $\boldsymbol{T}_{n_u,n_k}$ are defined according to elements in $\boldsymbol{x}_t$:

$$t_{i,k} = t_0 + \sum_{\kappa=1}^{k} |\Delta t_{i,\kappa}| \tag{12}$$

Thus, $\Delta t_{i,k}$ represents the time duration between the knots located at $t_{i,k-1}$ and $t_{i,k}$. The definition in Eq. (12) guarantees that the knot times remain in chronological order for a given control axis. That is,

$$t_{i,k-1} - t_{i,k} \le 0, \qquad k = 1, \ \ldots, \ n_k + 1 \tag{13}$$

where $t_{i,0} = t_0$ and $t_{i,n_k+1} = t_f$. For example, in Fig. 2, $t_{2,1}$ precedes $t_{2,2}$, which is followed by $t_{2,3}$. However, there is no relation between knot times in different control axes. Therefore, $t_{2,1}$ need not be after $t_{1,1}$. In general,

$$\iota < i \not\Rightarrow t_{\iota,\kappa} \le t_{i,k}$$
$$\forall \ i, \iota = 1, \ \ldots, \ n_u, \qquad \forall \ k, \kappa = 1, \ \ldots, \ n_k \tag{14}$$

Compare the arrangement of knot times for iteration p and iteration $p+1$ in the figure. Both illustrations represent valid arrangements for the knots. However, it is clear that the chronological ordering of all the knots may change during the optimization process.

This algorithm defines segment boundaries by the chronological ordering of knot times, including t_0 and t_f at the end points. Let the unordered segment boundaries be defined as

$$[\,t'_0 \quad t'_1 \quad \cdots \quad t'_\kappa \quad \cdots \quad t'_{n_{s-1}} \quad t'_{n_s}\,] = [\,t_0 \quad t_{1,1} \quad \cdots \quad t_{i,k} \quad \cdots \quad t_{n_u,n_k} \quad t_f\,] \tag{15}$$

Because knots are completely free to move on $[\,t_0 \quad t_f\,]$ and are independent between control axes, the segment boundaries $\boldsymbol{T}'_{n_s+1}$ are not necessarily in ascending order. Thus, a sorting algorithm converts $\boldsymbol{T}'_{n_s+1}$ into the sequential listing $\boldsymbol{T}_{n_s+1}$ of the segment boundaries. Now, t_{k-1} and t_k bound the kth segment.

Observe, for example, that segment 5 is bounded at the pth iteration by knots located at $t_{2,2}$ and $t_{3,2}$. Between iterations though, the axis durations $\Delta\boldsymbol{T}_{n_u,n_k+1}$ change values, thereby changing the positions of the knots at the next iteration. At iteration $p+1$, segment 5 is bounded by $t_{3,1}$ and $t_{2,3}$. Completely different sets of variables now define the segment boundaries. Expressing the segment boundaries for segment 5 in terms of $\boldsymbol{x}_t$, on the pth iteration,

$$t_4(\boldsymbol{x}_t) \equiv t_{2,2}(\boldsymbol{x}_t) = t_0 + |\Delta t_{2,1}| + |\Delta t_{2,2}| \tag{16}$$

$$t_5(\boldsymbol{x}_t) \equiv t_{3,2}(\boldsymbol{x}_t) = t_0 + |\Delta t_{3,1}| + |\Delta t_{3,2}| \tag{17}$$

and on the $(p+1)$th iteration,

$$t_4(\boldsymbol{x}_t) \equiv t_{3,1}(\boldsymbol{x}_t) = t_0 + |\Delta t_{3,1}| \tag{18}$$

$$t_5(\boldsymbol{x}_t) \equiv t_{2,3}(\boldsymbol{x}_t) = t_0 + |\Delta t_{2,1}| + |\Delta t_{2,2}| + |\Delta t_{2,3}| \tag{19}$$

Thus, the dependencies of segment boundaries continuously change throughout the optimization process. The term "segment time switching" is used to refer to the switching dependencies of a segment's time elements, demonstrated here. Segment time switching is the fundamental characteristic of the parameterization presented in this investigation. It is addressed in more detail subsequently.

III. Implementation of the Finite Set Control Transcription

In Sec. II, a formulation is developed for solving finite set control problems. Starting from a traditional collocation method, the transcription is modified to effectively treat problems with multiple independent switching controls. The resulting method is termed in this investigation as the finite set control transcription (FSCT) method. It is characterized by containing only the state and time elements in the parameter vector, the predesignated controls along

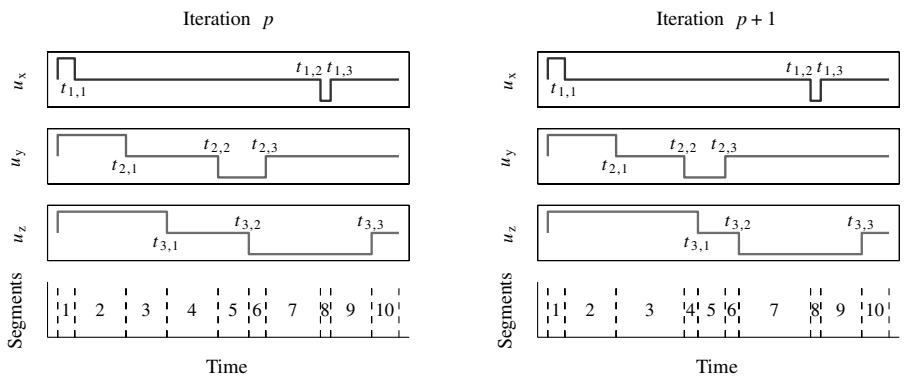

Fig. 2 Conceptual control profile with segment divisions at two consecutive iterations.

each segment, and a segment-time-switching phenomenon throughout the optimization process. Although the nature of switching dependencies is introduced previously in the paper, a more complete understanding can be gained through a description of the FSCT implementation.

Implementation considerations are presented in the context of how partial derivatives are calculated, as the performance of the optimization process is dependent upon these partials being evaluated quickly and accurately. Many literature sources discuss their role in a quasi-Newton technique [19–21]. Of course, partial derivatives can be calculated either numerically or analytically. If the user selects numerical derivatives for simplicity, some considerations are necessary to ensure success; these are discussed next. In this investigation, however, analytic derivatives are supplied to the NLP algorithm to reduce computational time. In developing analytic expressions for problem derivatives, some interesting insights into the segment-time-switching phenomenon may be observed. Specifically, switching dependencies occur within the continuity constraints and, consequently, the process of evaluating derivatives for these constraints is discussed next. This leads to further analysis of the characteristics of the FSCT method involving the accuracy of derivative evaluations and the convergence performance of the method. First, however, it is necessary to demonstrate how the optimization parameters are manipulated at the beginning of each iteration in order to properly evaluate any constraints or derivatives.

A. Optimization Parameters

In the FSCT method, the parameter vector is defined as

$$\boldsymbol{x} = [\cdots \quad y_{i,j,k} \quad \cdots \quad \cdots \quad \Delta t_{i,k} \quad \cdots \quad t_0 \quad t_f]^T \tag{20}$$

The parameterization consists of state values, time durations between control switches, and bounding times t_0 and t_f. In the evaluation of cost and constraint functions, the elements of the current point $\boldsymbol{x}$ are deparameterized and assigned to the respective state or time values that they represent. The three-dimensional array $\boldsymbol{Y}_{n_y,n_n,n_s}$ is assigned for the states, whereas the two-dimensional array $\Delta\boldsymbol{T}_{n_u,n_k+1}$ carries the time durations. The deparameterization is simply

$$\begin{aligned} y_{i,j,k} &= x_{n_y n_n(k-1)+n_y(j-1)+i}, \qquad \Delta t_{i,k} = x_{q_1+n_u(k-1)+i} \\ t_0 &= x_{q_2+1}, \qquad t_f = x_{q_2+2} \end{aligned} \tag{21}$$

where $q_1 = n_y n_n n_s$ and $q_2 = q_1 + n_u(n_k + 1)$. Adopting a shorthand notation, the expressions in Eq. (21) are implied by

$$\boldsymbol{x} \rightarrow \boldsymbol{Y}_{n_y,n_n,n_s}, \qquad \Delta\boldsymbol{T}_{n_u,n_k+1}, \qquad t_0, \qquad t_f \tag{22}$$

where $a \rightarrow b$ indicates a mapping from input a to output b. Likewise, the application of Eq. (12) reveals the times of each knot:

$$t_0, \qquad \Delta\boldsymbol{T}_{n_u,n_k+1} \rightarrow \boldsymbol{T}_{n_u,n_k} \tag{23}$$

Recall that the elements of $\boldsymbol{T}_{n_u,n_k}$ are $t_{i,k}$, where the control axis and knot number are distinguished by i and k, respectively. Thus, in this form, knot times are arranged in a two-dimensional array, and the chronological ordering of knot times is unknown.

