(Preprint) AAS 09-334

A SURVEY OF NUMERICAL METHODS FOR OPTIMAL CONTROL

Anil V. Rao*

A survey of numerical methods for optimal control is given. The objective of the article is to describe the major methods that have been developed over the years for solving general optimal control problems. In particular, the two broad classes of indirect and direct methods are discussed, the main approaches that are used in each class are described, and an extensive list is provided to relevant work in the open literature. Several important computational issues are then discussed and well known software programs for solving optimal control problems are described. Finally, a discussion is given on how to choose a method.

INTRODUCTION

Optimal control is a subject where it is desired to determine the inputs to a dynamical system that optimize (i.e., minimize or maximize) a specified performance index while satisfying any constraints on the motion of the system. Because of the complexity of most applications, optimal control problems are most often solved numerically. Numerical methods for solving optimal control problems date back nearly five decades to the 1950s with the work of Bellman.[1–6] From that time to the present, the complexity of methods and corresponding complexity and variety of applications has increased tremendously making optimal control a discipline that is relevant to many branches of engineering.

Before proceeding to the details of the survey, it is important to distinguish between the terms *trajectory optimization* and *optimal control*. Often, these two terms are used interchangeably. In the case of a problem where the inputs to the system are static parameters and it is desired to determine the values of these parameters and the trajectory that optimizes a given performance index (i.e., a *function optimization* problem), the term "trajectory optimization" is most appropriate. On the other hand, in the case where the inputs to the systems are themselves functions and it is desired to determine the particular input function and trajectory that optimize a given performance index (i.e., a *functional optimization problem*), the appropriate term "optimal control." In many cases, however, the some of the inputs to the system are static parameters while other inputs are functions of time. Because this paper considers numerical methods for problems where it is desired to determine both functions of time and static parameters (i.e., optimal control problems) we will use the more generic terminology "optimal control problem."

Numerical methods for solving optimal control problems are divided into two major classes: indirect methods and direct methods. In an indirect method, the *calculus of variations*[7–17] is used to determine the first-order optimality conditions of the original optimal control problem. The indirect approach leads to a multiple-point *boundary-value problem* that is solved to determine candidate optimal trajectories called *extremals*. Each of the computed extremals is then examined to see if it is a local minimum, maximum, or a saddle point. Of the locally optimizing solutions, the particular extremal with the lowest cost is chosen. In a direct method, the state and/or control of the optimal control problem is discretized in some manner and the problem is transcribed[18] to a *nonlinear optimization problem* or *nonlinear programming problem*[19–22] (NLP). The NLP is then solved using well known optimization techniques.[23–28]

It is seen that indirect methods and direct method emanate from two different philosophies. On the one hand, the indirect approach solves the problem indirectly (thus the name, *indirect*) by converting the optimal control problem to a boundary-value problem. As a result, in an indirect method the optimal solution is found by solving a system of differential equations that satisfies endpoint and/or interior point conditions. On the other hand, in a direct method the optimal solution is found by transcribing an infinite-dimensional optimization problem to a finite-dimensional optimization problem.

*Assistant Professor, Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL 32611.

The two different philosophies of indirect and direct methods have led to a dichotomy in the optimal control community. Researchers who focus on indirect methods are interested largely in differential equation theory (e.g., see Ref. 29), while researchers who focus on direct methods are interested more in optimization techniques. While seemingly unrelated, these two approaches have much more in common than initially meets the eye. Specifically, as we will discuss later in this survey, in recent years researchers have delved quite deeply into the connections between the indirect and direct forms. This research has uncovered that the optimality conditions from many direct methods have a well-defined meaningful relationship. Thus, these two classes of methods are merging as time goes by.

Several previous works that provide surveys on optimal control have been published in the open literature. Ref. 30 provides an excellent overview of computational techniques that were used prior to the advent of the modern digital computer. Ref. 31 provides a summary of advances to that point on the use of gradient-based methods for solving optimal control problems. Ref. 32 provides a history of optimal control from 1950 to 1985, including an elegant historical perspective on the roots of the calculus of variations dating all the way back to the 1600s.* Ref. 34 provides a brief list of commonly used methods for optimal control developed prior to the 1990s and emphasizes the usefulness of combining indirect methods and direct methods (called a *hybrid approach*). Ref. 35 provides a brief description of methods for converting a continuous-time optimal control problem to a parameter optimization problem. Ref. 36 provides an excellent survey numerical methods for trajectory optimization, discussing both indirect and direct collocation methods. In addition, Ref. 36 gives an excellent perspective on trajectory optimization applications and development at government and research centers throughout the United States. This paper is considered complementary to all of the previously published survey articles on optimal control and trajectory optimization because it reflects the research that has been done over the past decade in computational optimal control while simultaneously providing a summary of the vast amount of work that was done prior to the 1990s. Finally, while a great deal of optimal control research has been done in the aerospace community, this paper attempts to draw from work that has been done in other engineering disciplines (e.g., chemical engineering) and from work in applied mathematics.

GENERAL FORMULATION OF AN OPTIMAL CONTROL PROBLEM

An optimal control problem is posed formally as follows. Determine the *state* (equivalently, the *trajectory* or *path*), $\mathbf{x}(t) \in \mathbb{R}^n$, the *control* $\mathbf{u}(t) \in \mathbb{R}^m$, the vector of static parameters $\mathbf{p} \in \mathbb{R}^q$, the initial time, $t_0 \in \mathbb{R}$, and the terminal time, $t_f \in \mathbb{R}$ (where $t \in [t_0, t_f]$ is the independent variable) that optimizes the performance index

$$J = \Phi[\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f; \mathbf{p}] + \int_{t_0}^{t_f} \mathcal{L}[\mathbf{x}(t), \mathbf{u}(t), t; \mathbf{p}]dt \tag{1}$$

subject to the dynamic constraints (i.e., *differential equation constraints*),

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

the *path constraints*

$$\mathbf{C}_{\min} \leq \mathbf{C}[\mathbf{x}(t), \mathbf{u}(t), t; \mathbf{p}] \leq \mathbf{C}_{\max}, \tag{3}$$

and the boundary conditions

$$\boldsymbol{\phi}_{\min} \leq \boldsymbol{\phi}[\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f; \mathbf{p}] \leq \boldsymbol{\phi}_{\max} \tag{4}$$

The state, control, and static parameter can each be written in component form as

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ \vdots \\ x_n(t) \end{bmatrix} ; \quad \mathbf{u}(t) = \begin{bmatrix} u_1(t) \\ \vdots \\ u_m(t) \end{bmatrix} ; \quad \mathbf{p} = \begin{bmatrix} p_1 \\ \vdots \\ p_q \end{bmatrix} \tag{5}$$

*The history of the calculus of variations is also captured in the beautiful monograph by Goldstine.33

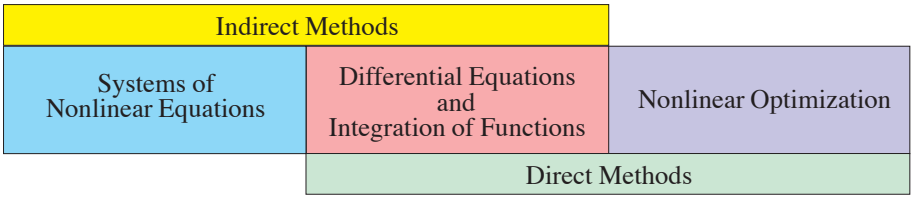

Figure 1: The Three Major Components of Optimal Control and the Class of Methods that Uses Each Component.

The differential equation of Eq. (2) describes the dynamics of the system while the performance index is a measure of the "quality" of the trajectory. When it is desired to minimize the performance index, a lower value of J is "better"; conversely, when it is desired to maximize the performance index, a higher value of J is "better."

Typically, an optimal control problem is divided into *phases*36 $p \in [1, \ldots, P]$ and the phases are connected or linked36 in some meaningful way. A multiple-phase optimal control problem is posed as follows. Optimize the cost

$$J = \sum_{k=1}^{P} J^{(k)} \tag{6}$$

[where the cost in each phase, $J^{(k)}$, $(k = 1, \ldots, P)$ has the form given in Eq. (1)] subject to the dynamic constraints

$$\dot{\mathbf{x}}^{(k)}(t) = \mathbf{f}\left(\mathbf{x}^{(k)}(t), \mathbf{u}^{(k)}(t), \mathbf{p}^{(k)}, t\right), \tag{7}$$

the *boundary conditions*,

$$\boldsymbol{\phi}_{\min}^{(k)} \leq \boldsymbol{\phi}^{(k)}\left(\mathbf{x}^{(k)}(t_0^{(k)}), t_0^{(k)}, \mathbf{x}^{(k)}(t_f^{(k)}), \mathbf{p}^{(k)}, t_f^{(k)}\right) \leq \boldsymbol{\phi}_{\max}^{(k)}, \tag{8}$$

the algebraic *path constraints*

$$\mathbf{C}_{\min}^{(k)} \leq \mathbf{C}^{(k)}\left(\mathbf{x}^{(k)}(t), \mathbf{u}^{(k)}(t), \mathbf{p}^{(k)}, t\right) \leq \mathbf{C}_{\max}^{(k)} \tag{9}$$

and the *linkage constraints*

$$\mathbf{L}_{\min}^{(s)} \leq \mathbf{L}\left(\mathbf{x}^{(l_s)}\left(t_f^{(l_s)}\right), \mathbf{u}^{(l_s)}\left(t_f^{(l_s)}\right), \mathbf{p}^{(l_s)}, t_f^{(l_s)}, \mathbf{x}^{(r_s)}(t_f^{(r_s)}), \mathbf{u}^{(r_s)}\left(t_f^{(r_s)}\right), \mathbf{p}^{(r_s)}, t_f^{(r_s)}\right) \leq \mathbf{L}_{\max}^{(s)}. \tag{10}$$

In Eq. (10) the parameter S is the number of pairs of phases to be linked, $r_s \in [1, \ldots, S]$ and $l_s \in [1, \ldots, S]$ are the *right phases* and *left phases*, respectively, of the linkage pairs, $r_s \neq l_s$ (implying that a phase cannot be linked to itself), and $s \in [1, \ldots, S]$.

NUMERICAL METHODS USED IN OPTIMAL CONTROL

At the heart of a well-founded method for solving optimal control problems are the following three fundamental components: (1) methods for solving the differential equations and integrating functions; (2) a method for solving a system of nonlinear algebraic equations; and (3) a method for solving a nonlinear optimization problem. Methods for solving differential equations and integrating functions are required for all numerical methods in optimal control. In an indirect method, the numerical solution of differential equations is combined with the numerical solution of systems of nonlinear equations while in a direct method the numerical solution of differential equations is combined with nonlinear optimization. A schematic with the breakdown of the components used by each class of optimal control methods is shown in Figure 1.

NUMERICAL SOLUTION OF DIFFERENTIAL EQUATIONS

Consider the *initial-value problem*[37–39] (**IVP**)

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), t), \qquad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{11}$$

Furthermore, consider a time interval $[t_i, t_{i+1}]$ over which the solution to the differential equation is desired. In other words, given the value of the state at t_i, $\mathbf{x}(t_i) \equiv \mathbf{x}_i$, it is desired to obtain the state at t_{i+1}, $\mathbf{x}(t_{i+1}) \equiv \mathbf{x}_{i+1}$. Integrating Eq. (11), we can write

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \int_{t_i}^{t_{i+1}} \dot{\mathbf{x}}(s)ds = \mathbf{x}_i + \int_{t_i}^{t_{i+1}} \mathbf{f}(\mathbf{x}(s), s)ds \tag{12}$$

We will now consider two approaches for solving the differential equation: time-marching and collocation.

Time-Marching

In a time-marching method, the solution of the differential equation at each time step t_k is obtained sequentially using current and/or previous information about the solution. Time-marching methods are divided into two categories: (1) multiple-step methods and (2) multiple-stage methods.

