Hindawi Publishing Corporation
Abstract and Applied Analysis
Volume 2014, Article ID 851720, 15 pages
http://dx.doi.org/10.1155/2014/851720

Review Article

Survey of Direct Transcription for Low-Thrust Space Trajectory Optimization with Applications

F. Topputo[1] and C. Zhang[2]

[1] *Politecnico di Milano, 20156 Milan, Italy*
[2] *Beijing University of Aeronautics and Astronautics, Beijing 100191, China*

Correspondence should be addressed to F. Topputo; francesco.topputo@polimi.it

Received 20 February 2014; Accepted 16 May 2014; Published 24 June 2014

Academic Editor: Ryan Loxton

Copyright © 2014 F. Topputo and C. Zhang. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Space trajectory design is usually addressed as an optimal control problem. Although it relies on the classic theory of optimal control, this branch possesses some peculiarities that led to the development of ad hoc techniques, which can be grouped into two categories: direct and indirect methods. This paper gives an overview of the principal techniques belonging to the direct methods. The technique known as "direct transcription and collocation" is illustrated by considering Hermite-Simpson, high-order Gauss-Lobatto, and pseudospectral methods. Practical examples are given, and several hints to improve efficiency and robustness are implemented.

1. Introduction

In January 1959, the Soviet Union launched Luna 1, a lunar probe that became the first spacecraft placed in heliocentric orbit (http://en.wikipedia.org/wiki/Luna_1). This event marked a new era for deep space exploration. In over half a century, various probes visited the major planets of the Solar System, and they provided a remarkable scientific return. Nowadays, many space agencies have established ambitious space programs, whose accomplishment stimulated the development of new technologies.

Electric propulsion is one of those technologies that improve the efficiency of space transport. Electric motors have a specific impulse that is approximately ten times that of the chemical engines, and therefore they allow a considerable saving of propellant mass, which in turn makes it possible to embark heavier instruments, so increasing the scientific return of a mission. Electric propulsion also involves long working hours, extended launch windows, and flexible and precise control abilities. Electric propulsion has been used in NASA's Deep Space 1 in 1998 [1], in JAXA's Hayabusa in 2003 [2], and in ESA's SMART-1 in 2003 [3]. Many future space missions foresee the use of this technology. If compared to that of the chemical engines, electric propulsion produces low levels of thrust, which cannot be modelled as producing instantaneous velocity changes [4]. This is also the case of solar sails, an emerging and prominent technology demonstrated by the recent success of JAXA's IKAROS [5] and NASA's NanoSail-D2 [6]. Solar sails are not propellant-constrained, and the effect of the solar radiation pressure on a large surface can be modelled as producing a low-thrust acceleration [7, 8]. The combination of low-thrust propulsion with gravity assists is among the most promising techniques for deep space explorations, which can be utilized to save fuel and shorten the flight time [9–12].

Low-thrust space trajectories are studied as a specialization of the optimal control problem for continuous time systems. Unfortunately, no analytic solutions exist for this problem even under the simple two-body dynamics. Thus, numerical methods must be used. Low-thrust trajectory optimization involves determining the control law (thrust magnitude and direction) and the associated transfer orbit while minimizing a given performance index (propellant mass or time-of-flight) and satisfying boundary conditions

(departure and arrival orbits), midpoint conditions (at patching points), and path constraints (thrust saturation). In this branch of optimal control, two main solution methods have been devised. Within direct methods, the state and control variables are discretized, and the optimal control problem is converted into a nonlinear programming (NLP) problem [13, 14]. This process is called direct transcription. Direct methods are generally robust and can easily accommodate path constraints, but they often require much computational effort especially for multispiral trajectories. Indirect methods rely on the calculus of variations. The necessary conditions of optimality require the solution of a two-point boundary value problem (TPBVP). This method ensures rapid convergence of good starting guesses, but most of the difficulties are related to the small convergence radius, the high sensitivity to the initial costates, and their lack of physical meaning.

This paper complements previous surveys on optimal control of space trajectories [15–18] by focussing on direct methods. In particular, the direct transcription and collocation is studied. The collocation schemes considered implementing different integration methods (Hermite-Simpson, high-order Gauss-Lobatto, and pseudospectral methods). These techniques are implemented to solve practical examples, and the behavior of each method is assessed against the implementation of several issues aimed at improving the computational efficiency and the methods robustness.

The remainder of the paper is organized as follows. In Section 2, the optimal control problem in space flight mechanics is stated, and the NLP problem as well as the direct transcription concept is recalled. In Section 3 direct transcription and collocation is treated. Computational issues are discussed in Section 4 and practical cases are solved in Section 5. Concluding remarks are given in Section 6.

2. The Optimal Control Problem in Space Flight Mechanics

Given a set of n first-order differential equations

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t), \tag{1}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is a vector of state variables, $\mathbf{u}(t) \in \mathbb{R}^m$ denotes vector of control variables, and t represents the independent time variable, $t \in [t_i \quad t_f]$, the following performance index has to be minimized

$$J = \varphi\left(\mathbf{x}\left(t_f\right), t_f\right) + \int_{t_i}^{t_f} L(\mathbf{x}, \mathbf{u}, t)\, \mathrm{d}t, \tag{2}$$

while satisfying q-dimensional final boundary conditions

$$\boldsymbol{\psi}\left(\mathbf{x}\left(t_f\right), \mathbf{u}\left(t_f\right), t_f\right) = 0. \tag{3}$$

The solution to this problem is derived by the calculus of variations, which leads to the derivation of the Euler-Lagrange equations. Let $\boldsymbol{\nu}$ be a q-dimensional constant vector of multipliers of the final boundary constraints, and let $\boldsymbol{\lambda}$ be the n-dimensional variable vector of adjoint or costate multipliers of the dynamics. The augmented performance index is defined as

$$\begin{aligned}\bar{J} = {} & \varphi\left(\mathbf{x}\left(t_f\right), t_f\right) + \boldsymbol{\nu}^T \boldsymbol{\psi}\left(\mathbf{x}\left(t_f\right), \mathbf{u}\left(t_f\right), t_f\right) \\ & + \int_{t_i}^{t_f} \left[L(\mathbf{x}, \mathbf{u}, t) + \boldsymbol{\lambda}^T (\mathbf{f}(\mathbf{x}, \mathbf{u}, t) - \dot{\mathbf{x}})\right] \mathrm{d}t.\end{aligned} \tag{4}$$

The augmented performance index (4) embeds the dynamics (1) as well as the final boundary conditions (3). The problem consists in deriving the necessary conditions for a stationary point of $\bar{J}$ [19]. This is achieved by imposing that its first variation is zero, namely, $\delta\bar{J} = 0$. In order to write the necessary conditions in a compact form, it is convenient to define the Hamiltonian

$$H(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{u}, t) = L(\mathbf{x}, \mathbf{u}, t) + \boldsymbol{\lambda}^T \mathbf{f}(\mathbf{x}, \mathbf{u}, t). \tag{5}$$

The necessary conditions for optimality, also referred to as the Euler-Lagrange equations [20, 21], are

$$\dot{\mathbf{x}} = H_\lambda, \qquad \dot{\boldsymbol{\lambda}} = -H_x, \qquad 0 = H_u, \tag{6}$$

where the subscript denotes partial derivation. The first of (6) is equivalent to (1), the second describes the dynamics of the costates, and the third is an algebraic equation for the control functions. This differential-algebraic system must be solved together with the final boundary conditions (3) and the following transversality conditions:

$$\boldsymbol{\lambda}\left(t_f\right) = \left[\varphi_{\mathbf{x}} + \boldsymbol{\nu}^T \boldsymbol{\psi}_{\mathbf{x}}\right]_{t=t_f}. \tag{7}$$

With $\mathbf{x}(t_i) = \mathbf{x}_i$ given, the problem represents a two-point boundary value problem. The last of (6) is an application of the Pontryagin maximum principle [22]. A more general expression is in fact

$$\mathbf{u} = \arg\min_{\mathbf{u} \in U} H(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{u}, t), \tag{8}$$

where U defines the domain of feasible controls. The maximum principle states that the control variables must be chosen to optimize the Hamiltonian at every instant of time: the solution of the optimal control problem is an extremum for H. In essence, the maximum principle is a constrained optimization problem in the function $\mathbf{u}(t)$ at all values of t.

2.1. The Optimal Trajectory Design Problem. An optimal trajectory design problem is a specialization of the classical optimal control problem above. In space flight mechanics, the equations of motion have the following form [16]:

$$\dot{\mathbf{x}} = \begin{Bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \end{Bmatrix} = \begin{Bmatrix} \mathbf{v} \\ \mathbf{g}(\mathbf{r}) + a_c \hat{\mathbf{u}} \end{Bmatrix}. \tag{9}$$

In (9), $\mathbf{r}$ and $\mathbf{v}$ are the spacecraft position and velocity vectors, respectively, $\mathbf{g}(\mathbf{r})$ is the gravitational vector field, a_c is the thrust acceleration magnitude, and $\hat{\mathbf{u}}$ is the thrust direction unit vector. The control variables are a_c and $\hat{\mathbf{u}}$. The control acceleration magnitude is upper bounded for technological

reasons; that is, $0 \le a_c \le a_c^{\max}$. To minimize the total velocity change, and therefore the propellant mass, the objective function (2) is such that $L = a_c$, $\varphi = 0$. The Hamiltonian (5) is, therefore,

$$\begin{aligned} H &= a_c + \boldsymbol{\lambda}_r \cdot \mathbf{v} + \boldsymbol{\lambda}_v \cdot \left[\mathbf{g}(\mathbf{r}) + a_c \hat{\mathbf{u}}\right] \\ &= a_c \left[1 + \boldsymbol{\lambda}_v \cdot \hat{\mathbf{u}}\right] + \boldsymbol{\lambda}_r \cdot \mathbf{v} + \boldsymbol{\lambda}_v \cdot \mathbf{g}(\mathbf{r}), \end{aligned} \tag{10}$$

where $\boldsymbol{\lambda}_r$ and $\boldsymbol{\lambda}_v$ are the costate vectors associated to the position and velocity vectors, respectively. As a_c is bounded, the last of (6) cannot be applied, and (8) has to be used instead. Since the Hamiltonian has to be minimized at any time, the following observations can be made.

(1) The thrust unit vector $\hat{\mathbf{u}}$ has to be parallel and opposite to $\boldsymbol{\lambda}_v$. That is, $\hat{\mathbf{u}} = -\alpha \boldsymbol{\lambda}_v$ with $\alpha > 0$; this explains why $\boldsymbol{\lambda}_v$ is also referred to as the *primer vector*.