To evaluate continuity constraint equations, the states, controls, and time must be known at each node of each segment. The states are contained in the array $\boldsymbol{Y}_{n_y,n_n,n_s}$ and extracted directly from $\boldsymbol{x}$. The node times $\boldsymbol{T}_{n_n,n_s}$ are easily determined through Eq. (9) if the segment boundaries $\boldsymbol{T}_{n_s+1}$ are known. Presently, then, $\boldsymbol{U}_{n_u,n_n,n_s}$ and $\boldsymbol{T}_{n_s+1}$ are extracted. Notice that the controls and segment boundaries are related, as the latter are simply the knot times which define switches in the control. The arrays are determined simultaneously, then, as the ordering of knots ultimately defines the control sequence.

In the first step, the segment boundaries are placed in an unordered listing according to Eq. (15). The segment boundaries include the initial time, the final time, and each knot contained in $\boldsymbol{T}_{n_u,n_k}$. In shorthand,

$$t_0, \boldsymbol{T}_{n_u,n_k}, t_f \rightarrow \boldsymbol{T}'_{n_{s+1}} \tag{24}$$

where the elements of the output are $t'_\kappa, \kappa = 0, \ldots, n_s$. It is clear that $t'_0 = t_0$ and $t'_{n_s} = t_f$; however, the interior segment boundaries are not arranged chronologically. Each interior segment boundary (t'_κ, $\kappa = 1, \ldots, n_s - 1$) is directly linked to an element $t_{i,k}$, and it is necessary to maintain in storage the i, k pairing associated with each segment boundary. Next, a sorting algorithm places the total collection of knots in chronological order to define the interior segment boundaries:

$$\boldsymbol{T}'_{n_s+1} \rightarrow \boldsymbol{T}_{n_s+1} \tag{25}$$

The segment boundaries $\boldsymbol{T}_{n_s+1}$ represent the chronological ordering of knots. When the segment boundaries are sorted, the respective i, k pairings must be passed along. Thus, it is crucial to identify not only the time for each knot in $\boldsymbol{T}_{n_s+1}$ but also the element in $\boldsymbol{T}_{n_u,n_k}$ from which it came. This provides the basis for determining the control sequence along each segment.

Let the element $u^*_{i,k}$ be the prespecified control value for the ith control axis between $t_{i,k-1}$ and $t_{i,k}$. The array of prespecified controls is $\boldsymbol{U}^*_{n_u,n_k+1}$ and has the dimension $n_u \times n_k + 1$. In conjunction with the sorted knots, the segment control values are determined:

$$\boldsymbol{T}_{n_u,n_k}, \boldsymbol{T}_{n_s+1}, \boldsymbol{U}^*_{n_u,n_{k+1}} \rightarrow \boldsymbol{U}_{n_u,n_s} \tag{26}$$

The control values are completely contained in $\boldsymbol{U}_{n_u,n_s}$, where $u_{i,k}$ indicates the control value for the ith axis on the kth segment. In this process, the control values along the first segment are simply the first prespecified control values; that is

$$u_{i,1} = u^*_{i,1} \tag{27}$$

Then, at each segment boundary, all control values are held constant except in the control axis associated with the given knot. Because this knot identifies a control switch in its corresponding axis, the value in that control axis is updated to its next prespecified value. Thus, if κ_i indicates the current control value of the ith axis, then

$$u_{i,k} = u^*_{i,\kappa_i} \tag{28}$$

$$u_{i,k+1} = \begin{cases} u^*_{i,\kappa_i+1} & t_k \equiv t_{i,\kappa_i} \\ u^*_{i,\kappa_i} & \text{otherwise} \end{cases} \tag{29}$$

Upon each control switch, the update $\kappa_i = \kappa_i + 1$ is performed in the switching axis to complete the recursive step. Equation (29) guarantees that along the final segment:

$$u_{i,n_s} = u^*_{i,n_k+1} \tag{30}$$

In a final step, the controls are determined at each node:

$$\boldsymbol{U}_{n_u,n_s} \rightarrow \boldsymbol{U}_{n_u,n_n,n_s} \tag{31}$$

Because the controls are assumed constant over all segments, it is clear that $u_{i,j,k} = u_{i,k}$ for all nodes $j = 1, \ldots, n_n$. With this, all necessary elements are extracted, and the cost function and constraints can be evaluated in terms of the extracted elements.

B. Dynamical Constraints Using Simpson Integration Equations

The dynamical constraints are the central feature of a collocation method. When they are satisfied to tolerance, state continuity exists in the context of the given dynamical model. The dynamical constraints are also the key functions affected by the segment-time-switching phenomenon. Although any integration scheme may be used, the Hermite–Simpson [18] integration equation is employed in this investigation. It is presented here to aid in the following discussion on the switching phenomenon.

Recall that the $n_y(n_n-1)n_s$ dynamical constraints are of the form

$$\boldsymbol{c}_{\dot{\boldsymbol{y}}}(\boldsymbol{x}) = [\,\boldsymbol{c}^T_{\dot{\boldsymbol{y}}_{1,1}}(\boldsymbol{x}) \quad \cdots \quad \boldsymbol{c}^T_{\dot{\boldsymbol{y}}_{j,k}}(\boldsymbol{x}) \quad \cdots \quad \boldsymbol{c}^T_{\dot{\boldsymbol{y}}_{n_n-1,n_s}}(\boldsymbol{x})\,]^T \tag{32}$$

where

$$\begin{aligned}\boldsymbol{c}_{\dot{\boldsymbol{y}}_{j,k}}(\boldsymbol{x}) = \boldsymbol{y}_{j+1,k} - \boldsymbol{y}_{j,k} - \tfrac{h}{6}[\boldsymbol{f}(t_{j,k},\, \boldsymbol{y}_{j,k},\, \boldsymbol{u}_{j,k}) \\ + 4\boldsymbol{f}(t_m,\, \boldsymbol{y}_m,\, \boldsymbol{u}_m) + \boldsymbol{f}(t_{j+1,k},\, \boldsymbol{y}_{j+1,k},\, \boldsymbol{u}_{j+1,k})]\end{aligned} \tag{33}$$

In Eq. (33), the time variables are determined from previously extracted quantities by

$$\begin{aligned}h = \frac{t_k - t_{k-1}}{n_n - 1}, \qquad t_{j,k} = t_{k-1} + h(j-1) \\ t_m = \frac{1}{2}(t_{j,k} + t_{j+1,k})\end{aligned} \tag{34}$$

Note that t_k are elements of the one-dimensional array $\boldsymbol{T}_{n_s+1}$, representing segment boundaries, whereas $t_{j,k}$ are members of the two-dimensional array $\boldsymbol{T}_{n_n,n_s}$, denoting the node times within a segment. The already extracted state and control values at the nodes $\boldsymbol{y}_{j,k}$ and $\boldsymbol{u}_{j,k}$ are used to determine the midpoint states and controls:

$$\boldsymbol{y}_m = \tfrac{1}{2}(\boldsymbol{y}_{j,k} + \boldsymbol{y}_{j+1,k}) + \tfrac{h}{8}(\boldsymbol{f}_{j,k} - \boldsymbol{f}_{j+1,k}) \tag{35}$$

$$\boldsymbol{u}_m = \tfrac{1}{2}(\boldsymbol{u}_{j,k} + \boldsymbol{u}_{j+1,k}) = \boldsymbol{u}_{j,k} = \boldsymbol{u}_{j+1,k} \tag{36}$$

The result in Eq. (36) is trivial, because the controls remain constant over any segment.