Multiple-Step Methods In a *multiple-step method*, the solution at time t_{k+1} is obtained from a defined set of previous values $t_{k-j}, \ldots, t_k$ where j is the number of steps. The simplest multiple-step method is a *single-step method* (where $j = 1$). The most common single-step methods are *Euler methods*. Euler methods have the general form

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h_k[\theta \mathbf{f}_k + (1-\theta)\mathbf{f}_{k+1}] \tag{13}$$

where $\mathbf{f}_k = \mathbf{f}[\mathbf{x}(t_k), \mathbf{u}(t_k), t_k]$ and $\theta \in [0, 1]$. The values $\theta = (1, 1/2, 0)$ correspond, respectively, to the particular Euler methods called *Euler forward*, *Crank-Nicolson*, and *Euler backward*. More complex multiple-step methods involve the use of more than one previous time point (i.e., $j > 1$). The two most commonly used multiple-step methods are the *Adams-Bashforth* and *Adams-Moulton* multiple-step methods[38]

$$\text{Adams-Bashforth:} \qquad \mathbf{x}_{k+1} = \mathbf{x}_k + h_k \left[\mathbf{f}_k + \tfrac{1}{2}\nabla\mathbf{f}_k + \tfrac{5}{12}\nabla^2\mathbf{f}_k + \tfrac{3}{8}\nabla^3\mathbf{f}_k + \tfrac{251}{720}\nabla^4\mathbf{f}_k + \cdots\right] \tag{14}$$

$$\text{Adams-Moulton:} \qquad \mathbf{x}_{k+1} = \mathbf{x}_k + h_k \left[\mathbf{f}_k - \tfrac{1}{2}\nabla\mathbf{f}_k - \tfrac{1}{12}\nabla^2\mathbf{f}_k - \tfrac{1}{24}\nabla^3\mathbf{f}_k - \tfrac{19}{720}\nabla^4\mathbf{f}_k + \cdots\right] \tag{15}$$

and ∇ is the backward difference formula,

$$\nabla\mathbf{f}_k = \mathbf{f}_k - \mathbf{f}_{k-1} \tag{16}$$

Euler backward and Crank-Nicolson are examples of *implicit methods* [because the value $\mathbf{x}(t_{k+1})$ appears implicitly on the right-hand side of Eq. (13)] whereas Euler forward is an example of an *explicit method* [because the value $\mathbf{x}(t_{k+1})$ does not appear on the right-hand side of Eq. (13)]. When employing implicit method, the solution at t_{k+1} is obtained using a *predictor-corrector* where the predictor is typically an explicit method (e.g., Euler-forward) while the corrector is the implicit formula. Implicit methods methods are more stable than explicit methods,[37] but an implicit method requires more computation at each step due to the need to implement a predictor-corrector.

Multiple-Stage Methods Suppose now that we divide the interval $[t_i, t_{i+1}]$ into K subintervals $[\tau_j, \tau_{j+1}]$ where

$$\tau_j = t_i + h_i\alpha_j, \quad (j = 1, \ldots, K), \quad h_i = t_{i+1} - t_i, \tag{17}$$

and $0 \leq \alpha_j \leq 1$, $(j = 1, \ldots, K)$. Each value τ_j is called a *stage* . The integral from t_i to t_{i+1} can be approximated via a *quadrature* as

$$\int_{t_i}^{t_{i+1}} \mathbf{f}(\mathbf{x}(s), s)ds \approx h_i \sum_{j=1}^{K} \beta_j \mathbf{f}(\mathbf{x}_j, \tau_j) \tag{18}$$

where $\mathbf{x}_j \equiv \mathbf{x}(\tau_j)$. It is seen in Eq. (18) that the values of the state at each stage are required in order to evaluate the quadrature approximation. These intermediate values are obtained as

$$\mathbf{x}(\tau_j) - \mathbf{x}(t_i) = \int_{t_i}^{\tau_j} \mathbf{f}(\mathbf{x}(s), s)ds \approx h_i \sum_{l=1}^{K} \gamma_{jl} \mathbf{f}(\mathbf{x}_l, \tau_l) \tag{19}$$

If we then combine Eqs. (18) and (19), we obtain the family of $K-$stage *Runge-Kutta* methods[36,38–43]

$$\begin{aligned} \int_{t_i}^{t_{i+1}} \mathbf{f}(\mathbf{x}(s), s)ds &\approx h_i \sum_{j=1}^{K} \beta_j \mathbf{f}_{ij} \\ \mathbf{f}_{ij} &= \mathbf{f}\left(\mathbf{x}_i + h_i \sum_{l=1}^{K} \gamma_{jl} \mathbf{f}_{il}, t_i + h_i \alpha_j\right) \end{aligned} \tag{20}$$

A Runge-Kutta method is a *time-marching* algorithm where the solution at the end of each step is obtained using the solution at the beginning of the step plus an approximation to the integral across the step. A Runge-Kutta method can be captured succinctly using the well-known *Butcher array*,[40,41]

$$\begin{array}{c|ccc} \alpha_1 & \gamma_{11} & \cdots & \gamma_{1K} \\ \vdots & \vdots & & \vdots \\ \alpha_K & \gamma_{K1} & \cdots & \gamma_{KK} \\ \hline & \beta_1 & \cdots & \beta_K \end{array}$$

Using the aforementioned definitions of an explicit and implicit method, a Runge-Kutta method is called *explicit* if $\gamma_{jl} = 0$ for all $l \geq j$ and is called *implicit* otherwise. In an explicit Runge-Kutta method, the approximation at t_{k+1} is computed using information prior to t_{k+1} whereas in an implicit Runge-Kutta method $\mathbf{x}(t_{k+1})$ is required in order to determine the solution at t_{k+1}. In the latter case, the solution is updated using a predictor-corrector approach.

As it turns out, the Euler methods described in Section are also Runge-Kutta methods of first-order. Typically, higher-order Runge-Kutta methods are employed. The most well-known Runge-Kutta method is the *classical Runge-Kutta method*, given as

$$\begin{aligned} \mathbf{k}_1 &= h_i \mathbf{f}_i \\ \mathbf{k}_2 &= h_i \mathbf{f}\left(\mathbf{x}_i + \tfrac{h_i}{2}\mathbf{k}_1, t_i + \tfrac{h_i}{2}\right) \\ \mathbf{k}_3 &= h_i \mathbf{f}\left(\mathbf{x}_i + \tfrac{h_i}{2}\mathbf{k}_2, t_i + \tfrac{h_i}{2}\right) \\ \mathbf{k}_4 &= h_i \mathbf{f}\left(\mathbf{x}_i + h_i \mathbf{k}_3, t_i + h_i\right) \\ \mathbf{x}_{i+1} &= \mathbf{x}_i + \tfrac{1}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right) \end{aligned} \tag{21}$$

Another well-known Runge-Kutta scheme is the *Hermite-Simpson* method

$$\begin{aligned} \bar{\mathbf{x}} &= \tfrac{1}{2}(\mathbf{x}_i + \mathbf{x}_{i+1}) + \tfrac{h_i}{8}(\mathbf{f}_i - \mathbf{f}_{i+1}) \\ \bar{\mathbf{f}} &= \mathbf{f}(\bar{\mathbf{x}}, t_i + \tfrac{h_i}{2}) \\ \mathbf{x}_{i+1} &= \mathbf{x}_i + \tfrac{h_i}{6}(\mathbf{f}_i + 4\bar{\mathbf{f}} + \mathbf{f}_{i+1}) \end{aligned} \tag{22}$$

The classical Runge-Kutta scheme is a *fourth-order method* while the Hermite-Simpson scheme is a *third-order methods*. The Butcher arrays for the three Euler methods, the classical Runge-Kutta method, and the Hermite-Simpson method are shown in Figure 2.

Collocation

Another way to solve differential equations is as follows. Suppose over a subinterval $[t_i, t_{i+1}]$ we choose to approximate the state using the following K^{th}-degree piecewise polynomial:

$$\mathbf{X}(t) \approx \sum_{k=0}^{K} \mathbf{a}_i (t - t_i)^k, \quad t \in [t_i, t_{i+1}] \tag{23}$$

0	0
	1

(a) Euler Forward

0	0	0
	0	1

(b) Euler Backward

0	0	0
	1/2	1/2

(c) Crank-Nicolson

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/3	1/3	1/6

(d) Classical Runge-Kutta

0	0	0	0
1/2	5/24	1/3	-1/24
1	1/6	2/3	1/6
	1/6	2/3	1/6

(e) Hermite-Simpson Method

Figure 2: Butcher Arrays for Common Single-Stage and Multiple-Stage Methods for Solving Initial-Value Problems.

Suppose further that the coefficients $(a_0, \ldots, a_K)$ of the piecewise polynomial are chosen to match the value of the function at the beginning of the step, i.e.,

$$\mathbf{X}(t_i) = \mathbf{x}(t_i) \tag{24}$$

Finally, suppose we choose to match the derivative of the state at the points defined in Eq. (17), i.e.,

$$\dot{\mathbf{X}}(\tau_j) = \mathbf{f}(\mathbf{x}(\tau_j), \tau_j), \quad (j = 1, \ldots, K) \tag{25}$$

Equation (25) is called a *collocation condition* because the approximation to the derivative is set equal to the right-hand side of the differential equation evaluated at each of the intermediate points $(\tau_1, \ldots, \tau_K)$. Collocation methods fall into three general categories:[36] Gauss methods, Radau methods, and Lobatto methods. In a *Gauss method*, neither of the endpoints t_k or t_{k+1} are collocation points. In a *Radau method*, at most one of the endpoints t_k or t_{k+1} is a collocation point. In a *Lobatto method*, both of the endpoints t_k and t_{k+1} are collocation points.

Examining the general form of Eq. (20) , it is seen that Euler and Runge-Kutta methods can be thought of equivalently as either time-marching or collocation methods. When an Euler or a Runge-Kutta method is employed in the form of collocation, the differential equation is said to be solved *simultaneously* because all of the unknown parameters are determined at the same time. Furthermore, collocation methods are said to simulate the dynamics of the system *implicitly* because the values of the state at each collocation point are obtained at the same time (as opposed to solving for the state sequentially as in a time-marching method). In order to implement simultaneous simulation, the discretized dynamics are written as *defect constraints* of the form

$$\boldsymbol{\zeta}_j = \dot{\mathbf{X}}(\tau_j) - \mathbf{f}(\mathbf{x}(\tau_j), \tau_j) \tag{26}$$

The defect constraints can then be stacked into a matrix as

$$\mathbf{Z} = \begin{bmatrix} \boldsymbol{\zeta}_1 \\ \vdots \\ \boldsymbol{\zeta}_M \end{bmatrix} \tag{27}$$

where $\mathbf{Z}$ is a function of the coefficients $(\mathbf{a}_0^{(k)}, \ldots, \mathbf{a}_K^{(k)}),\ (k = 1, \ldots, K)$. The objective then is to determine the values of these coefficients that result in $\mathbf{Z} = \mathbf{0}$. As an example, the defect constraints for the Crank-Nicolson method are given as

$$\boldsymbol{\zeta}_k = \mathbf{x}_{k+1} - \mathbf{x}_k - \frac{h_k}{2}\left(\mathbf{f}_k + \mathbf{f}_{k+1}\right) \tag{28}$$

In collocation (i.e., implicit simulation) it is desired to find a solution such that all of the defect constraints are zero. Finally, one of the key differences between collocation and time-marching is that in collocation it is

not necessary to use a predictor-corrector because the values of the state at each discretization point are being solve for simultaneously.