(2) The thrust magnitude a_c has to be chosen according to the sign of the *switching function:*

$$S = 1 + \boldsymbol{\lambda}_v \cdot \hat{\mathbf{u}}. \tag{11}$$

In particular, $a_c = a_c^{\max}$ when $S < 0$ and $a_c = 0$ when $S > 0$. This makes the function $a_c(t)$ have a *bang-bang* structure; that is, it is piece-wise discontinuous and it is either zero or maximum. This property is of great importance to evaluate the optimal control profile a posteriori.

The necessary conditions only guarantee that the optimal trajectory is an extremum for the Hamiltonian. Thus, to assess the optimality of the solutions, one is supposed to check the second-order conditions. However, due to the nature of the space trajectory problem, there is no upper bound to the propellant that can be consumed in one trajectory, so one may be confident that a solution that satisfies the necessary conditions is a local minimum and not a local maximum [16].

In trajectory optimization problems, the initial state, $\mathbf{x}(t_i)$, is generally given. If the initial costate, $\boldsymbol{\lambda}(t_i)$, was given as well, the optimal solution could be obtained by integrating the Euler-Lagrange equations with an implicit "bang-bang" thrusting structure [23]. Thus, the low-thrust optimal trajectory design can be converted into a TPBVP, which consists in finding the unknown initial costate vector. This is the essence of indirect methods. Another philosophy consists instead in translating the continuous optimal control problem into a NLP problem and solving for a finite set of variables [15, 24–27]. This procedure is the direct transcription and the approach is said direct method.

2.1.1. The General Optimal Trajectory Design Problem. It is convenient to state the optimal trajectory design problem in a more general fashion, which copes with the direct approach. The dynamics are let to incorporate a number of constant parameters $\mathbf{p}$; that is,

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t), \tag{12}$$

and initial and final conditions can be defined within some prescribed lower and upper bounds

$$\begin{aligned} \boldsymbol{\psi}_{i,l} \le \boldsymbol{\psi}_i(\mathbf{x}(t_i), \mathbf{u}(t_i), \mathbf{p}, t_i) \le \boldsymbol{\psi}_{i,u}, \\ \boldsymbol{\psi}_{f,l} \le \boldsymbol{\psi}_f(\mathbf{x}(t_f), \mathbf{u}(t_f), \mathbf{p}, t_f) \le \boldsymbol{\psi}_{f,u}. \end{aligned} \tag{13}$$

In addition, the solution can be subject to path constraints of the form

$$\mathbf{g}_l \le \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t) \le \mathbf{g}_u, \tag{14}$$

as well as simple bounds on the state variables

$$\mathbf{x}_l \le \mathbf{x}(t) \le \mathbf{x}_u, \tag{15}$$

and on the control variables

$$\mathbf{u}_l \le \mathbf{u}(t) \le \mathbf{u}_u. \tag{16}$$

The basic problem is to determine the control vectors $\mathbf{u}(t)$ to minimize the performance index

$$J = \phi(\mathbf{x}(t_f), t_f), \tag{17}$$

which is written in the Mayer form [15].

2.2. The Nonlinear Programming Problem. Essentially, any numerical method for solving the trajectory optimization problem incorporates some type of Newton method to solve for a finite set of unknowns. In Section 3 it is shown how an optimal control problem can be transformed into a NLP problem [16, 27]. A NLP problem is a decisional problem concerning a scalar objective function and a vector of constraints. As opposite to the optimal control problem, no dynamics is involved in a NLP problem. Suppose that the n variables $\mathbf{x}$ must be chosen to solve

$$\min_{\mathbf{x}} F(\mathbf{x}), \tag{18}$$

subject to the m equality constraints

$$\mathbf{c}(\mathbf{x}) = 0, \tag{19}$$

where $m \le n$. The Lagrangian of this problem is

$$L(\mathbf{x}, \boldsymbol{\lambda}) = F(\mathbf{x}) - \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{x}), \tag{20}$$

which is a scalar function of the n variables $\mathbf{x}$ and the m Lagrange multipliers $\boldsymbol{\lambda}$. The necessary conditions for a point $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ to be a constrained optimum require solving the following system:

$$\begin{aligned} \nabla_x L(\mathbf{x}, \boldsymbol{\lambda}) &= \mathbf{g}(\mathbf{x}) - \mathbf{G}^T(\mathbf{x}) \boldsymbol{\lambda} = 0, \\ \nabla_\lambda L(\mathbf{x}, \boldsymbol{\lambda}) &= -\mathbf{c}(\mathbf{x}) = 0, \end{aligned} \tag{21}$$

where $\mathbf{g} = \nabla_x F$ and $\mathbf{G}$ are the gradient of the objective function and the Jacobian of the equality constraint vector, respectively. The system (21) can be solved via a Newton method to find the $(n+m)$ variables $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$. Given a generic initial guess $(\mathbf{x}, \boldsymbol{\lambda})$, its corrections $(\Delta\mathbf{x}, \Delta\boldsymbol{\lambda})$ to construct the

new solution $(\mathbf{x} + \Delta\mathbf{x}, \boldsymbol{\lambda} + \Delta\boldsymbol{\lambda})$ are given by solving the linear system

$$\begin{bmatrix} \mathbf{H}_L & -\mathbf{G}^T \\ \mathbf{G} & 0 \end{bmatrix} \begin{Bmatrix} \Delta\mathbf{x} \\ \Delta\boldsymbol{\lambda} \end{Bmatrix} = \begin{Bmatrix} -\mathbf{g} \\ -\mathbf{c} \end{Bmatrix}, \tag{22}$$

also referred to as Karush-Kuhn-Tucker (KKT) system. In (22), $\mathbf{H}_L$ is the Hessian of (20) in x; namely,

$$\mathbf{H}_L = \nabla_x^2 F - \sum_{i=1}^{m} \lambda_i \nabla_x^2 c_i. \tag{23}$$

It is important to observe that an equivalent way to define the search direction $\Delta\mathbf{x}$ is to minimize the quadratic form

$$\frac{1}{2}\Delta\mathbf{x}^T \mathbf{H}_L \Delta\mathbf{x} + \mathbf{g}^T \Delta\mathbf{x} \tag{24}$$

subject to the linear constraints

$$\mathbf{G}\Delta\mathbf{x} = -\mathbf{c}. \tag{25}$$

This is the reason why this problem is also referred to as a quadratic programming (QP) problem.

The NLP problem formulated above can be generalized to the case that occur when inequality constraints are imposed; the m constraints are of the form

$$\mathbf{c}(\mathbf{x}) \geq 0. \tag{26}$$

Constraints that are strictly satisfied, that is, those for which $c_i(\mathbf{x}) > 0$, are called inactive; the remaining active set of constraints are on their bounds; that is, $c_j(\mathbf{x}) = 0$. If the active set of constraints is known, the inactive constraints are ignored and the problem is simply solved using the method for an equality constrained problem discussed above.

In summary, the general NLP problem requires finding the n vectors to solve

$$\min_{\mathbf{x}} F(\mathbf{x}), \tag{27}$$

subject to the m constraints

$$\mathbf{c}_L \leq \mathbf{c}(\mathbf{x}) \leq \mathbf{c}_U, \tag{28}$$

and bounds

$$\mathbf{x}_L \leq \mathbf{x} \leq \mathbf{x}_U. \tag{29}$$

In this formulation equality constraints can be imposed by setting $c_{j,L} = c_{j,U}$.

2.3. Direct Transcription. With a direct approach, the solution to the optimal control problem is strictly connected to the numerical integration of the differential equations. The core of this method consists in the way the dynamics are handled; the set of differential equations governing the motion of a spacecraft can be transcribed into a finite set of equality constraints. If these are respected, then the original optimal trajectory design problem is solved within the degree of accuracy of the numerical scheme used [16, 18, 27–30].

In problem (12)–(17), the time domain can be uniformly discretized as

$$t_i = t_1 < t_2 < \cdots < t_N = t_f, \tag{30}$$

where the time labels are referred to as mesh points or nodes; t_i and t_f are the initial and final time, respectively, and $h = (t_N - t_1)/(N-1)$ is the fixed step size of the discretization (a nonuniform time discretization is also possible). The states and the controls can be discretized over the mesh (30) by defining $\mathbf{x}_k = \mathbf{x}(t_k)$ and $\mathbf{u}_k = \mathbf{u}(t_k)$. The discretized states and the controls are now ready to be treated as a set of NLP variables. The whole variable vector of the problem is

$$\mathbf{y} = \{\mathbf{x}_1, \mathbf{u}_1, \ldots, \mathbf{x}_N, \mathbf{u}_N\}^T. \tag{31}$$

The differential equations are replaced by a finite set of defects constraints derived by the numerical integration scheme. For instance, if a forward Euler scheme is used, the defects are of the form

$$\boldsymbol{\zeta}_k \equiv \mathbf{x}_{k+1} - \mathbf{x}_k - h\mathbf{f}_k, \tag{32}$$

where $\mathbf{f}_k = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{p}, t_k)$. As a result of the transcription, the optimal control constraints (13)-(14) are replaced by the NLP constraints

$$\mathbf{c}_L \leq \mathbf{c}(\mathbf{y}) \leq \mathbf{c}_U, \tag{33}$$

where

$$\mathbf{c}(\mathbf{y}) \equiv \{\boldsymbol{\zeta}_1, \boldsymbol{\zeta}_2, \ldots, \boldsymbol{\zeta}_{N-1}, \boldsymbol{\psi}_1, \boldsymbol{\psi}_N, \mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_N\}^T, \tag{34}$$

with $\boldsymbol{\psi}_1 = \boldsymbol{\psi}_i$, $\boldsymbol{\psi}_N = \boldsymbol{\psi}_f$, and

$$\begin{aligned} \mathbf{c}_L &\equiv \{\mathbf{0}, \ldots, \mathbf{0}, \mathbf{g}_{1,L}, \ldots, \mathbf{g}_{N,L}\}^T, \\ \mathbf{c}_U &\equiv \{\mathbf{0}, \ldots, \mathbf{0}, \mathbf{g}_{1,U}, \ldots, \mathbf{g}_{N,U}\}^T. \end{aligned} \tag{35}$$

The first $n(N-1)$ equality constraints in (34) require that the defect vectors $\boldsymbol{\zeta}_k$, $k = 1, \ldots, N-1$, are zero, thereby they satisfy the differential equations (12) within the accuracy of the numerical integration. The boundary conditions (13) are enforced directly by the equality constraints on $\boldsymbol{\psi}_1$ and $\boldsymbol{\psi}_N$, and the nonlinear path constraints (14) are imposed at the grid points. In a similar fashion the objective function, either in the form (2) or (17), can be written in terms of $\mathbf{y}$; namely, $F = F(\mathbf{y})$. The optimal trajectory design problem is so translated into the forms (27)–(29) and it can be solved as a standard NLP problem through (18)–(23).

In this derivation, the simple forward Euler scheme is used. More accurate methods are likely to be used for practical applications. This is done to keep N within reasonable values, avoiding out-of-memory problems. The integration schemes influence the robustness of the method and the solution accuracy. In Section 3, several implicit quadrature methods are shown.