1. Partial Derivatives for the Simpson Integration Equations

For any set of dynamics, the general partial derivatives of $\boldsymbol{c}_{\dot{\boldsymbol{y}}_{j,k}}$ may be determined as follows:

$$\frac{\partial \boldsymbol{c}_{\dot{\boldsymbol{y}}_{j,k}}}{\partial \boldsymbol{y}_{j,k}} = -\boldsymbol{I} - \frac{h}{6}\left(\frac{\partial \boldsymbol{f}_{j,k}}{\partial \boldsymbol{y}_{j,k}} + 4\frac{\partial \boldsymbol{f}_m}{\partial \boldsymbol{y}_m}\frac{\partial \boldsymbol{y}_m}{\partial \boldsymbol{y}_{j,k}}\right) \tag{37}$$

$$\frac{\partial \boldsymbol{c}_{\dot{\boldsymbol{y}}_{j,k}}}{\partial \boldsymbol{y}_{j+1,k}} = \boldsymbol{I} - \frac{h}{6}\left(4\frac{\partial \boldsymbol{f}_m}{\partial \boldsymbol{y}_m}\frac{\partial \boldsymbol{y}_m}{\partial \boldsymbol{y}_{j+1,k}} + \frac{\partial \boldsymbol{f}_{j+1,k}}{\partial \boldsymbol{y}_{j+1,k}}\right) \tag{38}$$

$$\frac{\partial \boldsymbol{c}_{\dot{\boldsymbol{y}}_{j,k}}}{\partial \boldsymbol{u}_{j,k}} = -\frac{h}{6}\left(\frac{\partial \boldsymbol{f}_{j,k}}{\partial \boldsymbol{u}_{j,k}} + 4\frac{\partial \boldsymbol{f}_m}{\partial \boldsymbol{u}_m}\frac{\partial \boldsymbol{u}_m}{\partial \boldsymbol{u}_{j,k}}\right) \tag{39}$$

$$\frac{\partial \boldsymbol{c}_{\dot{\boldsymbol{y}}_{j,k}}}{\partial \boldsymbol{u}_{j+1,k}} = -\frac{h}{6}\left(4\frac{\partial \boldsymbol{f}_m}{\partial \boldsymbol{u}_m}\frac{\partial \boldsymbol{u}_m}{\partial \boldsymbol{u}_{j+1,k}} + \frac{\partial \boldsymbol{f}_{j+1,k}}{\partial \boldsymbol{u}_{j+1,k}}\right) \tag{40}$$

$$\begin{aligned}\frac{\partial \boldsymbol{c}_{\dot{\boldsymbol{y}}_{j,k}}}{\partial t_{k-1}} = -\frac{1}{6}(\boldsymbol{f}_{j,k} + 4\boldsymbol{f}_m + \boldsymbol{f}_{j+1,k})\frac{\partial h}{\partial t_{k-1}} - \frac{h}{6}\left[\frac{\partial \boldsymbol{f}_{j,k}}{\partial t_{j,k}}\frac{\partial t_{j,k}}{\partial t_{k-1}}\right. \\ \left. + 4\left(\frac{\partial \boldsymbol{f}_m}{\partial \boldsymbol{y}_m}\frac{\partial \boldsymbol{y}_m}{\partial h}\frac{\partial h}{\partial t_{k-1}} + \frac{\partial \boldsymbol{f}_m}{\partial t_m}\frac{\partial t_m}{\partial t_{k-1}}\right) + \frac{\partial \boldsymbol{f}_{j+1,k}}{\partial t_{j+1,k}}\frac{\partial t_{j+1,k}}{\partial t_{k-1}}\right]\end{aligned} \tag{41}$$

$$\begin{aligned}\frac{\partial \boldsymbol{c}_{\dot{\boldsymbol{y}}_{j,k}}}{\partial t_k} = -\frac{1}{6}(\boldsymbol{f}_{j,k} + 4\boldsymbol{f}_m + \boldsymbol{f}_{j+1,k})\frac{\partial h}{\partial t_k} - \frac{h}{6}\left[\frac{\partial \boldsymbol{f}_{j,k}}{\partial t_{j,k}}\frac{\partial t_{j,k}}{\partial t_k}\right. \\ \left. + 4\left(\frac{\partial \boldsymbol{f}_m}{\partial \boldsymbol{y}_m}\frac{\partial \boldsymbol{y}_m}{\partial h}\frac{\partial h}{\partial t_k} + \frac{\partial \boldsymbol{f}_m}{\partial t_m}\frac{\partial t_m}{\partial t_k}\right) + \frac{\partial \boldsymbol{f}_{j+1,k}}{\partial t_{j+1,k}}\frac{\partial t_{j+1,k}}{\partial t_k}\right]\end{aligned} \tag{42}$$

In Eqs. (37–42), $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}$, $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}$, and $\frac{\partial \boldsymbol{f}}{\partial t}$ are dependent on the chosen dynamics and

$$\begin{aligned}\frac{\partial \boldsymbol{y}_m}{\partial \boldsymbol{y}_{j,k}} = \frac{1}{2}\boldsymbol{I} + \frac{h}{8}\frac{\partial \boldsymbol{f}_{j,k}}{\partial \boldsymbol{y}_{j,k}}, \qquad \frac{\partial \boldsymbol{y}_m}{\partial \boldsymbol{y}_{j+1,k}} = \frac{1}{2}\boldsymbol{I} - \frac{h}{8}\frac{\partial \boldsymbol{f}_{j+1,k}}{\partial \boldsymbol{y}_{j+1,k}} \\ \frac{\partial \boldsymbol{u}_m}{\partial \boldsymbol{u}_{j,k}} = \frac{1}{2}\boldsymbol{I}, \qquad \frac{\partial \boldsymbol{u}_m}{\partial \boldsymbol{u}_{j+1,k}} = \frac{1}{2}\boldsymbol{I} \\ \frac{\partial h}{\partial t_{k-1}} = -\frac{1}{n_n - 1}, \qquad \frac{\partial h}{\partial t_k} = \frac{1}{n_n - 1} \\ \frac{\partial t_{j,k}}{\partial t_{k-1}} = 1 - \frac{j-1}{n_n - 1}, \qquad \frac{\partial t_{j,k}}{\partial t_k} = \frac{j-1}{n_n - 1} \\ \frac{\partial t_m}{\partial t_{k-1}} = 1 - \frac{j-\frac{1}{2}}{n_n - 1}, \qquad \frac{\partial t_m}{\partial t_k} = \frac{j-\frac{1}{2}}{n_n - 1}\end{aligned} \tag{43}$$

Considering the formulation presented in this investigation, one may observe immediately that the derivatives $(\partial \boldsymbol{c}_{\dot{\boldsymbol{y}}_{j,k}})/(\partial \boldsymbol{x})$ are completely defined without applying Eqs. (39) and (40). Because the control values are prespecified in this formulation, they are not optimization variables and their constraint partials can be ignored. They are included previously in the paper only for completeness.

In contrast, Eqs. (37) and (38) are directly implementable, as $\boldsymbol{y}_{j,k}$ are variables contained in the parameter vector $\boldsymbol{x}$. If the variables are divided into state elements and time elements, $\boldsymbol{x} = [\boldsymbol{x}_y^T \quad \boldsymbol{x}_t^T\,]^T$, then the Jacobian elements are

$$\frac{\partial \boldsymbol{c}_{\dot{\boldsymbol{y}}_{j,k}}}{\partial \boldsymbol{x}_{y_\gamma}} = \begin{cases} \frac{\partial \boldsymbol{c}_{\dot{\boldsymbol{y}}_{j,k}}}{\partial \boldsymbol{y}_{j,k}} & \boldsymbol{x}_{y_\gamma} \equiv \boldsymbol{y}_{j,k} \\ \frac{\partial \boldsymbol{c}_{\dot{\boldsymbol{y}}_{j,k}}}{\partial \boldsymbol{y}_{j+1,k}} & \boldsymbol{x}_{y_\gamma} \equiv \boldsymbol{y}_{j+1,k} \\ \boldsymbol{0} & \text{otherwise} \end{cases} \tag{44}$$