A subset of collocation methods that have seen extensive use in optimal control are *orthogonal collocation methods*. The key difference between an orthogonal collocation method and a standard collocation method is the manner in which the collocation points are chosen. Specifically, in an orthogonal collocation method the collocation points are the roots of a polynomial that is a member of a family of orthogonal polynomials, and the polynomial is associated with a quadrature rule for approximating the definite integral of a known function.[38,39] Common collocation points in orthogonal collocation are those obtained from the roots of either Chebyshev polynomials or Legendre polynomials. Furthermore, the state in an orthogonal collocation method is typically approximated on the time interval $\tau \in [-1, 1]$ as

$$\mathbf{x}(\tau) \approx \mathbf{X}(\tau) = \sum_{k=1}^{N} \mathbf{c}_k L_k(\tau) \tag{29}$$

where the functions $L_k(\tau), \ (k = 1, \ldots, N)$ are Lagrange polynomials.[38,39] It is known that Lagrange polynomials satisfy the *isolation property*,

$$L_k(\tau_i) = \begin{cases} 1 & , \quad k = j \\ 0 & , \quad k \neq j \end{cases} . \tag{30}$$

Equation (29) together with the isolation property leads to the fact that

$$\mathbf{c}_k = \mathbf{x}(\tau_k)$$

The use of orthogonal collocation to solve differential equations was originally introduced in Ref. 44. The benefit of using orthogonal collocation over standard collocation is that the quadrature approximation to a definite integral is extremely accurate when the collocation points are chosen in an orthogonal manner. For example, consider the approximation of the integral of a scalar function $g(\tau)$ using an N-point Legendre-Gauss (LG) quadrature,[38,39,45]

$$\int_{-1}^{1} g(\tau) d\tau \approx \sum_{k=1}^{N} w_k g(\tau_k) \tag{31}$$

where $\tau_k, \ (k = 1, \ldots, N)$ are the *Legendre-Gauss points* and are the roots of the N^{th}-degree Legendre polynomial,[38,39] $P_N(\tau)$. It is known that Eq. (31) will be exact if $g(\tau)$ is a polynomial of degree $2N - 1$ or less (where it is observed that only N points are used in the quadrature approximation). Thus, the rationale for using orthogonal collocation is that the accuracy obtained is significantly higher than the number of collocation points used. Following on a similar approach, other highly accurate quadrature rules include *Legendre-Gauss-Radau* (LGR, exact for a polynomial of degree $2N - 2$ where N is the number of LGR points) and *Legendre-Gauss-Lobatto* (LGL) (LGL, exact for a polynomial of degree $2N - 3$ where N is the number of LGL points). It is noted that the LGR points are the N roots of the polynomial $P_{N-1}(\tau) + P_N(\tau)$ while the LGL points are the roots of $\dot{P}_N(\tau)$ together with the points $\tau = -1$ and $\tau = 1$.

Integration of Functions

Because the objective is to solve an optimal control problem, it is necessary to approximate the cost function of Eq. (1). Typically, the cost is approximated using an quadrature that is consistent with the numerical method for solving the differential equation. For example, if one is using an Euler-forward rule for solving the differential equation, the cost would also be approximated using Euler-forward integration. In the case of an orthogonal collocation method, the integration rule is an orthogonally collocated quadrature rule [e.g., if one is using Legendre-Gauss points, then the Lagrange cost is approximated using a Gauss quadrature as given in Eq. (31)]. The requirement for consistency in the approximation of the differential equations and the cost can be thought of in another manner. Any Bolza problem can be converted to a Mayer problem by adding a state x_{n+1} and adding the differential equation

$$\dot{x}_{n+1} = \mathcal{L}[\mathbf{x}(t), \mathbf{u}(t), t; \mathbf{p}] \tag{32}$$

with the initial condition

$$x_{n+1}(t_0) = 0 \tag{33}$$

Then the cost function of Eq. (1) would be given in Mayer form as

$$J = \Phi[\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f; \mathbf{p}] + x_{n+1}(t_f) \tag{34}$$

A common scheme would then be used to solve the augmented system of differential equations

$$\begin{array}{rcl} \dot{\mathbf{x}}(t) & = & \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), t; \mathbf{p}] \\ \dot{x}_{n+1} & = & \mathcal{L}[\mathbf{x}(t), \mathbf{u}(t), t; \mathbf{p}] \end{array} \tag{35}$$

Converting back to Bolza form, the quadrature approximation to the term

$$\int_{t_0}^{t_f} \mathcal{L}[\mathbf{x}(t), \mathbf{u}(t), t; \mathbf{p}]dt$$

in Eq. (1) must be the same as the scheme used to solve the system of differential equations given in Eq. (35).

NONLINEAR OPTIMIZATION

A key ingredient to solving optimal control problems is the ability to solve *nonlinear optimization* or *nonlinear programming problems*[19–21,46] (NLPs). An NLP takes the following general mathematical form. Determine the vector of decision variables $\mathbf{z} \in \mathbb{R}^n$ that minimize the cost function

$$f(\mathbf{z}) \tag{36}$$

subject to the algebraic constraints

$$\mathbf{g}(\mathbf{z}) = \mathbf{0} \tag{37}$$

$$\mathbf{h}(\mathbf{z}) \leq \mathbf{0} \tag{38}$$

where $\mathbf{g}(\mathbf{z}) \in \mathbb{R}^m$ and $\mathbf{h}(\mathbf{z}) \in \mathbb{R}^p$. The first-order optimality conditions of the NLP given in Eqs. (36) and (37), also known as the *Karush-Kuhn-Tucker conditions*,[19–21] are given as

$$g_i(\mathbf{z}) = 0, \quad (i = 1, \ldots, m) \tag{39}$$

$$h_i(\mathbf{z}) \leq 0, \quad (i = 1, \ldots, p) \tag{40}$$

$$\nu_i \geq 0, \quad (i = 1, \ldots, p) \tag{41}$$

$$\nu_i h_i(\mathbf{z}) = 0, \quad (i = 1, \ldots, p) \tag{42}$$

$$\nabla f(\mathbf{z}) + \sum_{i=1}^{m} \lambda_i \nabla g_i(\mathbf{z}) + \sum_{i=1}^{p} \nu_i \nabla h_i(\mathbf{z}) = \mathbf{0} \tag{43}$$

The NLP may either be *dense* (i.e., a large percentage of the derivatives of the objective function and the constraint functions with respect to the components of $\mathbf{z}$ are nonzero) or may be *sparse* (i.e., a large percentage of the derivatives of the objective function and the constraint functions with respect to the components of $\mathbf{z}$ are zero). Dense NLPs typically are small (consisting of at most a few hundred variables and constraints) while sparse NLPs are often extremely large (ranging anywhere from thousands of variables and constraints to millions of variables and constraints). Numerical methods for solving NLPs fall into categories: *gradient-based methods* and *heuristic methods*. Each approach is now described.

Gradient Methods

In a gradient-based method, an initial guess is made of the unknown decision vector $\mathbf{z}$. At the k^{th} iteration, a *search direction*, $\mathbf{p}_k$, and a *step length*, α_k, are determined. The search direction provides a direction in $\mathbb{R}^n$ along which to change the current value $\mathbf{z}_k$ while the step length provides the magnitude of the change to $\mathbf{z}_k$. The update from $\mathbf{z}_k$ to $\mathbf{z}_{k+1}$ then has the form

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \alpha_k \mathbf{p}_k \tag{44}$$

In the case of minimization, the search direction is chosen to "sufficiently decrease" the objective function in the form

$$f(\mathbf{z}_{k+1}) \leq f(\mathbf{z}_k) + K\alpha_k \nabla f^{\mathsf{T}}(\mathbf{z}_k)\mathbf{p}_k \tag{45}$$

and K is a parameter between 0 and 1. The most commonly used gradient-based NLP solution methods are *sequential quadratic programming*[23–25,46] (SQP) and *interior-point* (IP) or *barrier methods*. For an equality constrained problem, the search direction in an SQP method is determined by solving the *quadratic program* (QP) problem

$$\begin{array}{ll} \min\limits_{\mathbf{p}} & \frac{1}{2}\mathbf{p}^{\mathsf{T}}\mathbf{W}\mathbf{p} + \nabla f^{\mathsf{T}}(\mathbf{z}_k)\mathbf{p} \\ \text{s.t.} & \nabla c_i^{\mathsf{T}}(\mathbf{z}_k)\mathbf{p} - c_i(\mathbf{z}_k) = \mathbf{0}, \quad (i \in \mathcal{A}) \\ & \nabla c_i^{\mathsf{T}}(\mathbf{z}_k)\mathbf{p} - c_i(\mathbf{z}_k) = \mathbf{0}, \quad (i \in \mathcal{I}) \end{array} \tag{46}$$

where $\mathcal{A}$ and $\mathcal{I}$ are the active and inactive constraint sets, respectively, at the current iteration and $\mathbf{W}_k$ is a positive semi-definite approximation of the Hessian of the Lagrangian, $\mathcal{L}$, where $\mathcal{L}$ defined as

$$\mathcal{L}(\mathbf{z}, \boldsymbol{\lambda}) = f(\mathbf{z}) - \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{c}(\mathbf{z}) \tag{47}$$

Commonly, the matrix $\mathbf{W}_k$ is obtained using a *quasi-Newton* approximation in the form of an *update method*. The two most well-known update methods are the *Broyden-Fletcher-Goldfarb-Shanno*[47–50] The BFGS update for the case of an unconstrained optimization problem is given as follows. At the first step in the optimization, $\mathbf{W}_k$ is generally set to the identity matrix. Then, at each step in the optimization, $\mathbf{W}_k$ is updated as

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \frac{\mathbf{v}_k \mathbf{v}_k^{\mathsf{T}}}{\mathbf{v}_k^{\mathsf{T}}\mathbf{s}_k} - \frac{\mathbf{W}_K \mathbf{s}_k \mathbf{s}_k^{\mathsf{T}} \mathbf{W}_k}{\mathbf{s}_k^{\mathsf{T}}\mathbf{W}_k\mathbf{s}_k} \tag{48}$$

where $\mathbf{v}_k = \nabla f(\mathbf{z}_{k+1}) - \nabla f(\mathbf{z}_k)$, $\mathbf{z}_{k+1} = \mathbf{z}_k + \alpha_k \mathbf{s}_k$, and $\mathbf{s}_k$ is obtained by solving the equation

$$\mathbf{W}_k \mathbf{s}_k = -\nabla f(\mathbf{z}_k) \tag{49}$$

In addition to using a quasi-Newton approximation for the Hessian of the Lagrangian, another common approach, that ensures global convergence to a solution, is the use of a *merit function*.[46] Essentially, a merit function combines the objective function with the infeasibilities in the constraints. As an example, in an interior point method a *barrier function* is used. Often, the barrier function has the *log barrier form*,

$$B(\mathbf{z}) = f(\mathbf{z}) - \mu \sum_{i=1}^{m} \log(c_i(\mathbf{z})) \tag{50}$$

where μ is a positive parameter. The barrier function is then minimized for a sequence of optimization problems where $\mu \downarrow 0$ (i.e., μ is decreased to zero from above).

Extensive research has been done over the past two decades in SQP and interior point methods. This research has led to extremely versatile and robust software programs for the numerical solution of NLPs. Examples of well-known software that use SQP methods include the dense NLP solver *NPSOL*[23] and the sparse NLP solvers *SNOPT*[24,25] and *SPRNLP*.[51] Well known sparse interior point NLP solvers include *BARNLP*,[27] *LOQO*,[52] *KNITRO*,[26,53] and *IPOPT*.[54–56] These NLP solvers are available in a variety of implementations including FORTRAN, C, MATLAB®, and AMPL.[57]

Heuristic Optimization Methods

A *heuristic optimization method* is fundamentally different from a gradient method. First, a gradient method is a *local method* in that, upon convergence, a locally optimal solution will generally be obtained. A heuristic method, on the other hand, is a *global* technique.[58] An excellent summary of various heuristic methods can be found in Refs. 59–61 and 62. The main idea behind a heuristic optimization method is that the search is performed in a stochastic manner as opposed to a deterministic manner.

One well known class of heuristic methods is the class of *genetic algorithms*. A genetic algorithm is an *evolutionary approach*[63] Essentially, genetic algorithms emulate on a computer mimic evolutionary processes in genetics.[63] Specifically, an initial population of possible solutions to a problem is chosen. Furthermore, each solution has a particular *fitness* that reflects the quality of that particular gene. These genes are then recombined (i.e., they are mutated) via a crossover mechanism which results in future generations of populations. Continuing generationally, those genes with the highest fitness survive to the later generations.