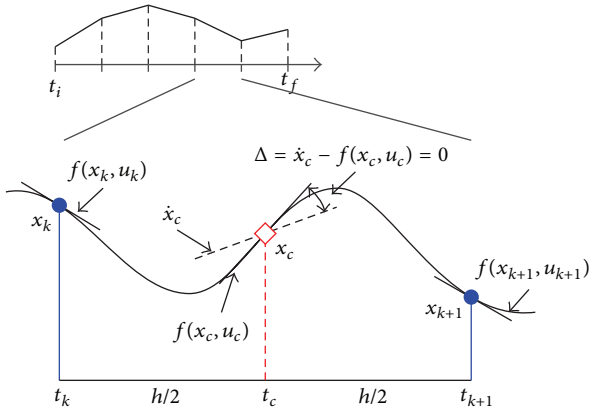


Figure 1: Hermite-Simpson collocation method.

3. Direct Transcription and Collocation

Collocation is used to transcribe differential dynamic constraints into a set of algebraic constraints. The basic idea is to choose a polynomial up to a certain degree with a number of points in the time domain (collocation points), and to enforce the polynomials to satisfy the equations of motion at the collocation points. The peculiarity of each collocation method relies on the way the state and control variables are discretized and how the dynamic constraints are satisfied [17].

3.1. Hermite-Simpson Method. A basic form of collocation is the Hermite-Simpson method [24], illustrated in Figure 1. For each of the segments $[t_k, t_{k+1}]$, the two end points, denoted as "nodes" (blue dots), represent the corresponding state and control NLP variables; that is, $[x_k, u_k, x_{k+1}, u_{k+1}]$ (scalar states and controls are used from now on to ease the notation). The dynamics are used to provide time derivative values at the two nodes, so the four pieces of information $[x_k, x_{k+1}, f(x_k, u_k), f(x_{k+1}, u_{k+1})]$ can be used to construct a third-order Hermite interpolate polynomial. This interpolate polynomial cannot satisfy the equations of motion at any time within $[t_k, t_{k+1}]$, because it does that only at the nodes. Let $[x_c, u_c]$ be the state and control at t_c, the middle point of $[t_k, t_{k+1}]$; this is called "collocation point" (red diamond). Enforcing $\Delta = \dot{x}_c - f(x_c, u_c) = 0$ makes it possible to have a polynomial that not only satisfies the dynamics at the two nodes but also does that at the collocation point. If a large number of intervals are used, the state motion approaches the real dynamics within the whole time domain.

The detailed procedure can be derived as follows [16]. Let the state function $x(t)$ be represented on each segment $[t_k, t_{k+1}]$ through a cubic polynomial of the form

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3, \tag{36}$$

which yields

$$\dot{x}(t) = a_1 + 2a_2 t + 3a_3 t^2, \tag{37}$$

where $[a_0, a_1, a_2, a_3]$ are the coefficients of the polynomial. In order to simplify the argument, the time domain is transformed such that $t \in [0, h]$ (h is the time interval of the segment). Let $x(0) = x_k$, $x(h) = x_{k+1}$, $\dot{x}(0) = \dot{x}_k$, and $\dot{x}(h) = \dot{x}_{k+1}$. Evaluating (36)-(37) at $t = 0$ and $t = h$ yields

$$\begin{bmatrix} x(0) \\ \dot{x}(0) \\ x(h) \\ \dot{x}(h) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & h & h^2 & h^3 \\ 0 & 1 & 2h & 3h^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}, \tag{38}$$

which allows us to compute the four coefficients of interpolate polynomial

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{3}{h^2} & -\frac{2}{h} & \frac{3}{h^2} & -\frac{1}{h} \\ \frac{2}{h^3} & \frac{1}{h^2} & -\frac{2}{h^3} & \frac{1}{h^2} \end{bmatrix} \begin{bmatrix} x(0) \\ \dot{x}(0) \\ x(h) \\ \dot{x}(h) \end{bmatrix}. \tag{39}$$

Substituting $[a_0, a_1, a_2, a_3]$ into (36)-(37) allows us to compute the collocation point as

$$\begin{aligned} x_c &= x\left(\frac{h}{2}\right) \\ &= \frac{1}{2}(x_k + x_{k+1}) + \frac{h}{8}\left[f(x_k, u_k) - f(x_{k+1}, u_{k+1})\right], \end{aligned} \tag{40}$$

as well as its time derivative

$$\begin{aligned} \dot{x}_c &= \dot{x}\left(\frac{h}{2}\right) \\ &= -\frac{3}{2h}(x_k - x_{k+1}) - \frac{1}{4}\left[f(x_k, u_k) + f(x_{k+1}, u_{k+1})\right]. \end{aligned} \tag{41}$$

The control variable at the collocation point can be computed by simple linear interpolation; that is,

$$u_c = \frac{u_k + u_{k+1}}{2}. \tag{42}$$

The difference between the interpolated and calculated derivatives at the collocation point defines the integration defect

$$\begin{aligned} \Delta &= \dot{x}_c - f(x_c, u_c) \\ &= -\frac{3}{2h}(x_k - x_{k+1}) - \frac{1}{4}\left[f(x_k, u_k) + f(x_{k+1}, u_{k+1})\right] \\ &\quad - f(x_c, u_c) \\ &= x_k - x_{k+1} \\ &\quad + \frac{h}{6}\left[f(x_k, u_k) + 4f(x_c, u_c) + f(x_{k+1}, u_{k+1})\right]. \end{aligned} \tag{43}$$

The NLP solver will select $[x_k, u_k, x_{k+1}, u_{k+1}]$ to drive Δ to zero and in this way the interpolating polynomial will approximate the true dynamics within the accuracy of the numerical integration. Note that the last row of (43) is actually an implicit Hermite integration. Thus, if the collocation constraints are satisfied, the system is said to be "implicitly" integrated.

3.2. High Order Gauss-Lobatto Method. In the Hermite-Simpson method, only two nodes are used to construct a third-order polynomial. As a general rule of thumb, a lower number of segments can be handled if higher order integration is performed. Herman and Conway [28] demonstrated that higher order Gauss-Lobatto methods are more robust and more efficient than the lower-order Hermite-Simpson scheme. Figure 2 illustrates the collocation constraints of fifth-order Gauss-Lobatto method. Similar to Hermite-Simpson method, in each segment, six pieces of information of nodes are used to construct a fifth-order Hermite polynomial to approximate the state time history. Then the resulting interpolation polynomial is used to evaluate the states at the remaining two collocation points.

In the fifth-order method, the collocation points are [28]

$$\begin{aligned} x_1 = {} & \frac{1}{686}\left\{\left(39\sqrt{21}+231\right)x_k + 224x_c \right. \\ & + \left(-39\sqrt{21}+231\right)x_{k+1} \\ & + h\left[\left(3\sqrt{21}+21\right)f_k - 16\sqrt{21}f_c \right. \\ & \left.\left. + \left(3\sqrt{21}-21\right)f_{k+1}\right]\right\}, \\ x_2 = {} & \frac{1}{686}\left\{\left(-39\sqrt{21}+231\right)x_k + 224x_c \right. \\ & + \left(39\sqrt{21}+231\right)x_{k+1} \\ & + h\left[\left(-3\sqrt{21}+21\right)f_k + 16\sqrt{21}f_c \right. \\ & \left.\left. + \left(-3\sqrt{21}-21\right)f_{k+1}\right]\right\}, \end{aligned} \tag{44}$$

and the two collocation constraints defects to zero are

$$\begin{aligned} \Delta_1 = {} & \frac{1}{360}\left\{\left(32\sqrt{21}+180\right)x_k - 64\sqrt{21}x_c \right. \\ & + \left(32\sqrt{21}-180\right)x_{k+1} \\ & + h\left[\left(9+\sqrt{21}\right)f_k + 98f_1 + 64f_c \right. \\ & \left.\left. + \left(9-\sqrt{21}\right)f_{k+1}\right]\right\}, \\ \Delta_2 = {} & \frac{1}{360}\left\{\left(-32\sqrt{21}+180\right)x_k + 64\sqrt{21}x_c \right. \\ & + \left(-32\sqrt{21}-180\right)x_{k+1} \\ & + h\left[\left(9-\sqrt{21}\right)f_k + 98f_2 + 64f_c \right. \\ & \left.\left. + \left(9+\sqrt{21}\right)f_{k+1}\right]\right\}. \end{aligned} \tag{45}$$

A detailed derivation of the formulas above is required when the argument is extended to arbitrary higher orders. In [31], an alternative framework for unifying arbitrary higher-order methods is presented. In this approach, the Legendre-Gauss-Lobatto (LGL) discrete points ξ_j are used to improve both interpolating precision and quadrature performance. Figure 3 graphically illustrates the location of three, five, and seven LGL points. Table 1 lists the corresponding position of these LGL points (the time interval is $[-1, 1]$).

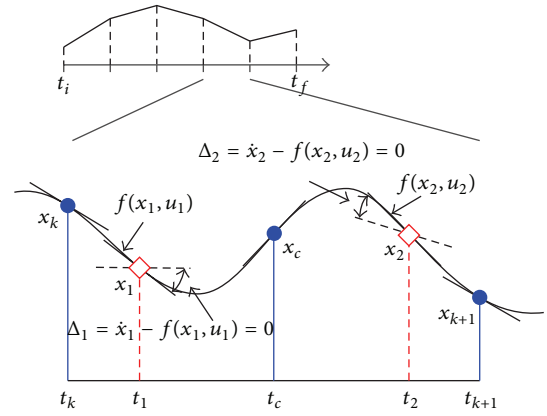


FIGURE 2: Fifth-order Gauss-Lobatto constraint formulation.

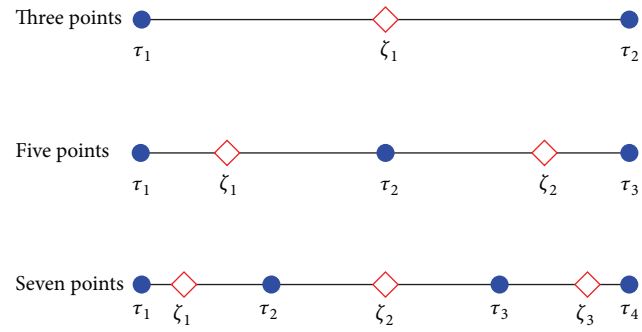


FIGURE 3: Three, five, and seven Legendre-Gauss-Lobatto points.