Alternatively, Eqs. (41) and (42) are not immediately implementable because the time variables t_{k-1} and t_k do not appear directly as parameters in $\boldsymbol{x}$. However, t_{k-1} and t_k are dependent upon the time parameters $\boldsymbol{x}_t$, such that

$$\frac{\partial \boldsymbol{c}_{\dot{\boldsymbol{y}}_{j,k}}}{\partial \boldsymbol{x}_t} = \frac{\partial \boldsymbol{c}_{\dot{\boldsymbol{y}}_{j,k}}}{\partial t_{k-1}}\frac{\partial t_{k-1}}{\partial \boldsymbol{x}_t} + \frac{\partial \boldsymbol{c}_{\dot{\boldsymbol{y}}_{j,k}}}{\partial t_k}\frac{\partial t_k}{\partial \boldsymbol{x}_t} \tag{45}$$

To apply Eq. (45), the partials $(\partial t_k)/(\partial \boldsymbol{x}_t)$ must be determined. Recall from Sec. III.A that segment boundaries of $\boldsymbol{T}_{n_s+1}$ are directly linked to knot times $\boldsymbol{T}_{n_u,n_k}$ that are functions of the parameters in $\boldsymbol{x}_t$. Therefore, it is possible to determine

$$\frac{\partial t_k}{\partial t_0}, \qquad \frac{\partial t_k}{\partial t_f} \tag{46}$$

and

$$\frac{\partial t_k}{\partial \Delta t_{i,\kappa}}$$

for all segments boundaries $k = 0, \ldots, n_s$, control elements $i = 1, \ldots, n_u$, and knots $\kappa = 1, \ldots, n_k$. Beginning with exterior segment boundaries, it is clear that, because $t_{n_s} = t_f$,

$$\begin{aligned}\frac{\partial t_0}{\partial t_0} = 1, \qquad \frac{\partial t_{n_s}}{\partial t_0} = 0, \qquad \frac{\partial t_0}{\partial t_f} = 0 \\ \frac{\partial t_{n_s}}{\partial t_f} = 1, \qquad \frac{\partial t_0}{\partial \Delta t_{i,\kappa}} = 0, \qquad \frac{\partial t_{n_s}}{\partial \Delta t_{i,\kappa}} = 0\end{aligned} \tag{47}$$

This is true regardless of the ordering for interior knots. However, interior segment boundaries in $\boldsymbol{T}_{n_s+1}$ correspond to elements in the knot array $\boldsymbol{T}_{n_u,n_k}$. That is,

$$t_k \equiv t_{i,\kappa}, \qquad 1 \le k \le n_s - 1 \tag{48}$$

for some pair i, κ. Likewise, the derivatives of the element $t_{i,\kappa}$ are assigned to the element t_k:

$$\frac{\partial t_k}{\partial \boldsymbol{x}_t} \equiv \frac{\partial t_{i,\kappa}}{\partial \boldsymbol{x}_t}, \qquad 1 \le k \le n_s - 1 \tag{49}$$

Thus, all that remains is to identify the proper derivatives for the knot times. Under the assumption that all $\Delta t_{i,\kappa} \ge 0$,

$$\frac{\partial t_{i,k}}{\partial t_0} = 1, \qquad \frac{\partial t_{i,k}}{\partial \Delta t_{\iota,\kappa}} = \begin{cases} 1, & \iota = i, \kappa \le k \\ 0, & \text{otherwise} \end{cases}, \qquad \frac{\partial t_{i,k}}{\partial t_f} = 0 \quad (50)$$

With these derivatives carefully matched, the components of Eq. (45) are completely defined.

To further understand the impact of segment time switching on derivative evaluation, consider again the illustration of Fig. 2. The partial derivatives of the segment boundary times with respect to $\boldsymbol{x}_t$ at the pth and $(p+1)$th iterations are given next:

$$\boldsymbol{x}_t = [\cdots \quad \Delta t_{i,k} \quad \cdots \quad t_0 \quad t_f]^T$$

$$\left(\frac{\partial t_4}{\partial \boldsymbol{x}_t}\right)_p = [0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0]$$

$$\left(\frac{\partial t_5}{\partial \boldsymbol{x}_t}\right)_p = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0]$$

$$\left(\frac{\partial t_4}{\partial \boldsymbol{x}_t}\right)_{p+1} = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0]$$

$$\left(\frac{\partial t_5}{\partial \boldsymbol{x}_t}\right)_{p+1} = [0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0]$$

Thus, the segment boundary derivatives with respect to elements in $\boldsymbol{x}_t$ are always either 0 or 1, and it is the ordering of the knots that determines the proper structure at each iteration. The constraint derivatives for the fifth segment, then, are defined by

$$\frac{\partial \boldsymbol{c}_{\dot{y}_{j,5}}}{\partial \boldsymbol{x}_t} = \frac{\partial \boldsymbol{c}_{\dot{y}_{j,5}}}{\partial t_4}\frac{\partial t_4}{\partial \boldsymbol{x}_t} + \frac{\partial \boldsymbol{c}_{\dot{y}_{j,5}}}{\partial t_5}\frac{\partial t_5}{\partial \boldsymbol{x}_t} \quad (51)$$

and clearly demonstrate dramatically different behavior on each iteration.

2. Time Invariance

The parameter t_0 appears in the definition of every knot time, with the exception of $t_{n_s} = t_f$. For an interior segment k, then, t_0 defines both t_{k-1} and t_k. If the dynamic model is time invariant, the partial derivatives $(\partial \boldsymbol{c}_{\dot{y}_{j,k}})/(\partial t_{k-1})$ and $(\partial \boldsymbol{c}_{\dot{y}_{j,k}})/(\partial t_k)$ have a canceling effect in Eq. (45), such that

$$\frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial t_0} = \boldsymbol{0} \quad (52)$$

on an interior segment. Thus, because the parameters are axis durations instead of absolute times, t_0 and t_f only have nonzero effects on the last segment.

Likewise, in a time invariant formulation, if a segment k is bounded by two knots associated with the same axis (such as $t_{i,\kappa-1}$ and $t_{i,\kappa}$), all of the partials will cancel out with exception to that associated with $\Delta t_{i,\kappa}$. That is,

$$\frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial \Delta t_{i,\kappa}} = \frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial t_k} \quad (53)$$

$$\frac{\partial \boldsymbol{c}_{\dot{y}_{j,k}}}{\partial (x_t)_\gamma} = \boldsymbol{0}, \qquad \text{otherwise} \quad (54)$$

3. Derivative Discontinuities for the Dynamical Constraints

The dynamical constraints exhibit discontinuities in function derivatives when the chronological ordering of the knots switch. This is demonstrated in how the partials $(\partial t_k)/(\partial \boldsymbol{x}_t)$ may change between the values 0 and 1 between iterations. A simple analogy is useful in conceptualizing this phenomenon. Consider a simple function of two variables defined by

$$f(x_1, x_2) = \min(x_1, x_2) \quad (55)$$

If this function appears in a parameter optimization problem as either the objective or a constraint, its evaluation is straightforward:

$$f = \begin{cases} x_1 & x_1 < x_2 \\ x_1 = x_2 & x_1 = x_2 \\ x_2 & x_1 > x_2 \end{cases} \quad (56)$$

Interestingly, when $x_1 < x_2$, variable x_2 has seemingly no impact on the function. However, there is a switch in dependency when x_2 is smaller than x_1. The characteristics of this function are quite similar to those of the dynamical constraints in the FSCT method. Consider the derivative with respect to the first variable:

$$\frac{\partial f}{\partial x_1} = \begin{cases} 1 & x_1 < x_2 \\ \text{undefined} & x_1 = x_2 \\ 0 & x_1 > x_2 \end{cases} \quad (57)$$

Thus, a derivative discontinuity exists when x_1 and x_2 are equal. However, an NLP algorithm requires derivatives to be tractable at any point within the range of optimization. Therefore, if evaluating $(\partial f)/(\partial x_1)$ or $(\partial f)/(\partial x_2)$ analytically, it is up to the user to choose an appropriate definition in the case that $x_1 = x_2$. Three reasonable candidates for $(\partial f)/(\partial x_1)|_{x_1=x_2}$ may be seen in how numerical derivatives could be evaluated using forward, backward, or central differences:

Forward

$$\left.\frac{\partial f}{\partial x_1}\right|_{x_1=x_2} = \frac{f(x_1+\delta, x_2) - f(x_1, x_2)}{\delta} = 0 \quad (58)$$

Backward

$$\left.\frac{\partial f}{\partial x_1}\right|_{x_1=x_2} = \frac{f(x_1, x_2) - f(x_1-\delta, x_2)}{\delta} = 1$$

Central

$$\left.\frac{\partial f}{\partial x_1}\right|_{x_1=x_2} = \frac{f(x_1+\delta, x_2) - f(x_1-\delta, x_2)}{2\delta} = \frac{1}{2}$$

Thus, choosing an analytic expression for $(\partial f)/(\partial x_1)|_{x_1=x_2}$ is equivalent to selecting a finite differencing scheme for numerically evaluated derivatives. In most cases, the selection is arbitrary, as there is probably no basis for valuing one method over another. However, it is clear that when $x_1 = x_2$, the path of the NLP algorithm can be altered dramatically simply by selecting a different derivative definition.