The following five components are fundamental to all GAs:[63] encoding, fitness, selection, crossover, and mutation. An *encoding mechanism* provides a means to represent the optimization variables in the problem. An appropriate encoding mechanism is chosen for the problem under consideration (e.g., variables whose values are real or are integers). For problems with continuous variables it is often the case that the encoding is done using integers where the number of integers used depends upon the number of significant figures used to represent the real numbers (e.g., for an interval [-1.34,1.34] we might use the integers [-134,134]). The *fitness function*, typically the objective function that is to be optimized, provides a mechanism for evaluating the quality of each string obtained by the encoding mechanism. Most often, the fitness functions are normalized to lie on the interval [0,1].[63] The *selection mechanism* is based on the concept of "survival-of-the-fittest"[63] In particular, solutions that are "more fit" are more likely to survive to the next generation as compared to solutions that are "less fit." Perhaps the most critical aspect of any genetic algorithm is the *crossover mechanism*. In crossover, pairs of strings are chosen at random and are crossed. This crossover results in a child string that has a different fitness (higher or lower) from either of its parents. In the case of a binary string representation, the crossed strings are subjected to *mutation* where certain bits in the string are changed at random.

A heuristic approach related to genetic algorithms is *Simulated Annealing*.[64,65] In simulated annealing, each point in the search space is analogous to a state of some physical process and the fitness function is analogous to an "internal energy" of the system. The goal is to minimize the internal energy of the system. Using a probabilistic approach, at each step a simulated annealing method decides whether to stay at the current state in the system or to move to a neighboring state, where the neighbors are specified by the user. The probability of choosing the current state or a neighboring state depends upon both the difference in the fitness values and a global *temperature* parameter.

Another heuristic approach is *particle swarm optimization*[66–68] (PSO). PSO is a population-based stochastic optimization method that is loosely based on the idea of swarms of animals (e.g., a flock of birds, a school of fish, or a swarm of bees). PSO shares many similarities with GAs in that the algorithm is given an initial population and optimal solutions are obtained by generational searches. PSO, however, has neither crossover or mutation. In a PSO, candidate solutions, which are called *particles*, move through the search space by following those particles that have the lowest cost at any iteration in the optimization. Each particle then keeps track of the best solution it has achieved during the process. Simultaneously, the best value of any neighbors to a particle particle is also tracked. The PSO then moves in a direction toward the particle's best solution and the neighbors best solution.

SYSTEMS OF NONLINEAR ALGEBRAIC EQUATIONS

Solving a system of nonlinear algebraic equations is equivalent to *root-finding*. In the case where all of the algebraic equations can be written as equalities, we have a problem of the form

$$\mathbf{g}(\mathbf{z}) = \mathbf{0} \tag{51}$$

Many methods for root-finding are limited to functions of a scalar variable (e.g., bracketing and bisection and the secant method[69]) and, thus, are limited in their functionality. The most common method for multi-dimensional root-finding is *Newton's method*.[69] In Newton's method, an initial guess is made of the vector $\mathbf{z}$. The iterates are then computed as

$$\mathbf{z}_{k+1} = \mathbf{z}_k - \left[\frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right]^{-1}_{\mathbf{z}_k} \mathbf{f}(\mathbf{z}_k) \tag{52}$$

It is well-known that the Newton's method converges when the initial guess is close to a root. Most systems of nonlinear equations, however, have multiple solutions. As a result, the root obtained will generally be in the vicinity of the initial guess and may not be the desired solution. Also, it is seen from Eq. (52) that Newton's method breaks down when the Jacobian of $\mathbf{f}$ (i.e., $\partial \mathbf{f}/\partial \mathbf{z}$) is singular. In order to ensure that the iterations do not become undefined, $\partial \mathbf{f}/\partial \mathbf{z}$ is often replaced with a quasi-Newton approximation approximation (e.g., a BFGS-type[47–50] approximation to the Jacobian). In fact, one way of using a quasi-Newton method to finding the root of a system via an NLP solver. In particular, Eq. (51) can be solved using an NLP solver such as *SNOPT*[25,46] by using a fictitious cost function such as

$$f(\mathbf{z}) = (\mathbf{z} - \mathbf{z}_0)^{\mathsf{T}}(\mathbf{z} - \mathbf{z}_0) \tag{53}$$

where $\mathbf{z}_0$ is the initial guess. The optimization problem is then to minimize Eq. (53) subject to the constraint of Eq. (51). The iterations will then proceed via the update procedure that is used in the NLP solver.

METHODS FOR SOLVING OPTIMAL CONTROL PROBLEMS

With the exception of simple problems (e.g., the infinite-horizon linear quadratic problem[70,71]), optimal control problems must be solved numerically. The need for solving optimal control problems numerically has given rise to a wide range of numerical approaches. These numerical approaches are divided into two broad categories: (1) indirect methods and (2) direct methods. The major methods that fall into each of these two broad categories are described in the next two sections.

INDIRECT METHODS

In an indirect method, the *calculus of variations*[7–17] is used to determine the first-order optimality conditions of the optimal control problem given in Eqs. (1)–(4). Unlike ordinary calculus (where the objective is to determine points that optimize a function), the calculus of variations is the subject of determining *functions* that optimize a *function of a function* (also known as *functional optimization*) . Applying the calculus of variations to the optimal control problem given in Eqs. (1)–(4) leads to the *first-order necessary conditions* for an extremal trajectory. These conditions are typically derived using the augmented Hamiltonian, $\mathcal{H}$, defined as

$$\mathcal{H}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{u}, t) = \mathcal{L} + \boldsymbol{\lambda}^{\mathsf{T}}\mathbf{f} - \boldsymbol{\mu}^{\mathsf{T}}\mathbf{C} \tag{54}$$

where $\boldsymbol{\lambda}(t) \in \mathbb{R}^n$ is the *costate* or *adjoint* and $\boldsymbol{\mu}(t) \in \mathbb{R}^c$ is the Lagrange multiplier associated with the path constraint. In the case of a single phase optimal control problem with no static parameters, the first-order optimality conditions of the continuous-time problem are given as follows:

$$\dot{\mathbf{x}} = \left[\frac{\partial \mathcal{H}}{\partial \boldsymbol{\lambda}}\right]^{\mathsf{T}} \quad , \quad \dot{\boldsymbol{\lambda}} = -\left[\frac{\partial \mathcal{H}}{\partial \mathbf{x}}\right]^{\mathsf{T}} \tag{55}$$

$$\mathbf{u}^* = \arg\min_{\mathbf{u}\in\mathcal{U}} \mathcal{H} \tag{56}$$

$$\boldsymbol{\phi}(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f) = \mathbf{0} \tag{57}$$

$$\boldsymbol{\lambda}(t_0) = -\frac{\partial \Phi}{\partial \mathbf{x}(t_0)} + \boldsymbol{\nu}^{\mathsf{T}}\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{x}(t_0)} \quad , \quad \boldsymbol{\lambda}(t_f) = \frac{\partial \Phi}{\partial \mathbf{x}(t_f)} - \boldsymbol{\nu}^{\mathsf{T}}\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{x}(t_f)} \tag{58}$$

$$\mathcal{H}(t_0) = \frac{\partial \Phi}{\partial t_o} - \boldsymbol{\nu}^{\mathsf{T}}\frac{\partial \boldsymbol{\phi}}{\partial t_0} \quad , \quad \mathcal{H}(t_f) = -\frac{\partial \Phi}{\partial t_f} + \boldsymbol{\nu}^{\mathsf{T}}\frac{\partial \phi}{\partial t_f} \tag{59}$$

$$\begin{aligned} \mu_j(t) = 0, \text{ when } \quad C_j(\mathbf{x}, \mathbf{u}, t) < 0, \quad j = 1, \ldots, c \\ \mu_j(t) \leq 0, \text{ when } \quad C_j(\mathbf{x}, \mathbf{u}, t) = 0, \quad j = 1, \ldots, c \end{aligned} \tag{60}$$

where $\mathcal{U}$ is the feasible control set and $\boldsymbol{\nu} \in \mathbb{R}^q$ is the Lagrange multiplier associated with the boundary condition $\boldsymbol{\phi}$. Because the dynamics of Eq. (55) arise from differentiation a Hamiltonian, Eq. (55) is called a *Hamiltonian system*.[10,11,72] Furthermore, Eq. (56) is known as *Pontryagin's Minimum Principle*[73] (PMP) and is a classical result to determine the optimal control. Finally, the conditions on the initial and final costate given in Eq. (58) are called *transversality conditions*[11–17,72,74] while the conditions on the Lagrange multipliers of the path constraints given in Eq. (60) are called *complementary slackness conditions*.[19–21] The Hamiltonian system, together with the boundary conditions, transversality conditions, and complementary slackness conditions, is called a *Hamiltonian boundary-value problem* (HBVP).[11,29,72] Any solution $(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\lambda}(t), \boldsymbol{\mu}(t), \boldsymbol{\nu})$ is called an *extremal* and consists of the state, costate, and any Lagrange multipliers that satisfy the boundary conditions and any interior-point constraints on the state and costate. In an indirect method extremal trajectories (i.e., solutions of the HBVP) are determined numerically. Because an indirect method requires solving a multiple-point boundary-value problem, the original optimal control problem is turned into the problem of solving a system of nonlinear equations of the form

$$\begin{aligned} \mathbf{f}(\mathbf{z}) &= \mathbf{0} \\ \mathbf{g}_{\min} \leq \mathbf{g}(\mathbf{z}) &\leq \mathbf{g}_{\max} \end{aligned} \tag{61}$$

The three two most common indirect methods are the shooting method, the multiple-shooting method, and collocation methods. Each of these approaches is now described.

Indirect Shooting Method

Perhaps the most basic indirect method is the *shooting method*.[75] In a typical shooting method, an initial guess is made of the unknown boundary conditions at one end of the interval. Using this guess, together with the known initial conditions, the Hamiltonian system Eq. (55) is integrated to the other end (i.e., either forward from t_0 to t_f or backward from t_f to t_0) using a method similar to that described in Section . Upon reaching t_f, the terminal conditions obtained from the numerical integration are compared to the known terminal conditions given in Eqs. (57) and (58) If the integrated terminal conditions differ from the known terminal conditions by more than a specified tolerance ϵ, the unknown initial conditions are adjusted and the process is repeated until the difference between the integrated terminal conditions and the required terminal conditions is less than some specified threshold. One can visualize the simple shooting method as follows. Suppose it is desired to determine the initial angle of a cannon so that, when a cannonball is fired, it strikes a desired target (see Figure 3). An initial guess is made for the angle of the cannon and the cannon is fired. If the cannon does does hit the target, the angle is adjusted (based on the amount of the miss) and another cannon is fired. The process is repeated until the target is hit. A schematic showing the steps in a typical implementation of a shooting method is shown in Algorithm 1.

Indirect Multiple-Shooting Method

While a simple shooting method is appealing due to its simplicity, it presents significant numerical difficulties due to ill-conditioning of the Hamiltonian dynamics. The reason for this ill-conditioning is that Hamiltonian systems have the property that the divergence of the flow of trajectories must be constant, i.e.,

$$\sum_{i=1}^{n} \left[\frac{\partial}{\partial x_i} \left(\frac{\partial \mathcal{H}}{\partial \lambda_i} \right) + \frac{\partial}{\partial \lambda_i} \left(-\frac{\partial \mathcal{H}}{\partial x_i} \right) \right] \equiv 0 \tag{62}$$

Equation (62) implies that, in a neighborhood of the optimal solution, there exist an equal number directions along which the solution will contract and expand and this expansion and contraction takes place at the same rate (the simultaneous expanding and contracting behavior is due to the fact that many Hamiltonian systems admit an exponential *dichotomy*[29]). As a result, errors made in the unknown boundary conditions will amplify as the dynamics are integrated in either direction of time. The shooting method poses particularly poor

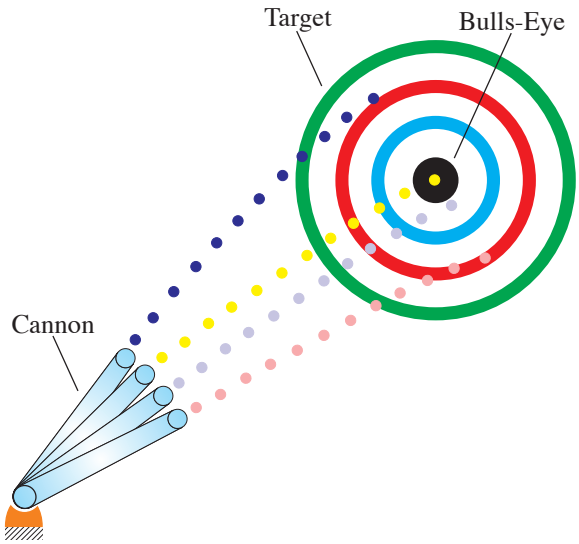

Figure 3: Schematic of Indirect Shooting Method Using the Analogy of a Cannon Firing a Cannonball to Strike a Target.