With reference to Figure 3, the blue points are the generic nodes, which are defined by

$$\boldsymbol{\tau}_j = \boldsymbol{\xi}_{2j-1}, \quad j = 1, \ldots, \frac{(n+1)}{2}, \tag{46}$$

while the red diamonds are the generic collocation points, denoted as

$$\boldsymbol{\zeta}_j = \boldsymbol{\xi}_{2j}, \quad j = 1, \ldots, \frac{(n-1)}{2}, \tag{47}$$

such that there is one collocation point between every two adjacent nodes [31]. The nodes are used for constructing the interpolation polynomial, while the collocation points are used to formulate the defect constraints (it is noted that nodes and collocation points should not overlap in Gauss-Lobatto method). With three points, the Gauss-Lobatto method degenerates to the Hermite-Simpson method. The state in the ith subinterval is approximated by the nth degree Hermite interpolating polynomial

$$\begin{aligned} x(\tau) \simeq {} & a_0 + a_1\tau + a_2\tau^2 + a_3\tau^3 + \cdots + a_n\tau^n, \\ & \tau \in [-1, 1], \end{aligned} \tag{48}$$

Table 1: Locations three, five, and seven Legendre-Gauss-Lobatto points.

Number of points	Location
3	0
	± 1
5	0
	$\pm\sqrt{\frac{3}{7}}$
	± 1
7	0
	$\pm\frac{\sqrt{495-66\sqrt{15}}}{33}$
	$\pm\frac{\sqrt{495+66\sqrt{15}}}{33}$
	± 1

where the coefficients of Hermite interpolating polynomial $\mathbf{a} = [a_0, a_1, a_2, \ldots, a_n]^T$ are determined by using the values of the states and vector field at the points $\boldsymbol{\tau}_j$; that is,

$$\begin{bmatrix} 1 & \tau_1^1 & \tau_1^2 & \cdots & \tau_1^{n-1} & \tau_1^n \\ 1 & \tau_2^1 & \tau_2^2 & \cdots & \tau_2^{n-1} & \tau_2^n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \tau_{(n+1)/2}^1 & \tau_{(n+1)/2}^2 & \cdots & \tau_{(n+1)/2}^{n-1} & \tau_{(n+1)/2}^n \\ 0 & 1 & 2\tau_1^1 & \cdots & (n-1)\,\tau_1^{n-2} & n\tau_1^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 2\tau_{(n+1)/2}^1 & \cdots & (n-1)\,\tau_{(n+1)/2}^{n-2} & n\tau_{(n+1)/2}^{n-1} \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \\ a_n \end{bmatrix} = \begin{bmatrix} x(\tau_1) \\ x(\tau_2) \\ \vdots \\ x(\tau_{(n+1)/2}) \\ \frac{h}{2} f(\tau_1) \\ \vdots \\ \frac{h}{2} f(\tau_{(n+1)/2}) \end{bmatrix}. \tag{49}$$

Let (49) be written as $\mathbf{Aa} = \mathbf{b}$. The term $\mathbf{A}$ depends only on the number and location of node points, whereas $\mathbf{b}$ is a concatenated vector of state and vector field values. Thus, given a vector of states, the coefficients of the Hermite interpolation in the ith subinterval, $\mathbf{a}$, can be found by $\mathbf{a} = [\mathbf{A}]^{-1}\mathbf{b}$. The values of the states located at collocation points are then

$$\mathbf{x}(\boldsymbol{\zeta}_j) = \begin{bmatrix} 1 & \zeta_1 & \zeta_1^2 & \cdots & \zeta_1^n \\ 1 & \zeta_2 & \zeta_2^2 & \cdots & \zeta_2^n \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \zeta_{(n-1)/2} & \zeta_{(n-1)/2}^2 & \cdots & \zeta_{(n-1)/2}^n \end{bmatrix} [\mathbf{A}]^{-1}\mathbf{b} = \boldsymbol{\Phi}\mathbf{b}, \quad j = 1, 2, \ldots, \frac{(n-1)}{2}. \tag{50}$$

The derivatives of the Hermite interpolating polynomial located at the collocation points are given by

$$\dot{\mathbf{x}}(\boldsymbol{\zeta}_j) = \begin{bmatrix} 0 & 1 & 2\zeta_1 & \cdots & n\zeta_1^{n-1} \\ 0 & 1 & 2\zeta_2 & \cdots & n\zeta_2^{n-1} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 1 & 2\zeta_{(n-1)/2} & \cdots & n\zeta_{(n-1)/2}^{n-1} \end{bmatrix} [\mathbf{A}]^{-1}\mathbf{b} = \boldsymbol{\Phi}'\mathbf{b}, \quad j = 1, 2, \ldots, \frac{(n-1)}{2}. \tag{51}$$

In this form the matrices $\boldsymbol{\Phi}$ and $\boldsymbol{\Phi}'$ are constants, thus a system of $(n-1)/2$ constraints per interval is obtained as follows:

$$\boldsymbol{\Delta}_i = \dot{\mathbf{x}}(\boldsymbol{\zeta}_j) - \frac{h}{2} f\left(\mathbf{x}(\boldsymbol{\zeta}_j), \mathbf{u}(\boldsymbol{\zeta}_j)\right) = \boldsymbol{\Phi}'\mathbf{b} - \frac{h}{2} f\left(\boldsymbol{\Phi}\mathbf{b}, \mathbf{u}(\boldsymbol{\zeta}_j)\right), \tag{52}$$

and the analytic Jacobian for the defect constraints can be derived by chain rule as

$$\begin{aligned} &\frac{\partial \boldsymbol{\Delta}_i}{\partial\left[\mathbf{x}(\boldsymbol{\tau}_j), \mathbf{u}(\boldsymbol{\tau}_j)\right]_i} \\ &\quad = \boldsymbol{\Phi}' \frac{\partial \mathbf{b}}{\partial\left[\mathbf{x}(\boldsymbol{\tau}_j), \mathbf{u}(\boldsymbol{\tau}_j)\right]_i} \\ &\qquad - \frac{h}{2}\left[\frac{\partial f\left(\mathbf{x}(\boldsymbol{\zeta}_j), \mathbf{u}(\boldsymbol{\zeta}_j)\right)}{\partial \mathbf{x}(\boldsymbol{\zeta}_j)} \boldsymbol{\Phi} \frac{\partial \mathbf{b}}{\partial\left[\mathbf{x}(\boldsymbol{\tau}_j), \mathbf{u}(\boldsymbol{\tau}_j)\right]_i}\right. \\ &\qquad\quad \left. + \frac{\partial f\left(\mathbf{x}(\boldsymbol{\zeta}_j), \mathbf{u}(\boldsymbol{\zeta}_j)\right)}{\partial \mathbf{u}(\boldsymbol{\zeta}_j)} \frac{\partial \mathbf{u}(\boldsymbol{\zeta}_j)}{\partial\left[\mathbf{x}(\boldsymbol{\tau}_j), \mathbf{u}(\boldsymbol{\tau}_j)\right]_i}\right]. \end{aligned} \tag{53}$$

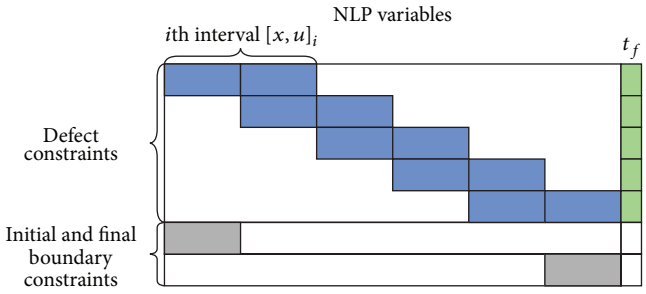


Figure 4: Jacobian structure for third-order Gauss-Lobatto method.

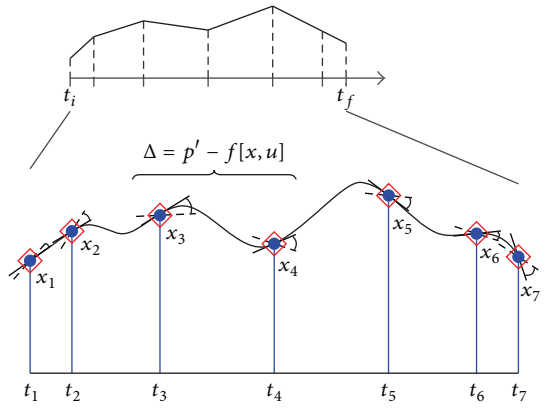


Figure 5: Formulation of the collocation constraints in the pseudospectral method.

If the final transfer time t_f is a variable, within each segment, the analytic Jacobian of t_f is written as

$$\frac{\partial \boldsymbol{\Delta}_i}{\partial t_f} = \boldsymbol{\Phi}' \frac{\partial \mathbf{b}}{\partial h} \frac{\partial h}{\partial t_f} - \left[\frac{1}{2} \frac{\partial h}{\partial t_f} f\left(\mathbf{x}\left(\boldsymbol{\zeta}_j\right), \mathbf{u}\left(\boldsymbol{\zeta}_j\right)\right) + \frac{h}{2} \frac{\partial f\left(\mathbf{x}\left(\boldsymbol{\zeta}_j\right), \mathbf{u}\left(\boldsymbol{\zeta}_j\right)\right)}{\partial \mathbf{x}\left(\boldsymbol{\zeta}_j\right)} \boldsymbol{\Phi} \frac{\partial \mathbf{b}}{\partial h} \frac{\partial h}{\partial t_f} \right]. \tag{54}$$

A graphical illustration of the Jacobian structure with free final time t_f is given in Figure 4. It can be seen that each segment has the same Jacobian module; thus (53)-(54) can be calculated offline and stored before solving the NLP. In this way, the computational time can be considerably reduced compared to the use of finite difference approximation.

3.3. Pseudospectral Method. Recently, new direct methods have been applied to low-thrust trajectory optimization, the most popular one being the pseudospectral scheme [32–35]. The major difference between Gauss-Lobatto and pseudospectral collocation schemes is the way in which the interpolation polynomial is constructed and the defect constraints are defined. When the pseudospectral method

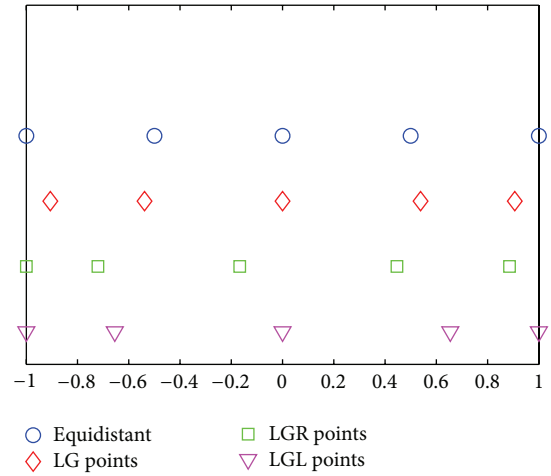


Figure 6: Legendre-Gauss (LG), Legendre-Gauss-Radau (LGR), and Legendre-Gauss-Lobatto (LGL) with five points.