This conceptualization is easily extended to the continuity constraints of the FSCT method. Consider as an example the four knot sequence described in Fig. 3, in which $t_0 \le t_1 \le t_2 \le t_3$. The first four knots are defined by t_0 and the knot times $t_{2,1}$, $t_{3,1}$, and $t_{1,1}$, respectively. This implies that, in this example,

$$\Delta t_{2,1} \le \Delta t_{3,1} \le \Delta t_{1,1} \quad (59)$$

According to the FSCT implementation of analytic derivatives, the associated dynamics equations for the three segments shown have time dependencies, as given in Table 3. Thus, when time derivatives are calculated for the dynamical constraints on one of these segments, nonzero Jacobian elements appear for their respective dependent members of $\boldsymbol{x}_t$. The derivatives with respect to all other members of $\boldsymbol{x}_t$, including t_f and other members of $\Delta \boldsymbol{T}_{n_u,n_k+1}$, are identically zero.

Fig. 3 Sample four knot sequence.

Table 3 Segment dependencies for analytic and numerical (forward differencing) derivatives

		Dependent Members of $\boldsymbol{x}_t$	
Segment	Boundaries	Analytic derivatives	Forward differences
1	t_0, t_1	$t_0, \Delta t_{2,1}$	$t_0, \boldsymbol{\Delta t}_{3,1}$
2	t_1, t_2	$t_0, \Delta t_{2,1}, \Delta t_{3,1}$	$t_0, \boldsymbol{\Delta t}_{3,1}, \boldsymbol{\Delta t}_{2,1}$
3	t_2, t_3	$t_0, \Delta t_{3,1}, \Delta t_{1,1}$	$t_0, \boldsymbol{\Delta t}_{2,1}, \Delta t_{1,1}$

As long as the knots are sufficiently far away, no difficulties confront the solution process. However, consider further the scenario that

$$\Delta t_{2,1} = \Delta t_{3,1} \leq \Delta t_{1,1} \tag{60}$$

such that $t_1 = t_2$. In the context of the dynamical problem, this situation indicates that the controls in two axes must switch simultaneously. Thus, knots come together to exist at the same point in time, but their ordering is still specified according to the implementation (in this case, $t_1 \equiv t_{2,1}$ is ordered before $t_2 \equiv t_{3,1}$). In this situation, it is possible for the $\boldsymbol{x}_t$ dependencies determined by numerical derivatives to be completely different than the analytic set. To illustrate, assume that finite differences are calculated using forward differences, such that

$$\frac{\partial c}{\partial x} \approx \frac{c(x+\delta) - c(x)}{\delta} \tag{61}$$

Then, when $\Delta t_{2,1}$ is perturbed forward by $\delta > 0$, the knot ordering will necessarily change because

$$\Delta t_{3,1} \leq \Delta t_{2,1} + \delta \leq \Delta t_{1,1} \tag{62}$$

The $\boldsymbol{x}_t$ dependencies via forward differences are also presented in Table 3. To contrast the two derivative methods, variations for forward differences are highlighted. Thus, according to the forward difference, changes to $\Delta t_{2,1}$ have no effect on segment 1, as its bounding times remain the same even when the parameter is perturbed. Instead, when $\Delta t_{2,1}$ is perturbed forward, it affects segment 3, despite the fact that knot ordering indicated otherwise. Likewise, on segment 2, analytic calculations consider $\Delta t_{2,1}$ as the left bounding time, whereas a forward differencing scheme necessarily implies it is the right bounding time. On segment 2, one method will result in $(\partial c)/(\partial \Delta t_{2,1}) > 0$, whereas the other will compute a negative value.

Therefore, as with the sample function $f = \min(x_1, x_2)$, several derivative definitions can be employed at the switching points for which derivative discontinuities exist. It is clear that the path of $\boldsymbol{x}$ chosen by the NLP algorithm will vary based on the derivatives calculated, but this does not inhibit the FSCT method. Considering the total number of parameters in $\boldsymbol{x}$ and the relatively few gradients that can be affected by the switching phenomenon, the NLP algorithm can still make improvements in $\boldsymbol{x}$ on an iteration in which a switch occurs, regardless of how the derivative is defined. Although the specific search direction of the switching iteration may vary, it is observed that an identical local minimum can be found via different paths. Thus, these derivative discontinuities do not present obstacles in determining an optimal solution.

It should be noted before proceeding that this situation only arises when two knots originating from different control axes occur simultaneously. Based on the definitions of knot times, knots in one control axis cannot switch in order. However, two knots in one axis can exist simultaneously when a given element $\Delta t_{i,k} = 0$. In fact, this is a common occurrence, as the user may prespecify a value of $u^*_{i,k}$ that is simply not desired in the optimal solution. The NLP algorithm makes zero a time duration in order to remove the nonoptimal control value. Having simultaneous knots in a single axis is an anticipated condition, and it is important that this situation does not cause any adverse effects in derivative calculations. This is guaranteed in the definition of Eq. (12), which is specifically formulated to produce this result.

C. Implementation of Numerical Derivatives

When a user selects a finite differencing method for evaluating function derivatives, some additional considerations may be required. Depending on the specific optimization algorithm, the NLP algorithm may calculate finite differences automatically in the absence of analytic derivative expressions. If this is the case, the FSCT method derivatives may not be effectively evaluated without user intervention.

Some sophisticated NLP algorithms perform initialization routines to more efficiently calculate numerical derivatives. To avoid excess computation, functions are evaluated from an initial or random point of $\boldsymbol{x}$ to determine the structure of the Jacobian matrix. Linear constraints (constant Jacobian elements) and nonlinear constraints (varying Jacobian elements) are identified. This allows the routine to characterize the sparsity of the Jacobian. By performing this task in the initialization, the algorithm seeks to avoid recalculating nonchanging elements. However, in the context of the FSCT implementation, this procedure will not be able to adequately identify the potential dependencies that would appear from a different evaluation point. Some Jacobian elements will appear to be nonvarying at a particular point based on the current arrangement of knots. If the initialization routine falsely identifies elements as constant (zero or nonzero), then proper derivatives are not evaluated for future iterations when the knot arrangement is different. Consequently, the NLP algorithm cannot determine the best search direction for the next iteration point, and it is unlikely that the NLP algorithm would converge on an optimal point.

This can be overcome with user-defined procedures to flag all potential dependencies (nonzero Jacobian elements) as varying gradients. With this information, the NLP algorithm will perform finite differences for each element, actively determining the dependencies on each iteration. Efficiency degrades, and there is necessarily excess computation, but derivatives can be calculated accurately.

D. Other Constraints

So far, the implementation of the FSCT method is presented in the context of the continuity constraints and the segment-time-switching phenomenon that exists in those constraints. However, the continuity constraints along segments are only a subset of the constraints necessary for successful implementation of the method. Of course, the specific set of constraints may be dependent upon the actual problem transcribed. However, common to most applications are three additional subsets of constraints described presently. These implement various initial, knot, and time conditions.

1. Initial States and Time

The initial conditions most often employed with optimal control problems have all of the initial states as well as the initial time fixed. Thus, the constraints $\boldsymbol{\psi}_0(t_0, \boldsymbol{y}_0) = \boldsymbol{0}$ will capture fixed initial states and time.