Input: Initial Guess of Unknown Initial Conditions

Output: State-Adjoint Extremal Trajectory

while *Error in Terminal Conditions is Larger Than Specified Tolerance* **do**

Integrate Trajectory from t_0 to t_f;

Compute Error in Terminal Conditions;

Update Unknown Initial Conditions;

end

Algorithm 1: Algorithm for Indirect Shooting Method.

characteristics when the optimal control problem is *hyper-sensitive*[76–80] (i.e., when time interval of interest is long in comparison with the time-scales of the Hamiltonian system in a neighborhood of the optimal solution).

In order to overcome the numerical difficulties of the simple shooting method, a modified method, called the *multiple-shooting method*,[39] has been developed. In a multiple-shooting method, the time interval $[t_0, t_f]$ is divided into $M+1$ subintervals. The shooting method is then applied over each subinterval $[t_i, t_{i+1}]$ with the initial values of the state and adjoint of the interior intervals being unknowns that need to be determined. In order to enforce continuity, the following conditions are enforced at the interface of each subinterval:

$$\mathbf{y}(t_i^-) = \mathbf{y}(t_i^+) \Longleftrightarrow \mathbf{y}(t_i^-) - \mathbf{y}(t_i^+) = \mathbf{0} \tag{63}$$

where $\mathbf{y}(t)$ is the combined state-costate vector, i.e.,

$$\mathbf{y}(t) = \left[\begin{array}{c} \mathbf{x}(t) \\ \boldsymbol{\lambda}(t) \end{array} \right].$$

The continuity conditions of Eq. (63) result in vector root-finding problem where it is desired to drive the values of the difference between $\mathbf{y}(t_i^-) - \mathbf{y}(t_i^+)$ to zero. It is seen that the multiple-shooting method requires

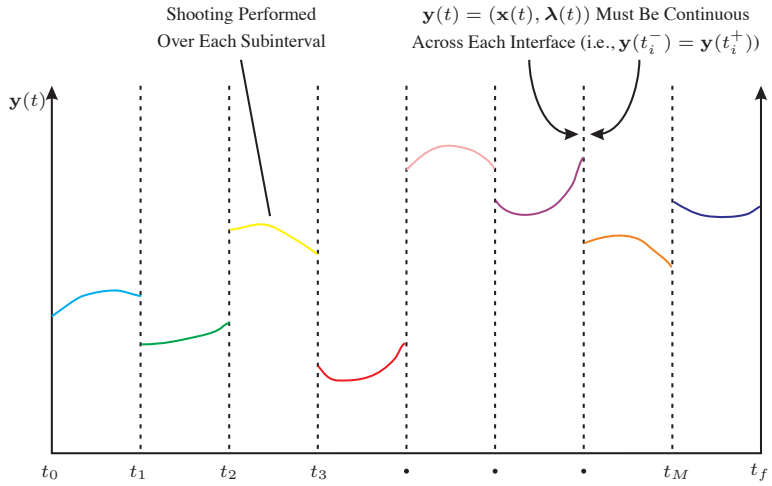

Figure 4: Schematic of the Indirect Multiple-Shooting Method.

extra variables be introduced into the problem (i.e., the values of the state and adjoint at the interface points). Despite the increased size of the problem due to these extra variables, the multiple-shooting method is an improvement over the standard shooting method because the sensitivity to errors in the unknown initial conditions is reduced because integration is performed over significantly smaller time intervals. Nevertheless, even multiple-shooting can present issues if a sufficiently good guess of the costate is not used.[81] A schematic of the multiple-shooting method is shown in Figure 4.

Indirect Collocation Methods

In an indirect collocation method, the state and costate are parameterized using piecewise polynomials as described in Section . The collocation The collocation procedure leads to a root-finding problem where the vector of unknown coefficients $\mathbf{z}$ consists of the coefficients of the piecewise polynomial. This system of nonlinear equations is then solved using an appropriate root-finding technique.

DIRECT METHODS

Direct methods are fundamentally different from indirect methods. In a direct method, the state and/or control of the original optimal control problem are approximated in some appropriate manner. In the case where only the control is approximated, the method is called a *control parameterization method*.[82] When both the state and control are approximated the method is called a *state and control parameterization method*. In either a control parameterization method or a state and control parameterization method, the optimal control problem is transcribed to a *nonlinear optimization problem* or *nonlinear programming problem*[19–21,36,46] (NLP).

Direct Shooting Method

The most basic direct method for solving optimal control problems is the *direct shooting method*. The direct shooting method is a control parameterization method where the control is parameterized using a specified functional form, e.g.,

$$\mathbf{u}(t) \approx \sum_{i=1}^{m} \mathbf{a}_i \psi_i(t) \tag{64}$$

where $\psi(t)_i, \ (i = 1, \ldots, m)$ are known functions and $\mathbf{a}_i, \ (i = 1, \ldots, m)$ are the parameters to be determined from the optimization. The dynamics are then satisfied by integrating the differential equations using a time-marching algorithm such as that given in Section . Similarly, the cost function of Eq. (1) is determined

using a quadrature approximation that is consistent with the numerical integrator used to solve the differential equations. The NLP that arises from direct shooting then minimizes the cost subject to any path and interior-point constraints. A schematic of a direct shooting method is shown in Figure 2.

Input: Initial Guess of Parameters in Control Parameterization

Output: Optimal Values of Parameters and Optimal Trajectory

while *Cost is Not at a Minimum and Constraints Not Satisfied* **do**

Integrate Trajectory from t_0 to t_f;

Compute Error in Terminal Conditions;

Update Unknown Initial Conditions By Driving Cost to Lower Value;

end

Algorithm 2: Basic Algorithm of the Direct Shooting Method

Direct Multiple Shooting Method

In a manner similar to that for indirect methods, in a *direct multiple-shooting method*, the time interval $[t_0, t_f]$ is divided into $M+1$ subintervals. The aforementioned direct shooting method is then used over each subinterval $[t_i, t_{i+1}]$ with the values of the state at the beginning of each subinterval and the unknown coefficients in the control parameterization being unknowns in the optimization. In order to enforce continuity, the following conditions are enforced at the interface of each subinterval:

$$\mathbf{x}(t_i^-) = \mathbf{x}(t_i^+) \Longleftrightarrow \mathbf{x}(t_i^-) - \mathbf{x}(t_i^+) = \mathbf{0} \tag{65}$$

The continuity conditions of Eq. (65) result in vector root-finding problem where it is desired to drive the values of the difference between $\mathbf{x}(t_i^-) - \mathbf{x}(t_i^+)$ to zero. It is seen that the direct multiple-shooting method increases the size of the optimization problem because the values of the state at the beginning of each subinterval are parameters in the optimization. Despite the increased size of the problem due to these extra variables, the direct multiple-shooting method is an improvement over the standard direct shooting method because the sensitivity to errors in the unknown initial conditions is reduced because integration is performed over significantly smaller time intervals. A schematic of direct multiple-shooting method is shown in Figure 5.

Direct Collocation Methods

Arguably the most powerful methods for solving general optimal control problems are *direct collocation methods*. A direct collocation method is a state and control parameterization method where the state and control are approximated using a specified functional form. The two most common forms of collocation are *local collocation* and *global collocation*. A *local collocation method* follows a procedure similar to that of Section in that the time interval $[t_0, t_f]$ is divided into S subintervals $[t_{s-1}, t_s]$, $(s = 1, \ldots, S)$ where $t_S = t_f$. In order to ensure continuity in the state across subintervals, the following *compatibility constraint* is enforced at the interface of each subinterval:

$$\mathbf{x}(t_i^-) = \mathbf{x}(t_i^+), \quad (s = 2, \ldots, S-1) \tag{66}$$

In the context of optimal control, local collocation has been employed using one of two categories of discretization: Runge-Kutta methods and orthogonal collocation methods. In the case of Runge-Kutta, nearly all of the methods used are *implicit*[83–92] because the stability properties of implicit Runge-Kutta methods are better than those of explicit methods. The seminal work on orthogonal collocation methods in optimal control is due to Reddien,[93] where Legendre-Gauss points was used together with cubic splines. Following on

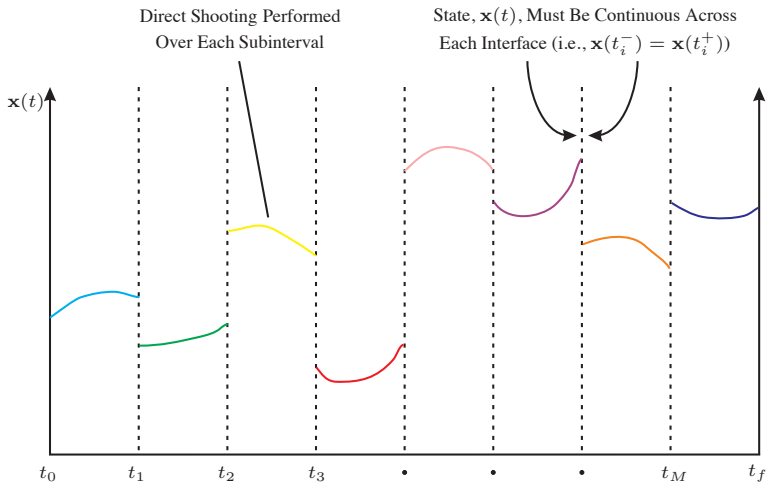

Figure 5: Schematic of the Direct Multiple-Shooting Method.

Reddien's work, Cuthrell and Biegler used LG points together with Lagrange polynomials.[94,95] Interestingly, Cuthrell[95] showed mathematically that the indirect transcription using LG points was equivalent to the KKT conditions obtained from the NLP of the direct formulation. In the 1990s, orthogonal collocation methods were developed using higher-order Gauss-Lobatto collocation methods.[96–99] Finally, the convergence rates of an orthogonal collocation method using Legendre-Gauss-Radau (LGR) points was studied.[100]

Generally, employing direct local collocation leads to a *large sparse* NLP, i.e., the NLP has thousands to tens of thousands of variables and a similar number of constraints. Because, however, the NLP is sparse, many of the derivatives of the constraint Jacobian are zero. This feature of local direct collocation makes it possible to solve such problem efficiently using appropriate NLP solvers such as *SNOPT*,[24,25] *SPRNLP*,[51] and *KNITRO*.[26]

Pseudospectral (Global Orthogonal Collocation) Methods

In recent years *pseudospectral methods* have increased in popularity. A pseudospectral method is a *global* form of orthogonal collocation, i.e., in a pseudospectral method the state is approximated using a *global* polynomial and collocation is performed at chosen points. Thus, as opposed to local collocation, where the degree of the polynomial is fixed and the number of segments (meshes) is varied, in a pseudospectral method the number of meshes is fixed and the degree of the polynomial is varied. As with a local orthogonal collocation method, the basis functions are typically Chebyshev or Lagrange polynomials. Pseudospectral methods were developed originally to solve problems in computational fluid dynamics[101,102] (CFD). The rationale for using a global polynomial with orthogonally collocated points is that the approximation will converge *spectrally* (i.e., at an exponential rate) as a function of the number of collocation points.

Pseudospectral methods were introduced introduced to the optimal control community in the 1990s and some of this early work that uses *Chebyshev points* (with the basis functions being Chebyshev polynomials) is given in Refs. 103–105. While Chebyshev polynomials can be used, they do not satisfy the isolation property given in Eq. (30) and, thus, lead to more complicated collocation conditions. As a result, the majority of pseudospectral methods have employed Lagrange polynomials as basis functions. The three different categories of pseudospectral methods that use Lagrange polynomials are now described.