Table 2: Position of equidistant, LG, LGR, and LGL with five points within $[-1, 1]$.

Case	Type	x_1	x_2	x_3	x_4	x_5
1	Equidistant	−1	−0.5	0	0.5	1
2	LG points	−0.906	−0.538	0	0.538	0.906
3	LGR points	−1	−0.720	−0.167	0.446	0.886
4	LGL points	−1	−0.654	0	0.655	1

is employed, a global Lagrange interpolation polynomial is constructed to approximate the state profile. The polynomial is then differentiated and evaluated at all the nodes to compute interpolated derivative values of the states, and equations of motion are used to provide physical time derivatives. The differences between the two sets of time derivatives form the defects. The formulation of defect constraints is sketched in Figure 5.

The defect constraints of the pseudospectral method can be derived as follows. The state curve $p(t)$ is approximated by using values of x at the discrete time points t_i and the corresponding Lagrange interpolating polynomials $L_i(t)$, $(i = 1, \ldots, n)$

$$p(t) = \sum_{i=1}^{n} L_i(t)\, x(t_i), \tag{55}$$

where $L_i(t)$ $(i = 1, \ldots, n)$ are defined as

$$L_i(t) = \prod_{j=1, j\neq i}^{n} \frac{t - t_j}{t_i - t_j}. \tag{56}$$

Then, the derivative of $p(t)$ is given:

$$p'(t) = \sum_{i=1}^{n} L_i'(t)\, x(t_i). \tag{57}$$

The time derivative of the polynomial can be expressed in compact form through

$$\mathbf{p}' = \mathbf{D}\mathbf{x}, \tag{58}$$

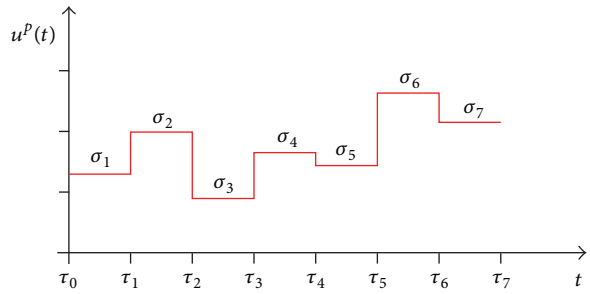


FIGURE 7: Piecewise-constant control approximation with equidistant knot points.

where $\mathbf{D}$ is differentiation matrix. This only depends on the chosen node spacing, so it can be calculated offline and stored. Then the defect constraints can be written as

$$\boldsymbol{\Delta} = \mathbf{p}' - \mathbf{f}(\mathbf{x}, \mathbf{u}) = \mathbf{D}\mathbf{x} - \mathbf{f}(\mathbf{x}, \mathbf{u}). \tag{59}$$

A variety of pseudospectral schemes emerged and the major differences are the way in which the nodes are selected. The most general form is Legendre-Gauss-Lobatto discretization [36], but Legendre-Gauss [34, 37] and Legendre-Gauss-Radau [35] are also commonly used. Figure 6 illustrates three different kinds of Gauss points. Table 2 lists the position of equidistant points with three kinds of Gaussian points. We can see that LG points are located on the open interval $(-1, 1)$, in contrast, LGR points are located on half-open interval $[-1, 1)$, and LGL points are located on closed interval $[-1, 1]$.

The main characteristic of the Gauss-Lobatto method is the combination of reasonable accuracy with highly sparse constraint Jacobians and Hessians. The pseudospectral method has a more elegant form of Jacobian and it offers spectral accuracy for smooth problems, but the constraint Jacobian is much denser. So there is a balance between accuracy and efficiency. In general, the Gauss-Lobatto method use a limited number of nodes per segment ($n < 4$) with many segments, while pseudospectral methods use many nodes per segment ($n > 10$) with a limited number of segments.

3.4. Control Parameterization Method. The control parameterization method is a further direct transcription technique for solving optimal control problems [38–40]. In this method, only the controls are discretized and represented by a linear combination of basis functions. This approximation scheme yields a suboptimal control for the original problem. The optimal control problem is converted to an approximate nonlinear optimization problem with a finite number of decision variables, which can be solved by nonlinear programming techniques. The advantage of piecewise-constant approximation scheme is due to its simplicity and convergence [41].

To solve optimal control problem using the control parameterization method, we approximate the control profile $\mathbf{u}$ as (see Figure 7)

$$\mathbf{u}(t) \approx \mathbf{u}^p(t) = \boldsymbol{\sigma}^k, \quad t \in [\tau_{k-1}, \tau_k), \ k = 1, \ldots, p, \tag{60}$$

where $p \geq 1$ is a given integer, τ_k, $k = 0, \ldots, p$ are knot points, and $\boldsymbol{\sigma}^k \in \mathbb{R}^r$, $k = 1, \ldots, p$ are vectors containing the approximate control values. The knot points satisfy

$$0 = \tau_0 \leq \tau_1 \leq \tau_2 \leq \cdots \leq \tau_{p-1} \leq \tau_p = T. \tag{61}$$

The approximate control $\mathbf{u}^p$ can be written as

$$\mathbf{u}^p(t) = \sum_{k=1}^{p-1} \boldsymbol{\sigma}^k \chi_{[\tau_{k-1}, \tau_k)}(t) + \boldsymbol{\sigma}^p \chi_{[\tau_{p-1}, \tau_p)}(t), \tag{62}$$

where for a given subinterval $\mathcal{I} \subset [0, T]$, the characteristic function $\chi_{\mathcal{I}} : \mathbb{R} \to \mathbb{R}$ is defined by

$$\chi_{\mathcal{I}} := \begin{cases} 1, & \text{if } t \in \mathcal{I}, \\ 0, & \text{otherwise.} \end{cases} \tag{63}$$

Note that $\mathbf{u}^p$ is a piecewise-constant function with potential discontinuities at the points $t = \tau_k$, $k = 1, \ldots, p-1$. These points are called *switching times*. Reference [40] surveys the key developments in the control parameterization.

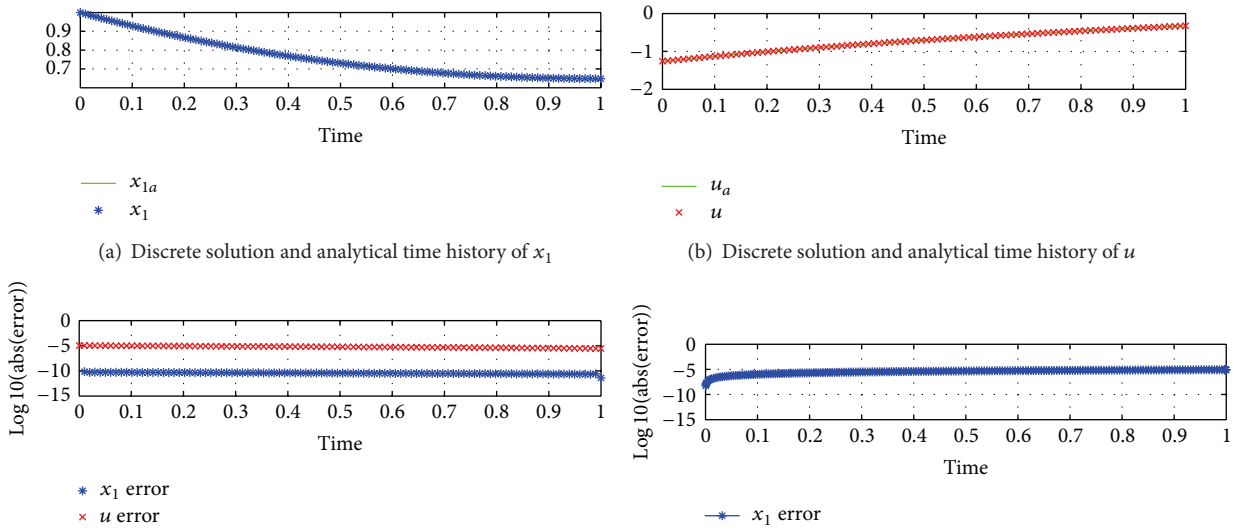
4. Computational Issues

Solving optimal trajectory design problems with direct methods yields a large NLP problem. Some computational issues should be implemented to improve efficiency and robustness. Some of these issues are discussed in this section.

4.1. Scaling. An important issue that arises in the solution of NLP is *scaling*. It is known that a poorly scaled problem can lead to either extremely slow convergence or divergence [27]. When the relative size of variables or constraints in a problem is vastly different, the variables should be transformed into a relatively similar scale. In trajectory optimization problems the scaling process can be implemented manually by introducing dimensionless distance unit L, time unit T, velocity unit V, and mass unit M.

4.2. Sparse Matrix. A sparse matrix is a matrix filled primarily with zeros; the opposite is referred to as dense matrix. One of the major advantages of using sparse matrix is that finite memory can be saved. In the case of direct collocation methods, it is well-known that the vast majority of the Jacobian and Hessian matrix elements are zero. When a large number of nodes are taken to have a smooth state and control time histories, "out-of-memory" problems can be avoided by using sparse matrices. Another advantage is that the computational time is reduced because only nonzero elements need to be manipulated. Some NLP solvers, such as SNOPT [42] and IPOPT [43], take advantage of sparse techniques. Furthermore IPOPT computes second derivatives with quasi-Newton methods.

4.3. Differentiation. Direct methods use gradient-based techniques for solving the NLP, which requires the gradient information of the objective function and constraints with respect to the NLP variables. It is known that providing the



(a) Discrete solution and analytical time history of x_1

(b) Discrete solution and analytical time history of u

(c) Error of x_1 and u located at discrete points

(d) Reintegrate dynamics with discrete control solution

Figure 8: Simulation results of Example 1, 100 points.

analytic derivative results in exact and fast optimization, but sometimes it is impossible or time consuming to compute derivatives analytically. Alternative methods are needed to obtain the necessary gradients. Finite differences are implemented to compute approximate derivatives. Forward and central difference schemes are

$$\frac{df}{dx} \simeq \frac{f(x+h) - f(x)}{h}, \qquad \frac{df}{dx} \simeq \frac{f(x+h) - f(x-h)}{2h}, \tag{64}$$

where h is a properly chosen perturbation, which should be sufficiently small to provide a good approximation to the derivatives but not too small to induce round-off errors.

5. Practical Examples

5.1. A Simple Optimal Control Problem. In this section, a simple optimal control problem [44] is solved with Hermite-Simpson method, and certain issues are set to speed-up the computation time and improving accuracy. Consider the problem of finding $u(t)$ with fixed initial and final time $[0, 1]$ to minimize the cost

$$J = x_2(1) \tag{65}$$

subject to system equations

$$\dot{x}_1 = 0.5x_1 + u, \qquad \dot{x}_2 = u^2 + x_1 u + \frac{5}{4}x_1^2 \tag{66}$$

and to the boundary conditions

$$x_1(0) = 1, \qquad x_2(0) = 0 \tag{67}$$

Table 3: Improvement with Jacobian and Hessian, 100 points.