Notice that the initial states are simply the states assigned to the first node of the first segment $y_{i,1,1}$. Let the specified initial states be identified as $\boldsymbol{y}^*_0$ and the specified initial time as t^*_0. Then, $n_{\psi_0} = n_y + 1$ constraints are imposed to represent $\boldsymbol{\psi}_0$ as

$$\boldsymbol{c}_{\psi_0}(\boldsymbol{x}) = \begin{bmatrix} y_{1,1,1} - (y^*_0)_1 \\ \vdots \\ y_{i,1,1} - (y^*_0)_i \\ \vdots \\ y_{n_y,1,1} - (y^*_0)_{n_y} \\ t_0 - t^*_0 \end{bmatrix} \tag{63}$$

The NLP algorithm drives this vector to zero during the optimization. Although, simple constraints like this could and should be met on the first iteration by appropriately assigning the initial conditions in the first guess for $\boldsymbol{x}$.

The Jacobian elements for these constraints are simple to compute:

$$\frac{\partial c_{\psi_{0_i}}}{\partial x_\gamma} = \begin{cases} 1 & x_\gamma \equiv y_{i,1,1} \Leftrightarrow \gamma = i \\ 0 & \text{otherwise} \end{cases} \tag{64}$$

$$\frac{\partial c_{\psi_{0_{n_y+1}}}}{\partial x_\gamma} = \begin{cases} 1 & x_\gamma \equiv t_0 \Leftrightarrow \gamma = n_y n_n n_s + n_u(n_k+1)+1 \\ 0 & \text{otherwise} \end{cases} \tag{65}$$

for $i = 1, \ldots, n_y$.

2. Segment Continuity Between Knots

Consider a set of dynamics in which all states must be continuous across segments. Because no time elapses between the end of the kth segment and the beginning of the $(k+1)$th segment, the states must be equal at those points. The $n_y(n_s-1)$ segment constraints are formulated as

$$\boldsymbol{c}_s(\boldsymbol{x}) = \begin{bmatrix} y_{1,1,2} - y_{1,n_n,1} \\ \vdots \\ y_{i,1,k+1} - y_{i,n_n,k} \\ \vdots \\ y_{n_y,1,n_s} - y_{n_y,n_n,n_s-1} \end{bmatrix} \tag{66}$$

with Jacobian elements

$$\frac{\partial c_{s_{n_y(k-1)+i}}}{\partial x_\gamma} = \begin{cases} 1 & x_\gamma \equiv y_{i,1,k+1} \\ -1 & x_\gamma \equiv y_{i,n_n,k} \\ 0 & \text{otherwise} \end{cases} \tag{67}$$

for $i = 1, \ldots, n_y$ and $k = 1, \ldots, n_s - 1$.

3. Time

Time equality constraints ensure that the sums of the axes durations for each control axis are equal to the trajectory time of flight $t_f - t_0$. That is, the n_u time constraints are

$$\boldsymbol{c}_t(\boldsymbol{x}) = \begin{bmatrix} t_f - t_0 - \sum_{\kappa=1}^{n_k+1} |\Delta t_{1,\kappa}| \\ \vdots \\ t_f - t_0 - \sum_{\kappa=1}^{n_k+1} |\Delta t_{i,\kappa}| \\ \vdots \\ t_f - t_0 - \sum_{\kappa=1}^{n_k+1} |\Delta t_{n_u,\kappa}| \end{bmatrix} \tag{68}$$

The Jacobian elements for these constraints are

$$\frac{\partial c_{t_i}}{\partial x_\gamma} = \begin{cases} 1 & x_\gamma \equiv t_f \\ -1 & x_\gamma \equiv t_0 \\ -1 & x_\gamma \equiv \Delta t_{i,k} \\ 0 & \text{otherwise} \end{cases} \tag{69}$$

for $i = 1, \ldots, n_u$ and $k = 1, \ldots, n_k + 1$. For these derivatives, it is assumed that $\Delta t_{i,k} \geq 0$. This is a reasonable assumption if simple bounds keep the time durations nonnegative. Alternatively, the derivatives associated with the durations are $-\text{sign}(\Delta t_{i,k})$ instead of -1.

IV. Application: Lunar Lander

With the FSCT method and its implementation considerations thus introduced, its characteristics are easily demonstrated through an application. The classical lunar launch/lunar lander problem is commonly treated in the literature on optimal control theory [22,23]. The objective for the launch problem is to transfer a rocket from the lunar surface into a lunar orbit in minimum time. The problem is constructed in two dimensions (range and altitude), yielding four states (position and velocity in each dimension) and one control variable (thrust direction angle). The thrust acceleration magnitude and the gravitational field are assumed constant. The rocket is initially at rest and must achieve a specified final altitude and range velocity. It is observed that the lunar lander problem (assuming a soft landing) is the identical problem, integrating backward. The simplicity and familiarity of this problem make it an interesting test case for the FSCT method.

Consider the lunar lander problem with one added complexity: the vehicle cannot alter its thrust vector. Instead, the vehicle can thrust with constant acceleration magnitude in each principal direction. The dynamics are described by

$$\dot{\boldsymbol{y}} = \begin{bmatrix} \dot{r}_1 \\ \dot{r}_2 \\ \dot{v}_1 \\ \dot{v}_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ u_1 \\ -g + u_2 \end{bmatrix} \tag{70}$$

where r, v, and u represent position, velocity, and control acceleration, respectively, and the subscripts indicate the horizontal and vertical dimensions. The gravitational constant of $g = 1.6231$ m/s^2 is obtained using the mass and equatorial radius of the moon. With initial conditions $\boldsymbol{r}_0 = [200 \quad 15]^T$ km and $0 = [-1.7 \quad 0]^T$ km/s and final conditions $\boldsymbol{r}_f = \boldsymbol{v}_f = \boldsymbol{0}$, the lander must achieve a soft landing on a specified target from a completely specified initial state. Both minimum time and minimum fuel optimizations are realized with the finite set control constraints:

$$u_1 \in \{-\tilde{u}_1, \quad 0, \quad \tilde{u}_1\} \tag{71}$$

$$u_2 \in \{-\tilde{u}_2, \quad 0, \quad \tilde{u}_2\} \tag{72}$$

which ensure constant thrust acceleration during thrusting arcs. These constraints are easily implemented with the FSCT method by prespecifying the control values between switching times (knots) according to the control constraint. In this example, let $n_n = 5$ nodes per segment and $n_k = 14$ knots per control axis. In addition, let the prespecified controls be identified as

$$u^*_{i,k} = \tilde{u}_i \cos\left[\frac{\pi}{2}(k-1)\right] \tag{73}$$

such that the control structure resembles that of Fig. 2. Thus, it is assumed a priori through the control sequence that the vehicle will thrust initially in the positive directions (uprange and up), then coast, then thrust in the negative directions (downrange and down). The resulting optimizations determine the appropriate times for all control switches, indicating the durations for each thrusting and coasting arc.

An initial guess is devised with $t_0 = 0$, $t_f = 300$ s, and all knot times are evenly distributed over the interval such that each segment duration is identical. The state parameters in $\boldsymbol{x}$ are constructed to create a linear progression in each state from its initial value to its final value. Initial, final, and knot condition constraints are satisfied by the $\boldsymbol{x}$ supplied to the NLP algorithm before the first iteration, but continuity constraints are not immediately satisfied. During the optimization process, $\boldsymbol{x}$ is improved such that all constraints are satisfied. In addition, the final $\boldsymbol{x}$ minimizes the objective function, representing $J = t_f - t_0$ for minimum time or

$$J = \int_{t_0}^{t_f} \boldsymbol{u}^T \boldsymbol{u} \mathrm{d}t$$

for minimum acceleration.