Gauss-Lobatto Pseudospectral Methods The first major pseudospectral approach uses the family of *Gauss-Lobatto* points[106,107] (i.e., either Legendre-Gauss-Lobatto points or Chebyshev-Gauss-Lobatto points). The Gauss-Lobatto pseudospectral methods of Refs. 106–109 have the common feature that the state is approxi-

mated globally as

$$\mathbf{x}(\tau) \approx \mathbf{X}(\tau) = \sum_{i=1}^{N} \mathbf{x}(\tau_i) L_i(\tau) \tag{67}$$

where $(\tau_1, \ldots, \tau_N)$ are the N Gauss-Lobatto points which include both endpoints of the time interval. Furthermore, the cost is approximated using a Gauss-Lobatto quadrature,

$$J \approx \frac{t_f - t_0}{2} \sum_{i=1}^{N} w_i^{\mathsf{GL}} \mathcal{L}[\mathbf{x}(\tau_i), \mathbf{u}(\tau_i), \tau_i] \tag{68}$$

where $w_i^{\mathsf{GL}}, \ (i = 1, \ldots, N)$ are the Gauss-Lobatto weights. The resulting collocation equations then have the form

$$\mathbf{D}^{\mathsf{GL}} \mathbf{X}^{\mathsf{GL}} = \frac{t_f - t_0}{2} \mathbf{F}(\mathbf{X}^{\mathsf{GL}}, \mathbf{U}^{\mathsf{GL}}) \equiv \frac{t_f - t_0}{2} \mathbf{F}^{\mathsf{GL}} \tag{69}$$

where $\mathbf{D}^{\mathsf{GL}}$ is an $N \times N$ matrix and

$$\mathbf{X}^{\mathsf{GL}} = \begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_N \end{bmatrix} \quad , \quad \mathbf{F}^{\mathsf{GL}} = \begin{bmatrix} \mathbf{F}(\mathbf{X}_1, \mathbf{U}_1, \tau_1) \\ \vdots \\ \mathbf{F}(\mathbf{X}_N, \mathbf{U}_N, \tau_N) \end{bmatrix}$$

It is noted that the i-th *row* of either $\mathbf{X}^{\mathsf{GL}}$ or $\mathbf{U}^{\mathsf{GL}}$ corresponds to a value of the state or control, respectively, at the i-th Gauss-Lobatto point. The key features of the structure of Gauss-Lobatto points is that the state is approximated using a polynomial of degree $N-1$ and the derivative of the state (a polynomial of degree $N-2$) is collocated at the N Gauss-Lobatto points. As a result, the differentiation matrix $\mathbf{D}^{\mathsf{GL}}$ is *singular*.

Several methods using Gauss-Lobatto points have been developed. The first of these are the *Legendre pseudospectral method*[106] and the *Chebyshev pseudospectral method*.[107,109] A more general version of the Gauss-Lobatto formulation called the *Jacobi pseudospectral method*[110] while a Gauss-Lobatto quadrature method was developed in Ref. 111. More recently, a Legendre pseudospectral method was developed for feedback linearizable systems.[112]

Pseudospectral Method Using Legendre-Gauss Points The second major pseudospectral approach uses *Legendre-Gauss* (LG) points.[94,95,113–115] It is noted that the work of Refs. 94 and 95 is a *local* LG method while the work of Refs. 113–116 is a global LG method. In particular, as a global method, Refs. 113–116 describe the *Gauss pseudospectral method* (GPM). Denoting the LG points by $(\tau_1, \ldots, \tau_N)$ (where $\tau_0 = -1$, $\tau_1 > -1$ and $\tau_N < 1$, and $\tau_{N+1} = 1$), in the GPM the state is approximated as

$$\mathbf{x}(\tau) \approx \mathbf{X}(\tau) = \sum_{i=0}^{N} \mathbf{x}(\tau_i) L_i(\tau) \tag{70}$$

Furthermore, the cost is approximated using a Gauss quadrature,

$$J \approx \frac{t_f - t_0}{2} \sum_{i=1}^{N} w_i^{\mathsf{GPM}} \mathcal{L}[\mathbf{x}(\tau_i), \mathbf{u}(\tau_i), \tau_i] \tag{71}$$

where $\mathbf{w}_i^{\mathsf{GPM}}, \ (i = 1, \ldots, N)$ are the Gauss weights. The derivative approximation of Eq. (70) is then enforced at the N LG-points. The GPM differs fundamentally from the LPM or the CPM because the time derivative of the state is a polynomial of degree $N-1$ and is collocated at N points (as opposed to $N+1$ points as in the Gauss-Lobatto pseudospectral methods). In vector form, the GPM can be written as

$$\mathbf{D}^{\mathsf{GPM}} \mathbf{X} = \mathbf{D}_0^{\mathsf{GPM}} \mathbf{X}_0 + \mathbf{D}_1^{\mathsf{GPM}} \mathbf{X}^{\mathsf{GPM}} = \frac{t_f - t_0}{2} \mathbf{F}^{\mathsf{GPM}} \tag{72}$$

where

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_0 \\ \vdots \\ \mathbf{X}_N \end{bmatrix} \quad , \quad \mathbf{X}^{\mathsf{GPM}} = \begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_N \end{bmatrix} \quad , \quad \mathbf{F}^{\mathsf{GPM}} = \begin{bmatrix} \mathbf{F}(\mathbf{X}_1, \mathbf{U}_1, \tau_1) \\ \vdots \\ \mathbf{F}(\mathbf{X}_N, \mathbf{U}_N, \tau_N) \end{bmatrix}$$

and the rows of $\mathbf{X}$ or $\mathbf{X}^{\mathsf{GPM}}$ corresponds to the value of the state at the either initial point or one of the N interior LG points. The value of the state at $\tau_{N+1} \equiv 1$ is then obtained from a Gauss quadrature[45] as

$$\mathbf{X}_{N+1} = \mathbf{X}_0 + \frac{t_f - t_0}{2} \mathbf{w}^{\mathsf{GPM}} \mathbf{F}^{\mathsf{GPM}} \tag{73}$$

where $\mathbf{w}^{\mathsf{GPM}}$ is a row vector of Gauss weights. It turns out that $\mathbf{D}^{\mathsf{GPM}}$ is an $N \times (N+1)$ is full rank non-square matrix.

Pseudospectral Methods Using Legendre-Gauss-Radau Points Most recently, the use of *Legendre-Gauss-Radau* (LGR) points has received attention. First, local orthogonal collocation using LGR points was explored in Ref. 100. Next, two fundamentally different global LGR collocation techniques have been developed. First, in Ref. 117 an LGR method is developed for the special class of *infinite-horizon problems*. First, the time interval $t \in [0, \infty]$ is transformed to $\tau \in [-1, 1]$ via the transformation

$$t = \frac{1+\tau}{1-\tau} \tag{74}$$

Next, the state is approximated as

$$\mathbf{x}(\tau) \approx \mathbf{X}(\tau) = \sum_{i=1}^{N} \mathbf{x}(\tau_i) L_i(\tau) \tag{75}$$

where $(\tau_1, \ldots, \tau_N)$ are the LGR points. The infinite-horizon cost is then approximated using a Gauss-Radau quadrature,

$$J \approx \sum_{i=1}^{N} \frac{1}{(1-\tau_i)^2} w_i^{\mathsf{LGR}} \mathcal{L}[\mathbf{x}(\tau_i), \mathbf{u}(\tau_i), \tau_i] \tag{76}$$

where $w_i^{\mathsf{LGR}}, \ (i = 1, \ldots, N)$ are the Legendre-Gauss-Radau weights. The collocation equations then have the form

$$\mathbf{D}^{\mathsf{LGR}} \mathbf{X}^{\mathsf{LGR}} = \text{diag}\left(\frac{1}{(1-\tau_i)^2}\right) \mathbf{F}^{\mathsf{LGR}}, \quad (i = 1, \ldots, N) \tag{77}$$

where

$$\mathbf{X}^{\mathsf{LGR}} = \begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_N \end{bmatrix} \quad , \quad \mathbf{F}^{\mathsf{LGR}} = \begin{bmatrix} \mathbf{F}(\mathbf{X}_1, \mathbf{U}_1, \tau_1) \\ \vdots \\ \mathbf{F}(\mathbf{X}_N, \mathbf{U}_N, \tau_N) \end{bmatrix}$$

where it is noted that $\tau_N < +1$ is the last LGR point (which, by definition is *not* equal to +1). In addition, similar to the Gauss-Lobatto pseudospectral methods, the differentiation matrix $\mathbf{D}^{\mathsf{LGR}}$ in Ref. 117 is *singular*. In the formulation of Ref. 117, the singularity of Eq. (74) is avoided because no approximation to the dynamics is made at $\tau = +1$. Consequently, using the approach of Ref. 117 it is not possible to obtain the solution on the entire infinite horizon of the problem.

A second version of LGR collocation, called the *Radau pseudospectral method* (RPM), has been developed in Ref. 118. In particular, the method of Ref. 118 has the feature that it can be used to solve *either* finite-horizon *or* infinite-horizon problems. In the RPM, the state is approximated as

$$\mathbf{x}(\tau) \approx \mathbf{X}(\tau) = \sum_{i=1}^{N+1} \mathbf{x}(\tau_i) L_i(\tau) \tag{78}$$

where, again, $(\tau_1, \ldots, \tau_N)$ are the LGR points and $\tau_{N+1} = +1$ is a *noncollocated point*. The finite-horizon cost is then approximated as

$$J \approx \frac{t_f - t_0}{2} \sum_{i=1}^{N} w_i^{\mathsf{RPM}} \mathcal{L}[\mathbf{x}(\tau_i), \mathbf{u}(\tau_i), \tau_i] \tag{79}$$

where $w_i^{\mathsf{RPM}}, \ (i = 1, \ldots, N)$ are the Legendre-Gauss-Radau weights. The collocation equations then have the form

$$\mathbf{D}^{\mathsf{RPM}}\mathbf{X} = \tilde{\mathbf{D}}^{\mathsf{RPM}}\mathbf{X}_1 + \bar{\mathbf{D}}^{\mathsf{RPM}}\tilde{\mathbf{X}} = \frac{t_f - t_0}{2}\mathbf{F}^{\mathsf{LGR}} \tag{80}$$

where

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_{N+1} \end{bmatrix} \quad , \quad \tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X}_2 \\ \vdots \\ \mathbf{X}_{N+1} \end{bmatrix} \quad , \quad \mathbf{F}^{\mathsf{LGR}} = \begin{bmatrix} \mathbf{F}(\mathbf{X}_1, \mathbf{U}_1, \tau_1) \\ \vdots \\ \mathbf{F}(\mathbf{X}_N, \mathbf{U}_N, \tau_N) \end{bmatrix}$$

Similar to the GPM, in the RPM the state is solved for at the LGR points *plus* the terminal point. It is also noted that applying the transformation of Eq. (74) leads to a modified version of the RPM for infinite-horizon problems.[118] In the infinite-horizon RPM, the cost function is approximated as

$$J \approx \sum_{i=1}^{N} \frac{1}{(1 - \tau_i)^2} w_i^{\mathsf{RPM}} \mathcal{L}[\mathbf{x}(\tau_i), \mathbf{u}(\tau_i), \tau_i] \tag{81}$$

while the dynamics are approximated as

$$\mathbf{D}^{\mathsf{RPM}}\mathbf{X} = \tilde{\mathbf{D}}^{\mathsf{RPM}}\mathbf{X}_1 + \bar{\mathbf{D}}^{\mathsf{RPM}}\tilde{\mathbf{X}} = \mathbf{T}\mathbf{F}^{\mathsf{LGR}} \tag{82}$$

where

$$\mathbf{T} = \text{diag}\left(\frac{1}{(1 - \tau_i)^2}\right), \quad (i = 1, \ldots, N)$$

and

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \vdots \\ \mathbf{X}_{N+1} \end{bmatrix} \quad , \quad \tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X}_2 \\ \vdots \\ \mathbf{X}_{N+1} \end{bmatrix} \quad , \quad \mathbf{F}^{\mathsf{LGR}} = \begin{bmatrix} \mathbf{F}(\mathbf{X}_1, \mathbf{U}_1, \tau_1) \\ \vdots \\ \mathbf{F}(\mathbf{X}_N, \mathbf{U}_N, \tau_N) \end{bmatrix}$$

Because in the RPM the state is *not* collocated at $\tau = +1$, neither the cost of Eq. (81) nor the dynamics of Eq. (82) are evaluated at $\tau = +1$. The left-hand side of Eq. (82), however, contains an approximation of the state at $N + 1$ (where the index $N + 1$ corresponds to $\tau = +1 \Leftrightarrow t = \infty$). As a result, the state in the infinite-horizon RPM is solved for on the *entire* horizon. The reader is referred to Ref. 118 for more details on the Radau pseudospectral method.