Case	Method	Number of iterations	CPU time
1	Finite differences (default)	116	17.21 s
2	Jacobian only	116	8.77 s
3	Jacobian and Hessian	12	0.38 s

with the following bound box of state and control variables

$$-10 \le x_1(t) \le 10, \qquad -10 \le x_2(t) \le 10, \qquad -10 \le u(t) \le 10. \tag{68}$$

The analytical solution of this problem is

$$x_{1a}(t) = \frac{\cosh(1-t)}{\cosh(1)}, \qquad u_a(t) = \frac{-(\tanh(1-t) + 0.5)\cosh(1-t)}{\cosh(1)}. \tag{69}$$

This problem is solved by using Matlab's "fmincon" with "interior-point" algorithm [45]. The analytic solution and final optimized results are plotted in Figures 8(a) and 8(b), where the green line indicates analytic solutions (i.e., x_{1a} and u_a), and the discrete results obtained from NLP solver are marked by blue asterisk and red cross, respectively. Figure 8(c) shows the error of x_1 and u in detail, it can be seen that the error of x_1 is much lower than that of u (10^{-10} for x_1 versus 10^{-5} for u). This is because the third-order Hermite interpolation polynomial is implemented to approximate the states, while linear interpolation is used to represent the control curve. In order to check the control accuracy, (66) are integrated from the initial condition with

the discrete optimal solution u, and then the error of x_1 is plotted in Figure 8(d). It can be found that the error of x_1 in the whole time domain $[0, 1]$ is below 10^{-5}.

The "finite differences" method is the default option to obtain the gradient information, however computational efficiency can be remarkably improved by adding analytical Jacobian and Hessian. The comparison of these three different cases is listed in Table 3. All of the three cases use 100 discrete points. In case 1, the optimization takes 116 steps and is 17.21 s. In case 2, adding Jacobian only does not reduce the number of iterations but saves almost half of the CPU time (17.21 s versus 8.77 s). In case 3, providing analytical Jacobin and Hessian highly reduces both the number of iterations and the computational time, which drops dramatically down to 0.38 s.

The structure of Jacobian is shown in Figure 9(a), where the rows are constraints while the columns are NLP decision variables. The diagonal elements are Jacobian of collocation constraints and the last two lines are Jacobian of initial boundary constraints. Figure 9(b) illustrates the structure of Hessian, where both the row and the column represent NLP decision variables. It can be seen that Hermite-Simpson method has highly sparse Jacobian and Hessian structures.

Because only few elements in Figures 9(a) and 9(b) are nonzero, sparse matrices can be used. In Figure 8, the accuracy of state and control solution is not so good. A finer mesh with 5000 points is used with "interior-point" method and matrix sparsity, which is impossible with other fmincon algorithms (that use dense matrixes). The results are illustrated in Figure 10. Compared to Figure 8, it is noted that the error of u reduces to around 10^{-8}. In order to check the accuracy of control solution, the dynamics are reintegrated with the initial condition and optimal control history; the maximal error on x_1 drops down to 10^{-8} (see Figure 10(d)). A parametric analysis has been carried out with variable grid points. The outcome is reported in Table 4.

5.2. Planar Low-Thrust Orbit Transfer. For a transfer trajectory to be determined, using Cartesian coordinates is the simplest but most disadvantageous choice; this is because a lot of discrete points will be used to catch the rapidly changing position and velocity variables. On the contrary, slowly changing state variables make the NLP problem both efficient and robust. Polar coordinates are then used for the planar two-body dynamics; that is,

$$\begin{aligned} \dot{r} &= v_r, \\ \dot{\theta} &= \frac{v_t}{r}, \\ \dot{v}_r &= \frac{v_t^2}{r} - \frac{\mu}{r^2} + u \sin\phi, \\ \dot{v}_t &= -\frac{v_r v_t}{r} + u \cos\phi. \end{aligned} \tag{70}$$

Table 4: Comparison of different grid points.

Case	Points	Number of iterations	CPU time	Maximum error of u
1	100	12	0.38 s	$1.07e-5$
2	200	12	0.74 s	$2.65e-6$
3	500	10	1.82 s	$4.24e-7$
4	1000	13	4.64 s	$1.06e-7$
5	2000	17	15.55 s	$2.82e-8$
6	5000	15	65.04 s	$9.16e-9$

In (70), r is the spacecraft radius, θ is phase angle, v_r and v_t are the radial and transversal velocities, respectively, μ is the gravitational constant, u is the propulsive acceleration, and ϕ is thrust angle (see Figure 11). The initial boundary conditions set the spacecraft in its initial circular orbit at t_i:

$$\begin{aligned} r(t_i) &= 1, \\ \theta(t_i) &= 0, \\ v_r(t_i) &= 0, \\ v_t(t_i) &= 1, \end{aligned} \tag{71}$$

while the final boundary conditions place the spacecraft into a target circular orbit at t_f:

$$\begin{aligned} r(t_f) &= 4, \\ v_r(t_f) &= 0, \\ v_t(t_f) &= 0.5. \end{aligned} \tag{72}$$

The nondimensional μ and maximum thrust acceleration $u_{\max}$ are

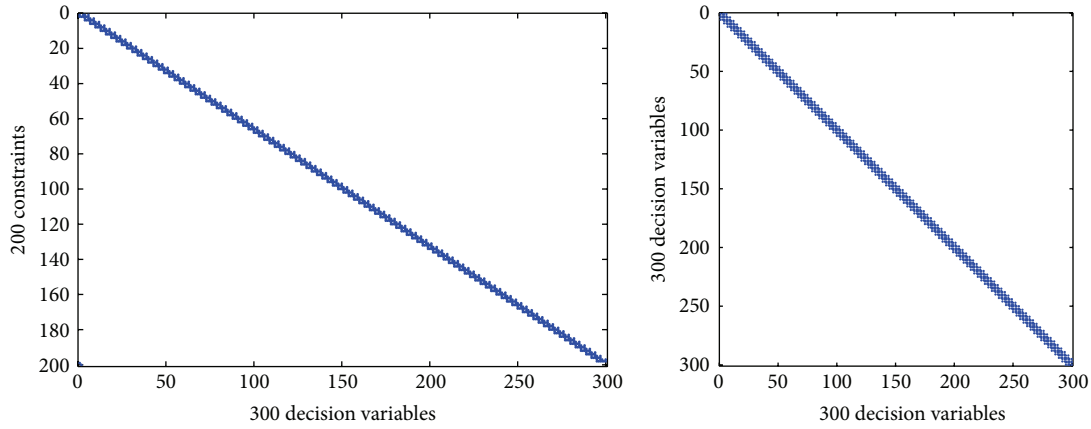
$$\begin{aligned} \mu &= 1, \\ u_{\max} &= 0.01. \end{aligned} \tag{73}$$

In the following, the high efficient NLP solver IPOPT [43] is used for large-scale nonlinear optimization.

5.2.1. Time-Optimal Problem. The goal of time-optimal problem is to find the functions $u(t)$ and $\phi(t)$ that minimize the performance index

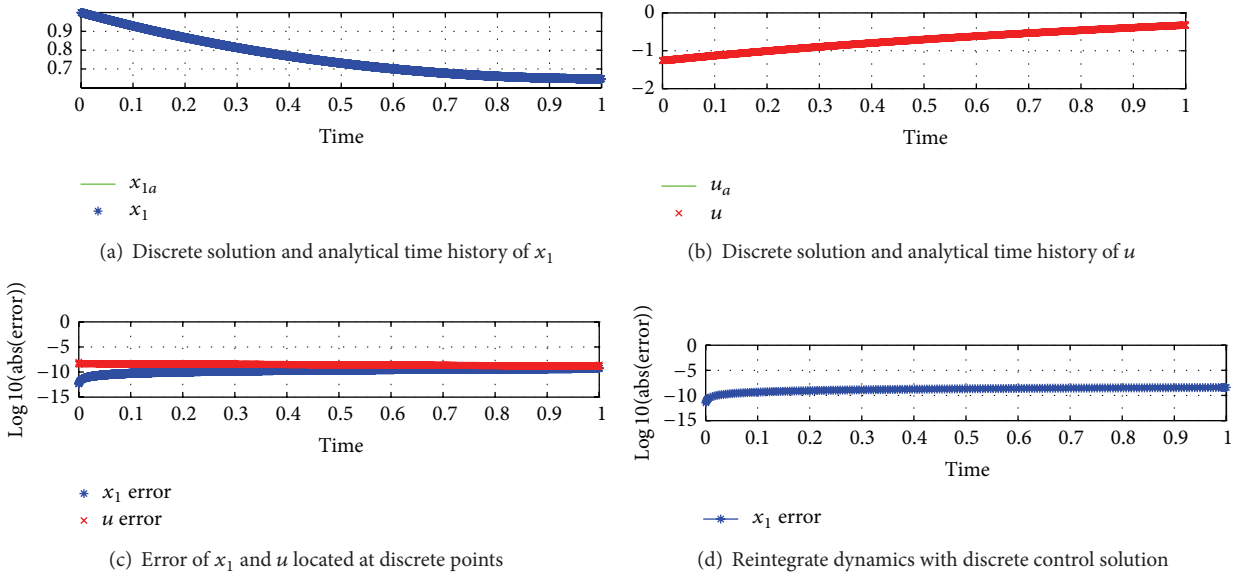
$$J = t_f \tag{74}$$

under the dynamic constraints and the boundary conditions above. This problem is solved with $N = 400$ uniformly spaced points with Hermite-Simpson method. (That is, 2401 NLP variables: the states and the controls at the nodes plus the final time.) Figure 12 shows the transfer trajectory; the cyan dashed line is the initial guess (although the convergence basin of direct methods is larger than that of indirect methods, a good initial guess is essential; in time-optimal problem,



(a) Jacobian structure, 100 points (b) Hessian structure, 100 points

Figure 9: Jacobian and Hessian structures.



(a) Discrete solution and analytical time history of x_1 (b) Discrete solution and analytical time history of u

(c) Error of x_1 and u located at discrete points (d) Reintegrate dynamics with discrete control solution

Figure 10: Simulation results of Example 1, 5000 points.

tangential thrust can be used as a simple but suitable approach to guess an initial solution), the thin blue line denotes the final transfer orbit, and the thick red line shows the thrust arc. Figure 13 shows the time history of state variables. The final transfer time is 55.5. Figure 14 shows the profile of the optimal control variables u, ϕ; it can be seen that the engine is on duty along the entire orbit, to shorten the transfer time.