Figure 4 displays the solutions of both the minimum time and minimum acceleration problem. Vehicle positions and controls are plotted for both minimizations in Figs. 4a and 4b. Notice the control history u_1 for the minimum time solution. In essence, this solution represents bang–bang control in the first axis, with $u_1(t) = -\tilde{u}_1 = -50$ m/s^2 on $t \in [0 \quad 33.66]$ s and $u_1(t) = \tilde{u}_1$ for the remaining time until t_f at 101.32 s. Of course, this control behavior is expected for a minimum time optimization. Recall, however, that the prespecified initial value for u_1 is $\tilde{u}_1$. As the illustration demonstrates, there is

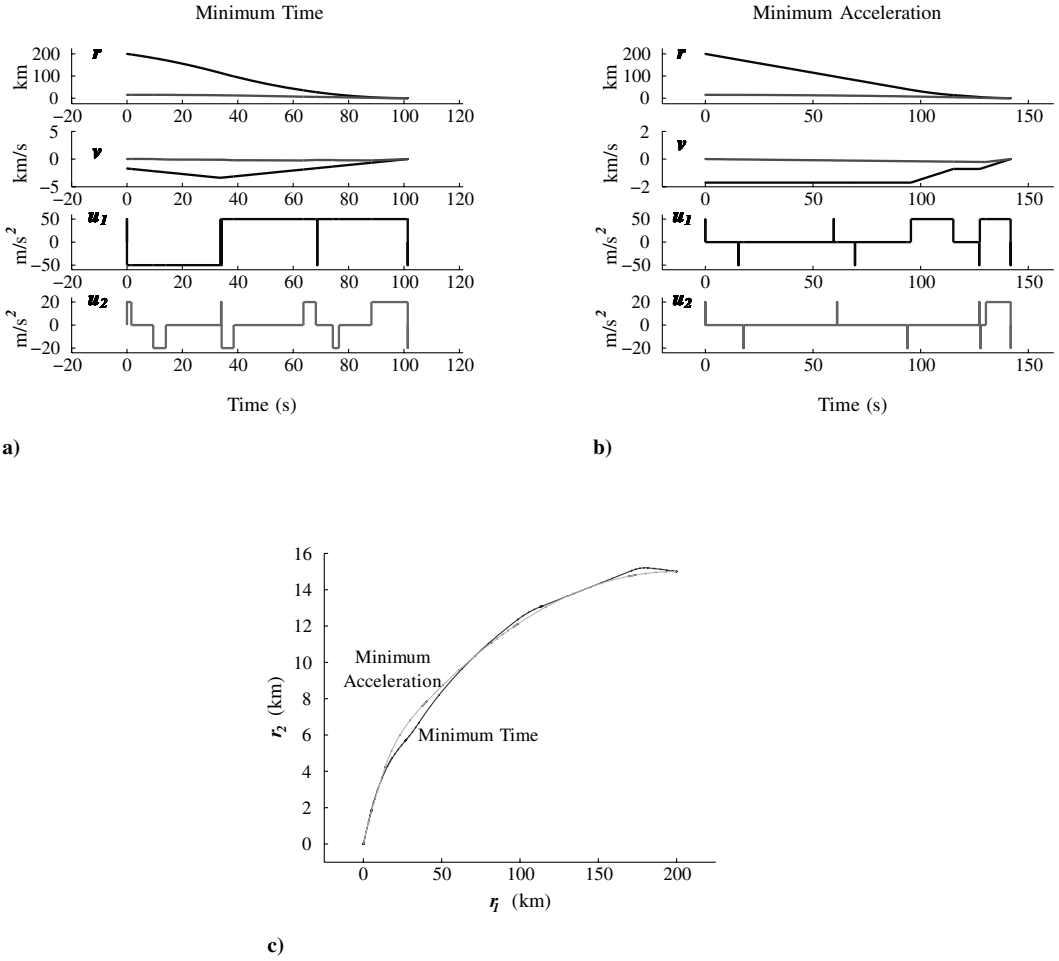

Fig. 4 Optimal solutions for the minimum time and minimum acceleration lunar lander problem.

an instantaneous switch in the control at $t_0 = 0$ from $\tilde{u}_1$ to 0 and then from 0 to $-\tilde{u}_1$. The solution exhibits that $\Delta t_{1,1} = \Delta t_{1,2} = 0$ in order to accomplish this. In addition, there are instantaneous switches at $t = 34.06$, 68.65, and 101.32 s. At each of these times, there exists time durations $\Delta t_{1,k}$ for coasting and negative-direction thrusting, and each has been optimized to be identically zero. This behavior is a common artifact of the FSCT formulation. It does not indicate that control switches should occur at these times; rather, it indicates that the problem has been overparameterized with more knots than necessary. However, because control values are prespecified in the optimization, it is useful to overparameterize the problem, allowing for more control switches than needed. Overparameterizing allows the NLP algorithm to demonstrate the optimal number of switches (less than the parameterized number) by driving to zero superfluous control axis durations. The overparameterization also allows the user additional flexibility to arbitrarily prespecify control values, knowing that nonoptimal control values are eliminated in the final solution. In this case, specifying $n_k = 14$ knots represented an overparameterization in u_1 but not necessarily in u_2. In the vertical control axis, only three time durations are driven to zero by the optimization. These are the positive thrusting arc occurring at $t = 33.93$ s and the coasting and negative thrusting arcs occurring simultaneously at the final time.

This same behavior is observed for the minimum acceleration optimization displayed in Fig. 4b. One may easily observe that most thrusting arcs are reduced to identically zero by the NLP algorithm for both control axes. This indicates that far fewer switches were necessary to identify this local minimum, and it provides confidence that the formulation has not underparameterized the problem by providing too few control switching opportunities.

Figure 4c plots the minimum time and minimum acceleration trajectories concurrently. For each trajectory, the dots indicate the actual values of the state parameters optimized in $\boldsymbol{x}$. Thus, these are the states at the nodes along each segment. It is clear that the nodes are not evenly distributed spatially nor are they distributed evenly in time. Again, this is due to the varying durations of each segment. Regardless, there are $n_n = 5$ nodes on each segment and, by segment, they are evenly distributed in time. The lines between the nodes are not simply a connection of the dots. Rather, the lines indicate a propagation of the initial conditions along with their respective control solutions using a variable-step integrator. Although the notions of overparameterization imply that enough knots are included in the parameterization, this illustration indicates the sufficiency of the node count, as the integrated trajectory matches nearly identically to the distributed nodes. The consistency between node locations and propagated states demonstrates the accuracy of the Hermite–Simpson integration equations. At the final time, state errors for both optimization solutions are $\mathcal{O}(10^{-6})$ m in positions and $\mathcal{O}(10^{-10})$ m/s in velocities, only slightly larger than the integration tolerances for the propagation.

An important discovery from the lunar lander example is the extent by which the FSCT method results in implementable control solutions. First, it is clear that the solution requires some interpretation. Superfluous control switches must be discounted before implementing the control history. Actuators with a minimum on times do not support thrust durations approaching zero; however, within the tolerance of the optimization, zero or near-zero burn durations actually indicate that the respective actuation is not desirable. Clearly, an optimization must be scaled properly in time to differentiate short actuation times from nonoptimal control sequences.

Second, once a control solution is adequately interpreted, the performance of the solution in a continuous-time setting can be nearly identical. Although this collocation technique does rely on a

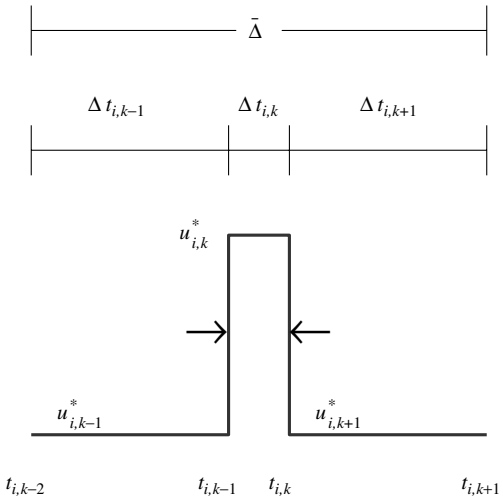

Fig. 5 Scenario resulting in multiple optimal solutions.

time discretization along each segment, the switching times between control values are optimized over a continuous domain. Therefore, the control solution represents exact switching times within the tolerance of the optimization. In this example, the continuous-time system is simulated with state and control propagation, showing negligible deviations from the FSCT solution.

One final observation regarding overparameterization is now explored. Consider the case in which the optimal solution requires that $\Delta t_{i,k} = 0$ for some i and k. Thus, $t_{i,k-1} = t_{i,k}$ and the prespecified control value $u^*_{i,k}$ are not part of the optimal solution. In addition, let

$$u^*_{i,k-1} = u^*_{i,k+1} \tag{74}$$

This is a common situation, as many applications may prespecify control values to alternate between two values (or on and off). In this case, the control effectively remains at the value of $u^*_{i,k-1}$ from $t_{i,k-2}$ to $t_{i,k+1}$ (see Fig. 5). Define $\bar{\Delta}$ as the total duration spent at this control value, such that

$$\bar{\Delta} = t_{i,k+1} - t_{i,k-2} = \Delta t_{i,k-1} + \Delta t^{\nearrow 0}_{i,k} + \Delta t_{i,k+1} \tag{75}$$

Notice that $\Delta t_{i,k-1}$ and $\Delta t_{i,k+1}$ can take any values, such that their sum remains $\bar{\Delta}$ while still ultimately communicating the same control profile.