Relationship Between Different Pseudospectral Methods From the aforementioned discussion, we can see how the different pseudospectral methods compare. In the LGL or infinite-horizon LGR approach, the state is approximated using a global polynomial of degree M and the dynamics are collocated at $M + 1$ points. This approach leads to a differentiation operator that is square and singular, because the derivative is collocated at all of the discretization (i.e., approximation) points. On the other and, in the GPM or the RPM, the state is approximated using a polynomial of degree M and the dynamics are collocated at M points. In this second approach, the derivative is *not* collocated at all of the approximation points, leading to *full-rank* and *non-square* differentiation matrices. Interestingly, the LGL and GPM/RPM approaches are consistent with their respective quadrature rules for integration (LGL⇔Gauss-Lobatto quadrature; GPM⇔Gauss quadrature; RPM⇔Radau quadrature). Furthermore, in accordance with the quadrature rules and because the derivative is not collocated at all of the approximation points, the GPM and the RPM have the feature that the dynamics can be written equivalently in either differential or implicit integral form. The implicit integral form for the GPM or RPM is given generically as

$$\mathbf{X} = \mathbf{X}_0 + \frac{t_f - t_0}{2}\mathbf{A}\mathbf{F} \tag{83}$$

where $\mathbf{A}$ is an *integration matrix*.[113,118,119] As described in Ref. 113, the differential form of the GPM or RPM has the advantage *over the integral form* that the collocation equations have sparse constraint Jacobians (whereas in the integral form the GPM/RPM constraint Jacobian is *highly dense*). We note, however, that in the differential form all three methods (LPM, GPM, and RPM) have equally sparse constraint Jacobians.

Comparison of GPM/RPM with Other LG/LGR Collocation Schemes As a final point, it is important to clarify the distinction between the GPM/RPM and the LG/LGR schemes considered in Ref. 120. In particular, the LG/LGR schemes of Ref. 120 are modifications of the Lobatto pseudospectral method applied to the LG and LGR points. In other words, in the LG/LGR schemes of Ref. 120, the state is approximated using Eq. (67), where the indices "1" and "N" are the first and last collocation point, respectively (and not the initial and terminal point). Thus, the LG scheme of Ref. 120, the dynamics are collocated and discretized at the N LG points. Similarly, in the LGR scheme of Ref. 120, the dynamics are collocated and discretized at the N LGR points. It is seen that the GPM and RPM are fundamentally different from the LG/LGR schemes of Ref. 120 because the GPM and RPM discretize the state at the collocation points plus any uncollocated endpoints. In addition, similar to the LPM, the LG/LGR schemes of Ref. 120 employ singular differentiation matrices whereas the GPM and RPM employ non-square full-rank matrices that lead to the ability to write the GPM and RPM equivalently in either differential or integral form.

Solving Nonsmooth Problems Using Pseudospectral Methods One drawback to using a pseudospectral method is due to *nonsmoothness*. In particular, it is well-known that spectral accuracy only holds when the functions under consideration are smooth.[102,121] In the case of nonsmooth optimal control problems, several variations to the basic pseudospectral (orthogonal collocation) methods have been considered. In Ref. 94, LG points and finite elements were used together with the concept of an *orthogonal collocation knot* to divide a nonsmooth problem into *finite elements* so that discontinuities could be captured. In Ref. 122, an approach using LGL points is considered where the nonsmooth problem is solved in the same manner as that used in Ref. 106 and applies a filter to recover the nonsmooth solution to within spectral accuracy. In Ref. 123 an approach called *spectral patching* (similar to the approach used Ref. 94) was developed to capture discontinuities in control without changing the state variable representation at the local of the discontinuity. Ref. 124 considers the problem of nonlinear transformations between non-sequential trajectory segments and allows for possible discontinuities at the segment interfaces. Finally, Ref. 125 describes an approach similar to that of Ref. 123, but allowing for nonlinear transformations of the state between segments.

Costate Estimation Using Pseudospectral Methods A key property of the structure of a pseudospectral method that was observed elegantly in Ref. 126 is the similarity between the continuous-time first-order optimality conditions [see Eqs. (55)–(60)] and the discrete KKT conditions of the NLP [see Eqs. (39)–(43)].[126] One of the first works on costate estimation using pseudospectral methods is that given in Ref. 126, where it was found that the LGL collocation method[106] yielded a "near equivalence" between the indirect and direct forms. This "near equivalence" exhibited a mismatch at only the boundary points due to a mixture of the costate dynamics and the transversality conditions. Numerically, this mismatch led to the costate estimates being "noisy." As a result, a great deal of research was subsequently done to see if improved costates could be obtained using LGL points. The first enhancement to the result of Ref. 126 was to introduce *closure conditions*. Essentially, in the closure conditions the transversality conditions are manually from the discretized costate dynamic equations at the boundary points.[127] The introduction of the closure conditions leads to a mixed primal-dual problem where the transversality conditions are augmented to the KKT conditions of the NLP. Realizing that solving such a problem can be cumbersome due to the need to derive the KKT conditions, the closure condition theory was followed by a theory of *relaxation* where the defect constraints on the differential equations were solved in a least-squares fashion.[128]

In parallel with the work on LGL costate estimation, the work on the Gauss pseudospectral method (GPM) done in Refs. 113–115 showed that Legendre-Gauss points could be used to develop an *exact* equivalence between the direct and indirect forms.[113–115] In fact, Ref. 113 developed both the integral and derivative forms of the GPM and showed that the same costate mapping is obtained regardless of which form of the GPM is used. The work of Refs. 113–115 was fundamentally different from the work of Ref. 126 and 127 because the GPM costate was related to the KKT multipliers of the NLP in a simple algebraic manner.

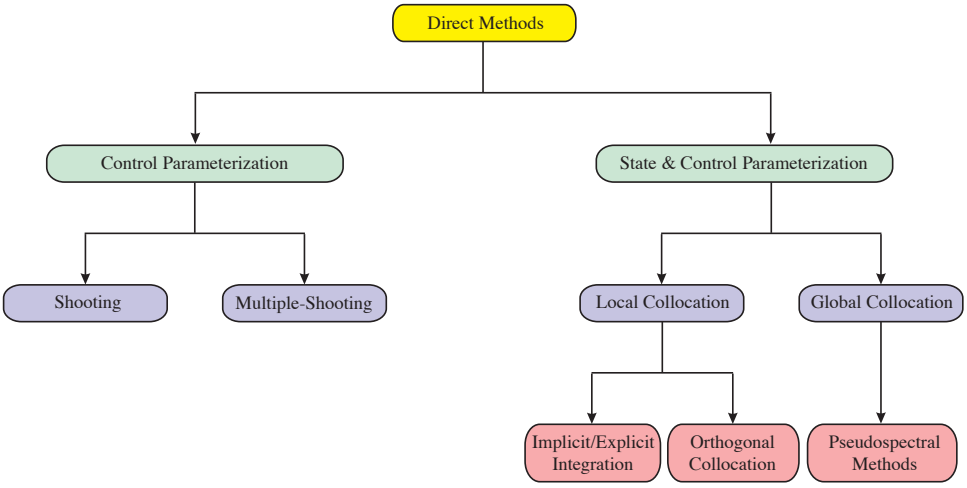

Figure 6: Different Types of Direct Methods.

Most recently, the aforementioned LGR pseudospectral methods[117,118] have led to costate estimates. The costate mapping of Ref. 117 shows an equivalence between the indirect and direct forms at the LGR points for an infinite-horizon problem. Similar to the result using the GPM, the costate mapping of Ref. 118 is also obtained via a simple algebraic transformation of the KKT multipliers of the NLP. Interestingly, while the RPM does not have the exact equivalence of the GPM, it does not exhibit the noise that was originally found in the LGL costate estimation of Ref. 126.

COMPUTATIONAL ISSUES

Four important computational issues that arise in the numerical solution of optimal control problems are (a) consistent approximations for the solution of differential equations; (b) the scaling of optimal control problem; (c) exploitation of sparsity in the NLP; and (d) computation of derivatives of the objective and constraint functions. The manner in which the differential equations are discretized is of great important because an inconsistent approximation to the differential equations can lead to either nonconvergence or convergence of the optimization problem to the wrong solution. Scaling and exploitation of sparsity in the NLP are issues that greatly affect both computational efficiency and convergence of the NLP. Finally, the manner in which derivatives are computed is of great importance because accurate derivatives can greatly improve both computational efficiency and reliability. Each of these issues is now discussed.

Consistent Approximations in Discretization of Differential Equations

It is well known that any time-marching method for solving an initial-value problem (IVP) must satisfy the *Lax-Richtmyer equivalence theorem*.[129,130] The Lax-Richtmyer equivalence theorem states that any consistent finite difference method for a linear IVPs is convergent if and only if it is stable. While many consistent and stable schemes have been developed for IVPs, it is not always the case that such methods converge when used to solve optimal control problems. Schwartz and Polak provide an excellent discussion on consistent approximations for solving optimal control problems using Runge-Kutta methods[84,85] while Ref. 131 provides a compendium of mathematical theory on consistent approximations in optimal control. In addition, it was shown in Ref. 86 that the requirements for a stable and consistent Runge-Kutta method are different when solving an optimal control problem as opposed to solving a differential equation. Furthermore, Ref. 86 showed that, while certain implicit Runge-Kutta methods are convergent for differential equations, these methods may be divergent for optimal control problems. While it is beyond the scope of this paper to provide an in-depth analysis of the validity of different discretization method, this brief discussion suggests that a method must be chosen with care to ensure stability and convergence.

Scaling

It is known that a properly scaled NLP can be solved much more easily as compared to one that is poorly scaled. Scaling issues arises most commonly when the relative sizes of variables or constraints in a problem are vastly different and when the variables or constraints are not close to unit size. The best solution to dealing with scaling is to scale a problem manually. For problems that arises in flight dynamics where the governing equations are based on Newton's and Euler's laws,[132] a common way to scale a problem is via a canonical transformation of the units. For example, if the original problem is posed in either English or SI units, one chooses the four fundamental scale quotients for length, speed, time, and mass. Denoting these four scale quotients as L, V, T, and M, a canonical transformation of units would be obtained by setting

$$\frac{LT}{V} = 1 \tag{84}$$

Then, the value of all other scale quotients would be obtained from these four quantities. For example, the acceleration and force scale quotients, denoted A and F, respectively, would be given as

$$A = \frac{V}{T} \tag{85}$$

$$F = MA \tag{86}$$

While procedures such as canonical scaling are preferred, such approaches do not always work. In such cases it is necessary to devise automatic methods for scaling optimal control problems. While these methods tend to be heuristic, the following method, described in Ref. 36, has been found to work well in practice. In the method of Ref. 36, scale quotients for the state and control are chosen so that the scaled value of these quantities lies on the interval $[-1, 1]$. The differential equations are then scaled using the same scale quotients used to scale the state. Finally, the path and event constraints are scaled by sampling the Jacobians of these functions at several randomly chosen points. The scale quotients for these constraints are then taken to be the average of the row norms of these Jacobians.