5.2.2. Fuel-Optimal Problem. In the fuel-optimal problem, the functions $u(t)$ and $\phi(t)$ are sought to minimize the performance index

$$J = \int_{t_i}^{t_f} u \, dt. \tag{75}$$

Figure 15 illustrates the optimal transfer trajectory. This solution has been obtained with $N = 800$ mesh points (i.e., 4801 NLP variables). In this problem, tangential thrust with magnitude of $0.5u_{\max}$ is used to produce the initial guess (cyan dashed line in Figure 15). It can be seen that the thruster is on duty across the periapsis; the only maneuver performed at the apoapsis injects the spacecraft into a nearly circular orbit to acquire the final orbit. The final time of flight is 122.3. Figure 16 illustrates the time history of the state variables, whereas Figure 17 illustrates the time history of control variables. A bang-bang structure is found, indicating the optimality of the solution found.

6. Conclusions

In this survey, some issues related to the direct transcription and collocation for optimal low-thrust space trajectory

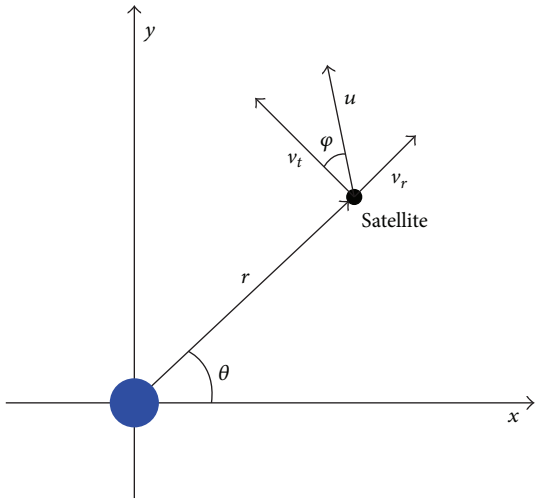


Figure 11: Two-body orbit transfer in polar coordinates.

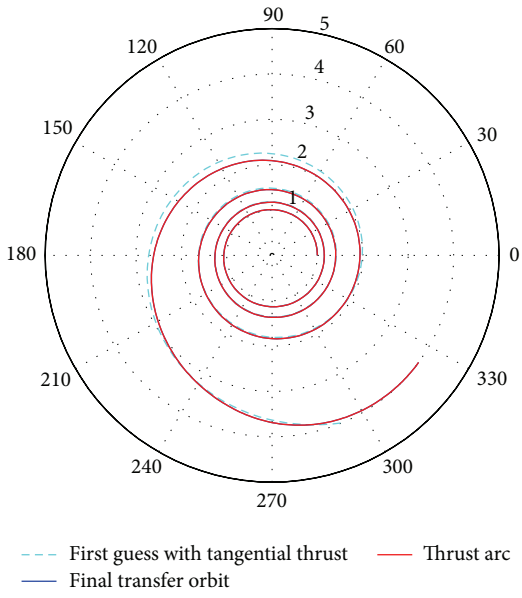


Figure 12: Optimal transfer orbit of time-optimal problem.

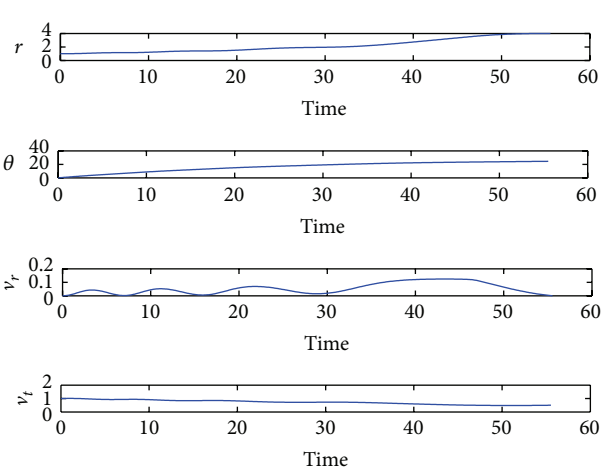


Figure 13: Time history of state variables.

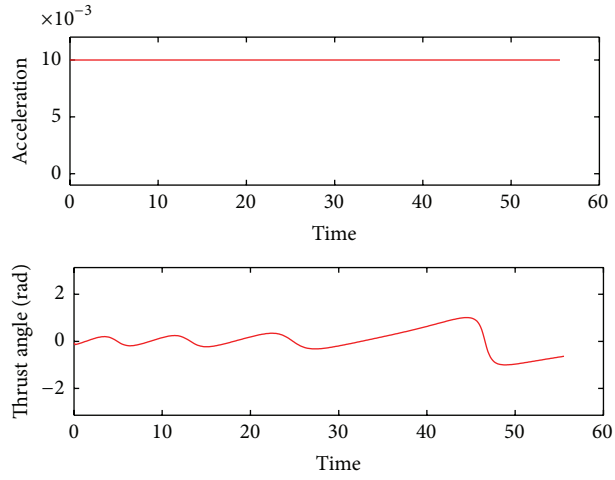


Figure 14: Time history of control variables.

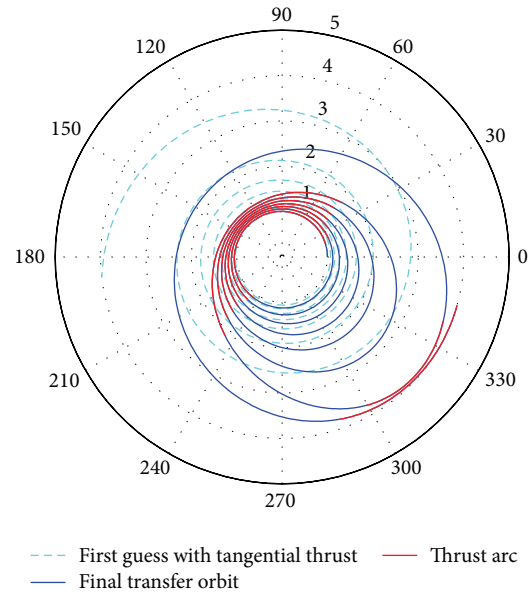


Figure 15: Optimal transfer orbit of fuel-optimal problem.

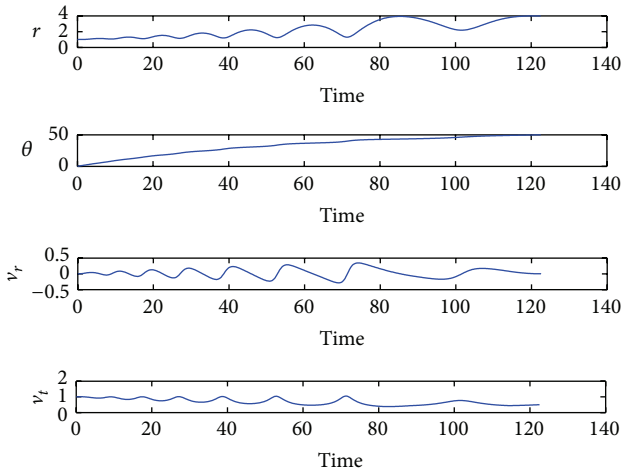


Figure 16: Time history of state variables.

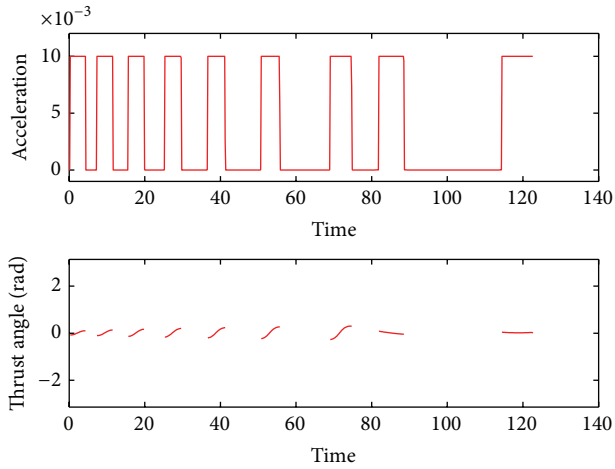


Figure 17: Time history of control variables.

design are discussed. The principles of direct transcription and collocation are given, and emphasis is put on high-order numerical integration. Hermite-Simpson, high-order Gauss-Lobatto, and the pseudospectral schemes are introduced. For the Gauss-Lobatto method, a scheme to handle arbitrary order is given, together with the procedure to derive analytical gradients. The pseudospectral method has an impressive convergence rate; however, it is more suitable for smooth problems. A simple example (two states, one control) has been solved with the Hermite-Simpson method. This example shows outstanding efficiency when analytical gradient is provided. A second example consists in low-thrust transfers in the two-body model. From this example it can be concluded that compared with pseudospectral method, the Gauss-Lobatto method is more suitable for fuel-optimal problems with discontinued control structures. In both examples, effort is put to give insights on the computational issues for a practical implementation of the method. However, if a long duration problem is encountered (with increasing number of on/off structures) or a spacecraft equipped with very small thrust engines is dealt with (large number of spirals), the NLP solver becomes ill-conditioned. In these cases, indirect approach with homotopic or hybrid methods could be more suitable.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

C. Zhang would like to acknowledge the Chinese Scholarship Council (CSC) which financially supported his visit to Politecnico di Milano.

References

[1] M. D. Rayman, P. Varghese, D. H. Lehman, and L. L. Livesay, "Results from the deep space 1 technology validation mission," *Acta Astronautica*, vol. 47, no. 2, pp. 475–487, 2000.

[2] J. Kawaguchi, A. Fujiwara, and T. K. Uesugi, "The ion engines cruise operation and the earth swingby of "Hayabusa" (Muses-C)," *Space Technology*, vol. 25, no. 2, pp. 105–115, 2005.

[3] J. Kugelberg, P. Bodin, S. Persson, and P. Rathsman, "Accommodating electric propulsion on SMART-1," *Acta Astronautica*, vol. 55, no. 2, pp. 121–130, 2004.

[4] R. Armellin, P. Di Lizia, F. Topputo, M. Lavagna, F. Bernelli-Zazzera, and M. Berz, "Gravity assist space pruning based on differential algebra," *Celestial Mechanics and Dynamical Astronomy*, vol. 106, no. 1, pp. 1–24, 2009.

[5] Y. Tsuda, O. Mori, R. Funase et al., "Flight status of IKAROS deep space solar sail demonstrator," *Acta Astronautica*, vol. 69, no. 9-10, pp. 833–840, 2011.

[6] C. Katan, "Nasa's next solar sail: lessons learned from nanosail-d2," in *Proceedings of the 26th Annual AIAA/USU Conference on Small Satellites: Enhancing Global Awareness through Small Satellites*, Logan, UT, USA, 2012.

[7] A. Owis, F. Topputo, and F. Bernelli-Zazzera, "Radially accelerated optimal feedback orbits in central gravity field with linear drag," *Celestial Mechanics and Dynamical Astronomy*, vol. 103, no. 1, pp. 1–16, 2009.