Thus, there are an infinite number of combinations of parameters that represent an identical optimal solution. However, it is observed that changes to the two nonzero time durations have significant impact on the dynamic constraints, as the knot locations determined through $\Delta t_{i,k-1}$ and $\Delta t_{i,k+1}$ further determine the node locations at which states are defined in $\boldsymbol{x}_y$. This inertia deters arbitrary variations in time parameters, thus facilitating convergence. In other words, minor changes in the values of $\Delta t_{i,k-1}$ and $\Delta t_{i,k+1}$ require major changes in the elements of $\boldsymbol{Y}_{n_y,n_n,n_s}$ on the three corresponding segments in order to realize an equivalent optimal solution.

V. Conclusions

The continuous-time hybrid optimal control problem, exhibiting continuous state variables and spatially discrete control variables, is the subject of this investigation. Traditionally, this type of problem is formulated in the context of mixed-integer optimization and sometimes solved using strictly NLP methods. However, these earlier formulations do not adequately treat the problem of asynchronous switching across multiple spatially discrete control variables, each subject to a unique discrete range. Although the present investigation also employs a strictly NLP approach, the fundamental difference between this work and earlier efforts is a novel transcription formulation that effectively treats the asynchronous switching problem. The approach is termed the FSCT method. In contrast to earlier formulations that also employ a strictly NLP approach, the FSCT method significantly enhances the computational efficiency of the nonlinear program by reducing the size of the underlying parameter optimization problem.

The present work is devoted to the theoretical development of the FSCT method and discussion of special considerations regarding its numerical implementation in a nonlinear program. The effectiveness of the method is demonstrated with a simple aerospace engineering application. Specifically, a lunar lander for which the thrusters are each independently constrained to a unique and finite set of thrust levels. Each thruster can switch between their allowed levels independently of the others. This gives rise to asynchronous switching. Although this example is relevant specifically to aerospace engineering, the FSCT is formulated in a generalized form and is thus useful in any application involving multiple spatially discrete variables for which asynchronous switching is desirable.

Acknowledgment

The views expressed in this article are those of the author and do not reflect the official policy or position of the U.S. Air Force, the U.S. Department of Defense, or the U.S. Government.

References

[1] Branicky, M., Borkar, V., and Mitter, S., "A Unified Framework for Hybrid Control: Model and Optimal Control Theory," *IEEE Transactions on Automatic Control*, Vol. 43, No. 1, Jan 1998, pp. 31–45. doi:10.1109/9.654885

[2] Branicky, M., "Stability of Switched and Hybrid Systems," *Proceedings of the 33rd Conference on Decision and Control*, IEEE, Piscataway, NJ, 1994, pp. 3498–3503.

[3] Pettersson, S., and Lennartson, B., "Controller Design of Hybrid Systems," *Hybrid and Real-Time Systems*, Lecture Notes in Computer Science, Vol. 1201, Springer–Verlag, Berlin, Mar 1997, pp. 240–254. doi:10.1007/BFb0014706

[4] Branicky, M., "Multiple Lyapunov Functions and Other Analysis Tools for Switched and Hybrid Systems," *IEEE Transactions on Automatic Control*, Vol. 43, No. 4, Apr 1998, pp. 475–482. doi:10.1109/9.664150

[5] Sager, S., Bock, H. G., Diehl, M., Reinelt, G., and Schlöder, J. P., "Numerical Methods for Optimal Control with Binary Control Functions Applied to a Lotka–Volterra Type Fishing Problem," *Recent Advances in Optimization*, Springer–Verlag, Berlin, 2006, pp. 269–289. doi:10.1007/3-540-28258-0

[6] Stryk, O., and Glocker, M., "Numerical Mixed-Integer Optimal Control and Motorized Traveling Salesmen Problems," *European Journal of Control*, Vol. 35, No. 4, 2001, pp. 519–533.

[7] Gerdts, M., "Solving Mixed-Integer Optimal Control Problems by Branch and Bound: A Case Study From Automobile Test-Driving With Gear Shift," *Optimal Control Applications and Methods*, Vol. 26, No. 1, 2005, pp. 1–18. doi:10.1002/oca.751

[8] Gerdts, M., "A Variable Time Transformation Method for Mixed-Integer Optimal Control Problems," *Optimal Control Applications and Methods*, Vol. 27, No. 3, 2006, pp. 169–182. doi:10.1002/oca.778

[9] Floudas, C. A., *Nonlinear and Mixed-Integer Optimization*, Oxford Univ. Press, New York, 1995, pp. 98–107.

[10] Geoffrion, A. M., "Generalized Benders Decomposition," *Journal of Optimization Theory and Applications*, Vol. 10, No. 4, Oct 1972, pp. 237–260. doi:10.1007/BF00934810

[11] Duran, M. A., and Grossmann, I. E., "An Outer-Approximation Algorithm for a Class of Mixed-Integer Nonlinear Programs," *Mathematical Programming*, Vol. 36, No. 3, Oct 1986, pp. 307–339. doi:10.1007/BF02592064

[12] Fletcher, R., and Leyffer, S., "Solving Mixed Integer Nonlinear Programs by Outer Approximation," *Mathematical Programming*, Vol. 66, Nos. 1–3, Aug 1994, pp. 327–349. doi:10.1007/BF01581153

[13] Allgor, R. J., and Barton, P., "Mixed-Integer Dynamic Optimization 1: Problem Formulation," *Computers and Chemical Engineering*, Vol. 23, Nos. 4–5, 1999, pp. 567–584. doi:10.1016/S0098-1354(98)00294-4

[14] Lee, H. W. J., Teo, K. L., Rehbock, V., and Jennings, L. S., "Control

Parametrization Enhancing Technique for Optimal Discrete-Valued Control Problems," *Automatica*, Vol. 35, No. 8, Aug 1999, pp. 1401–1407.
doi:10.1016/S0005-1098(99)00050-3

[15] Wei, S., Uthaichana, K., Žefran, M., DeCaralo, R. A., and Bengea, S., "Applications of Numerical Optimal Control to Nonlinear Hybrid Systems," *Nonlinear Analysis: Hybrid Systems*, Vol. 1, No. 2, 2007, pp. 264–279.
doi:10.1016/j.nahs.2006.10.007

[16] 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, Jul–Aug 1987, pp. 338–342.
doi:10.2514/3.20223

[17] Betts, J. T., "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, Mar–Apr 1998, pp. 193–207.
doi:10.2514/2.4231

[18] Betts, J. T., *Practical Methods for Optimal Control Using Nonlinear Quadratic Programming*, Society of Industrial and Applied Mathematics, Philadelphia, PA, 2001, pp. 89–92.

[19] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Journal on Optimization*, Vol. 12, No. 4, 2002, pp. 979–1006.
doi:10.1137/S1052623499350013

[20] Lukšan, L., and Spedicato, E., "Variable Metric Methods for Unconstrained Optimization and Nonlinear Least Squares," *Journal of Computational and Applied Mathematics*, Vol. 124, Nos. 1–2, Dec 2000, pp. 61–95.
doi:10.1016/S0377-0427(00)00420-9

[21] Pourshaghaghy, A., Kowsary, F., and Behbahaninia, A., "Comparison of Four Different Versions of the Variable Metric Method for Solving Inverse Heat Conduction Problems," *Heat and Mass Transfer*, Vol. 43, No. 3, Jan. 2007, pp. 285–294.
doi:10.1007/s00231-006-0107-9

[22] Bryson, A. E., and Ho, Y.-C., *Applied Optimal Control: Optimization, Estimation, and Control*, Taylor and Francis, Washington, D.C., 1975, pp. 59–63.

[23] Hull, D. G., *Optimal Control Theory for Applications*, Springer–Verlag, New York, 2003, p. 246.