[8] G. G. Wawrzyniak and K. C. Howell, "Numerical techniques for generating and refining solar sail trajectories," *Advances in Space Research*, vol. 48, no. 11, pp. 1848–1857, 2011.

[9] L. Casalino, G. Colasurdo, and D. Pastrone, "Optimal low-thrust escape trajectories using gravity assist," *Journal of Guidance, Control, and Dynamics*, vol. 22, no. 5, pp. 637–642, 1999.

[10] L. Casalino, G. Colasurdo, and M. R. Sentinella, "Low-thrust trajectories to mercury with multiple gravity assists," in *Proceedings of the 43rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, pp. 2276–2284, July 2007.

[11] T. Guo, F. Jiang, H. Baoyin, and J. Li, "Fuel optimal low thrust rendezvous with outer planets via gravity assist," *Science China: Physics, Mechanics and Astronomy*, vol. 54, no. 4, pp. 756–769, 2011.

[12] F. Jiang, H. Baoyin, and J. Li, "Practical techniques for low-thrust trajectory optimization with homotopic approach," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 1, pp. 245–258, 2012.

[13] F. Topputo, "On optimal two-impulse Earth-Moon transfers in a four-body model," *Celestial Mechanics and Dynamical Astronomy*, vol. 117, no. 3, pp. 279–313, 2013.

[14] G. Mingotti, F. Topputo, and F. Bernelli-Zazzera, "Efficient invariant-manifold, low-thrust planar trajectories to the Moon," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 2, pp. 817–831, 2012.

[15] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, vol. 19, SIAM, Philadelphia, Pa, USA, 2nd edition, 2010.

[16] J. Prussing, "Primer vector theory and applications," in *Spacecraft Trajectory Optimization*, pp. 16–36, Cambridge University Press, Cambridge, UK, 2010.

[17] S. W. Paris, J. P. Riehl, and W. K. Sjauw, "Enhanced procedures for direct trajectory optimization using nonlinear programming and implicit integration," in *Proceedings of the AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, pp. 21–24, August 2006.

[18] A. Rao, "A survey of numerical methods for optimal control," *Advances in the Astronautical Sciences*, vol. 135, no. 1, pp. 497–528, 2009.

[19] A. Bryson and Y. Ho, *Applied Optimal Control*, vol. 8, Blaisdell, Waltham, Mass, USA, 1969.

[20] D. Hull, *Optimal Control Theory for Applications*, Springer, 2003.

[21] D. Naidu, *Optimal Control Systems*, vol. 2, CRC press, 2002.

[22] D. F. Lawden, *Optimal Trajectories for Space Navigation*, Butterworths, 1963.

[23] R. P. Russell, "Primer vector theory applied to global low-thrust trade studies," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 2, pp. 460–472, 2007.

[24] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *Journal of Guidance, Control, and Dynamics*, vol. 0, no. 4, pp. 338–342, 1987.

[25] P. J. Enright and B. A. Conway, "Discrete approximations to optimal trajectories using direct transcription and nonlinear programming," *Journal of Guidance, Control, and Dynamics*, vol. 15, no. 4, pp. 994–1002, 1992.

[26] J. T. Betts, "Optimal interplanetary orbit transfers by direct transcription," *Journal of the Astronautical Sciences*, vol. 42, no. 3, pp. 247–268, 1994.

[27] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[28] A. L. Herman and B. A. Conway, "Direct optimization using collocation based on high-order Gauss-Lobatto quadrature rules," *Journal of Guidance, Control, and Dynamics*, vol. 19, no. 3, pp. 592–599, 1996.

[29] C. J. Goh and K. L. Teo, "Control parametrization: a unified approach to optimal control problems with general constraints," *Automatica*, vol. 24, no. 1, pp. 3–18, 1988.

[30] P. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, 1981.

[31] P. Williams, "Hermite-legendre-gauss-lobatto direct transcription in trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 4, pp. 1392–1395, 2009.

[32] I. M. Ross and F. Fahroo, "Pseudospectral knotting methods for solving optimal control problems," *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 3, pp. 397–405, 2004.

[33] F. Fahroo and I. M. Ross, "Costate estimation by a Legendre pseudospectral method," *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 2, pp. 270–277, 2001.

[34] D. Benson, *A gauss pseudospectral transcription for optimal control [Ph.D. thesis]*, Massachusetts Institute of Technology, 2005.

[35] G. Huntington, *Advancement and analysis of a Gauss pseudospectral transcription for optimal control problems [Ph.D. thesis]*, Citeseer, 2007.

[36] G. Elnagar, M. A. Kazemi, and M. Razzaghi, "Pseudospectral Legendre method for discretizing optimal control problems," *IEEE Transactions on Automatic Control*, vol. 40, no. 10, pp. 1793–1796, 1995.

[37] A. V. Rao, D. A. Benson, C. Darby et al., "Algorithm 902: GPOPS, A MATLAB software for solving multiple-phase optimal control problems using the gauss pseudospectral method," *ACM Transactions on Mathematical Software*, vol. 37, no. 2, article 22, 2010.

[38] Q. Lin, R. Loxton, and K. L. Teo, "The control parameterization method for nonlinear optimal control: a survey," *Journal of Industrial and Management Optimization*, vol. 10, no. 1, pp. 275–309, 2014.

[39] Q. Chai, R. Loxton, K. L. Teo, and C. Yang, "A max-min control problem arising in gradient elution chromatography," *Industrial and Engineering Chemistry Research*, vol. 51, no. 17, pp. 6137–6144, 2012.

[40] Q. Lin, R. Loxton, K. L. Teo, and Y. H. Wu, "A new computational method for a class of free terminal time optimal control problems," *Pacific Journal of Optimization*, vol. 7, no. 1, pp. 63–81, 2011.

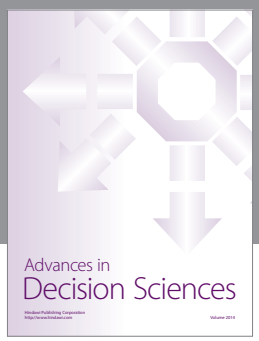
[41] R. Loxton, Q. Lin, V. Rehbock, and K. Teo, "Control parameterization for optimal control problems with continuous inequality constraints: new convergence results," 2012.

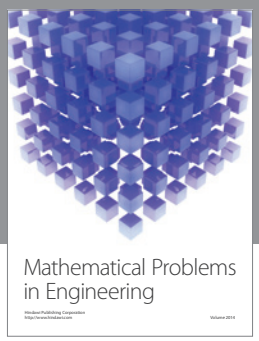
[42] P. Gill, W. Murray, and M. Saunders, *Users Guide for Snopt Version 7: Software for Large-Scale Nonlinear Programming*, 2006.

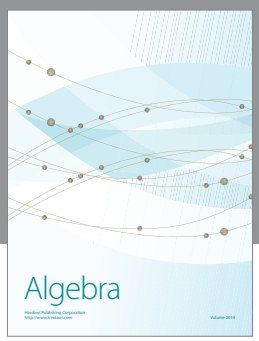
[43] Y. Kawajir, C. Laird, and A. Wachter, "Introduction to ipopt: a tutorial for downloading, installing, and using ipopt," Tech. Rep., Carnegie Mellon University, 2006.

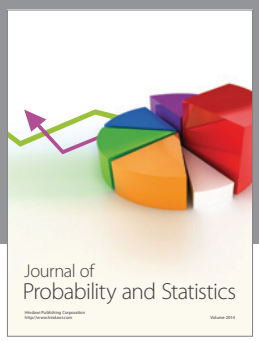
[44] V. M. Becerra, "Practical direct collocation methods for computational optimal control," in *Modeling and Optimization in Space Engineering*, pp. 33–60, Springer, New York, NY, USA, 2013.

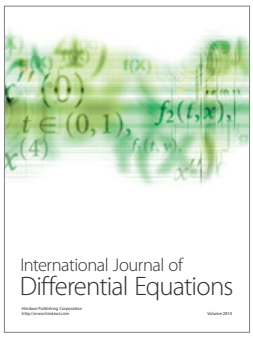
[45] A. Grace and M. Works, *Optimization Toolbox: for USe with MATLAB: USer's Guide*, Math Works, 1990.

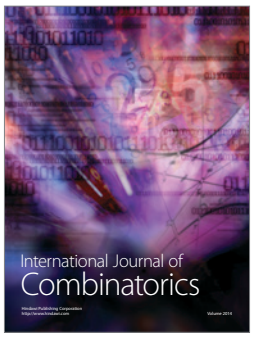


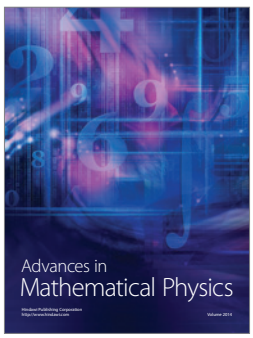


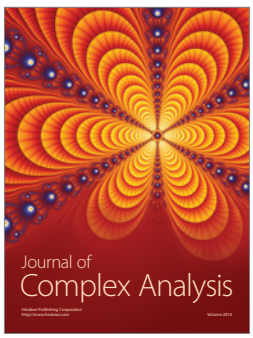


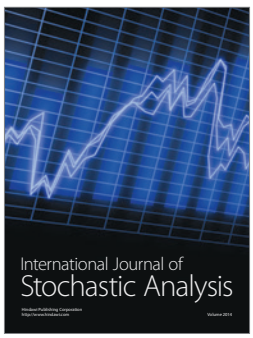


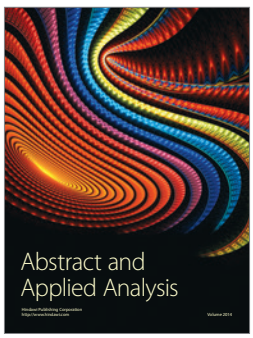


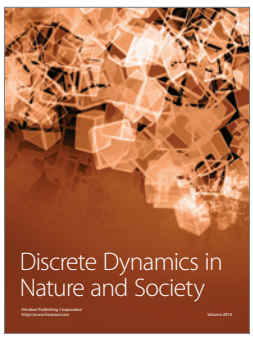


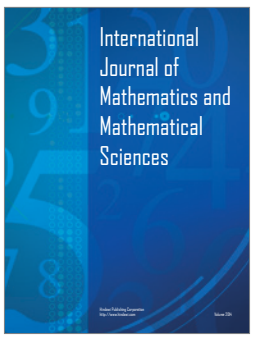


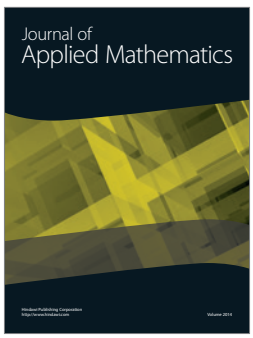












Hindawi
Submit your manuscripts at
http://www.hindawi.com