Sparsity

Sparsity is a key feature that is extremely important to exploit, particularly in the case of direct collocation methods. In particular, it is well known that the vast majority of the derivatives of the defect and path constraints in a direct collocation method are *zero*. Because the constraint Jacobians in a direct collocation method are very large (thousands of to tens of thousands of variables and constraints), it is absolutely essential that only the nonzero derivatives be computed and stored. In fact, NLP solvers such as *SNOPT*,[25,46] *SPRNLP*,[51] *KNITRO*,[26] and *IPOPT*,[27,54] take advantage of this sparsity both from the standpoint of computation and memory. Furthermore, *SPRNLP*,[51] *KNITRO*,[26] and *IPOPT*[27,54] have the added advantage that they utilize *second derivatives* (using the Hessian of the Lagrangian) which turns these NLP solvers from quasi-Newton methods (where convergence near the optimal solution is *superlinear*) to *Newton methods* (where convergence near the optimal solution is *quadratic*). An excellent discussion about sparsity and its importance in direct collocation methods can be found Ref. 36.

Differentiation

As is seen from the earlier discussion, direct methods for optimal control use gradient-based techniques for solving the NLP. Gradient methods for solving NLPs require the computation of the derivatives of the objective function and constraints. Perhaps the most obvious way to computing these derivatives is by analytic differentiation. In fact, some optimal control software has been designed such that the derivatives of many commonly used models have been coded and can be reused when necessary (e.g., *GMAT*[133]). While such an approach is appealing because analytic derivatives are exact and generally result in faster optimization, for much software development it is impractical to compute derivatives analytically. As a result, alternative means must be employed to obtain the necessary gradients.

The most basic way to estimate derivatives for use in an NLP is by *finite difference approximation*. The two most common methods for the finite-difference approximation of a derivative are *forward differencing* and *central differencing*. A forward and central difference approximations to the derivative of a function $f(x)$ are given, respectively, as

$$\frac{df}{dx} \approx \frac{f(x+h)-f(x)}{h} \tag{87}$$

$$\frac{df}{dx} \approx \frac{f(x+h)-f(x-h)}{2h} \tag{88}$$

where h is a perturbation that is sufficiently small to provide a good approximation to the derivative, but not so small that it creates roundoff error in the computation of $f(x+h)-f(x)$. While finite difference approximations work for solving some NLPs, for many NLPs using these approximations will result either in slow convergence to a solution or non-convergence. Because finite difference approximation has limits, more efficient and more accurate methods for estimating derivatives of functions is an ongoing topic of research and development.[‡]

One approach to computing approximations to derivatives that is both accurate and relatively efficient is *complex-step differentiation*.[134,135] As its name implies, complex-step differentiation requires that the programming language allow for complex arithmetic. The basic idea behind complex-step is as follows. Consider an analytic complex function $f = u + iv$ of the complex variable $x + iy$. It is known that f satisfies the *Cauchy-Riemann* equations,[136] i.e.,

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}$$

$$\frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}$$

Then

$$\frac{\partial u}{\partial x} = \lim_{h\to 0} \frac{v(x+i(y+h))-v(x+iy)}{h} \tag{89}$$

where h is real. Because we are interested in computing the derivative of real-valued functions, we have

$$\begin{aligned} y &\to 0 \\ u(x) &\to f(x) \\ v(x) &\to 0 \end{aligned}$$

Eq. (89) can then be written as

$$\frac{df}{dx} = \lim_{h\to 0} \frac{\mathrm{Im}[f(x+ih)]}{h} \tag{90}$$

which can be approximated as

$$\frac{df}{dx} \approx \frac{\mathrm{Im}[f(x+ih)]}{h} \tag{91}$$

Eq. (91) is called the *complex-step derivative approximation*[134,135] and is not subject to the subtractive cancellation errors associated with real finite difference approximations. It turns out that the complex-step derivative approximation is $\mathcal{O}(h^2)$ accurate (as opposed to $\mathcal{O}(h)$ for real finite-differencing). In addition, because Eq. (91) does not suffer from subtractive cancellation, extremely small values of h can be used, thus providing an extremely accurate derivative.[135] Finally, it is noted that the complex-step derivative approximation is straightforward to implement in MATLAB because MATLAB uses complex numbers by default.*

While complex-step differentiation is accurate and reasonably efficient, it has a few limitations. First, complex-step differentiation requires double the memory because of the need to perform complex arithmetic.

[‡]Quoting from the SNOPT User's Manual,[46] "For maximum reliability, compute all gradients."

*On a cautionary note, the functions abs, min, and max must be redefined to work properly with complex-step differentiation. See Ref. 135 for more details.

Second, not all programming languages are easily set up to handle complex functions (e.g., FORTRAN 77, which is still widely used today). Another powerful approach for the differentiation of functions is *automatic differentiation*[137–146] has been developed. In automatic differentiation, the derivatives of the functions are computed to the same precision as they would be if they had been computed analytically. Two commonly used approaches to automatic differentiation are *source transformation*[137,138] and *function overloading*[139–146] In a source transformation approach, the code used to compute the functions is transformed into a new code that is used to compute the derivatives. The most well known source transformation tool that has been used widely is the program *ADIFOR*.[137,138] An alternative to source transformation that takes advantage of modern computer programming languages such as C++ and MATLAB is *function overloading*. In function overloading, a class is defined and functions are defined that operate on objects of the class. Because the language permits function overloading, the built-in functions can be defined to output both the function value and its derivative. In this manner, the chain rule for differentiation can be implemented in the same number of operations as would be required to compute the derivatives by hand.

SOFTWARE FOR SOLVING TRAJECTORY OPTIMIZATION PROBLEMS

A wide variety of software tools have been developed for solving trajectory optimization problems. Most of these software programs use direct methods. One well known software program employing indirect methods is *BNDSCO*[147] which employs a multiple-shooting method. Perhaps the oldest software tool that employs direct methods is the *Program to Simulate and Optimize Trajectories*[148] (*POST*). *POST* was originally developed to solve problems in launch vehicle trajectory optimization and it still in use today for such applications.

The late 1980s saw a transformation in the available tools for solving optimal control problems. This transformation was coincident with the observation of the power of direct collocation methods. The first well-known direct collocation software was *Optimal Trajectories by Implicit Simulation*[149] (*OTIS*). *OTIS* is a FORTRAN software that has general-purpose capabilities for problems in aeronautics and astronautics. OTIS has been used widely in the aerospace and defense industries and its theoretical foundations are found in Ref. 150. Following shortly after the development of *OTIS* is the program *Sparse Optimal Control Software*[151] ($\mathbb{SOCS}$). $\mathbb{SOCS}$ is an incredibly powerful FORTRAN software that is capable of solving the most challenging optimal control problems one can realistically pose. Some of the applications solved using $\mathbb{SOCS}$ are found in Refs. 152–156. Finally, three other direct collocation FORTRAN programs are *MISER*,[157] *Direct Collocation*[158] (*DIRCOL*), *Graphical Environment for Simulation and Optimization*[159] (*GESOP*), and *Nonlinear Trajectory Generation*[160] (*NTG*). Like *OTIS* and $\mathbb{SOCS}$, *DIRCOL* and *GESOP* use local direct collocation techniques while *NTG* is designed for rapid trajectory generation of differentially flat systems.

In recent years, interest in the particular application of optimal control to space flight has led to the development of several useful programs. One such program is *Mission Design and Analysis Software*[161] (*MIDAS*) which is designed to solve complex ballistic heliocentric transfer trajectories for interplanetary space flight missions. Another tool that has been recently developed is the NASA *Generalized Mission Analysis Tool*[133] (*GMAT*). Another tool that has been widely used in the last several years is *COPERNICUS*.[162,163] Both *GMAT* and *COPERNICUS* are designed to solve optimal control problems where the maneuvers can be treated as either impulsive or finite-thrust burns.

While earlier software programs used compiled languages such as FORTRAN, in recent years, MATLAB® has become increasingly popular for solving optimization problems. The increased appeal for MATLAB emanates from the fact that MATLAB is an extremely easy environment in which to program along with the fact that many of today's most powerful NLP solvers are now available for use in MATLAB® (e.g., standalone MATLAB mex versions are now available for the NLP solvers *SNOPT*[24,25] and *KNITRO*[26]). In addition, the TOMLAB[164–169] package has facilitated additional solvers for use in MATLAB. In addition, because of major computational improvements, the computational efficiency between MATLAB and compiled languages is growing ever closer. Examples of MATLAB-based optimal control software programs include RIOTS_95,[170] *DIDO*,[171] *DIRECT*,[172] *PROPT*,[173] *OPTCONTROLCENTRE*,[174] and $\mathcal{GPOPS}$.[116]

It is important to note that all of the optimal control software programs described above incorporate gradient methods for solving the NLP. In a less formal manner, heuristic methods have also been used to solve

optimal control problems. For example, interplanetary trajectory optimization problems using a genetic algorithm have been considered in Ref. 175,176 while low-thrust orbit transfers using a genetic algorithm have been studied in Ref. 177 and 178. In addition, a calculus of variations technique has been used together with a genetic algorithm to optimize low-thrust Mars-to-Earth trajectories for the Mars Sample Return Mission.[179] Thus, while gradient methods are somewhat the de facto standard for optimal control, the aforementioned research demonstrates that genetic algorithms may be well-suited for some applications.

CHOOSING A METHOD

Choosing a method for solving an optimal control problem is based largely on the type of problem to be solved and the amount of time that can be invested in coding. An indirect shooting method has the advantage that it is simple to understand and produces highly accurate solutions when it converges. Unfortunately, a shooting method suffers from numerical problems for a wide range of problems that are summarized elegantly in the following quote from Ref. 12:

> The main difficulty with these methods is getting started; i.e., finding a first estimate of the unspecified conditions at one end that produces a solution reasonably close to the specified conditions at the other end. The reason for this peculiar difficulty is the extremal solutions are often very sensitive to small changes in the unspecified boundary conditions.... Since the system equations and the Euler-Lagrange equations are coupled together, it is not unusual for the numerical integration, with poorly guessed initial conditions, to produce "wild" trajectories in the state space. These trajectories may be so wild that values of $\mathbf{x}(t)$ and/or $\boldsymbol{\lambda}(t)$ exceed the numerical range of the computer!

While the above quotation addresses the fact that the indirect shooting method is extremely sensitive to the unknown boundary conditions, it does not address the other, perhaps even more important, shortcoming of indirect shooting: an indirect shooting requires the derivation of the first-order optimality conditions of the optimal control problem [see Eqs. (55)–(60]! While for simple problems it may be possible to derive the first-order optimality conditions, deriving such conditions for complex optimal control problems is tedious, error-prone, and sometimes impossible (e.g., problem with table lookups). Furthermore, the need to derive the optimality conditions makes implementing indirect shooting difficult in a general-purpose software program. For example, if it was required to derive first-order optimality conditions, a program such as *POST* would become nearly impossible to use because every new problem would require the derivation of these conditions! A multiple-shooting method overcomes some of the numerical difficulties of standard shooting, but does not avoid the issue of having to derive the optimality conditions.

The accuracy and robustness of a direct method is highly dependent upon the form of direct method used. Direct shooting methods are very good for problems where the control can be parameterized in a simple manner (e.g., piecewise linear functions of time) and the problem can be characterized accurately using a small number optimization parameters. Software programs such as *POST* perform well on launch vehicle ascent trajectories because these problems can be approximated accurately using simple control parameterizations. As the complexity of the problem increases, it becomes more and more apparent that the workhorse for solving optimal control problems is the direct collocation method. The two main reasons that direct collocation methods work so well is because highly complex problems can be formulated and solved with today's NLP solvers. The reason that the NLP solvers can handle such complex problems is because they are designed to converge with poor initial guesses (e.g., straight line guesses in the state and control) and are extremely computationally efficient because they exploit the sparsity of the derivatives in the constraints and objective function. In fact, one of the virtues of the software $\mathbb{SOCS}$ is that *SPRNLP* (the NLP solver used in $\mathbb{SOCS}$) exploits sparsity in the *second derivatives*. As a result, *SPRNLP* will converge at a quadratic rate when it is in the neighborhood of a minimizing solution to the NLP! Second derivative information, if available, is also used by the NLP solvers *BARNLP*, *KNITRO*,[26] and *IPOPT*.[27,54]

In many cases, the solution of an optimal control problem is a means to an end, i.e., the user does not want to know all of the details about a method, but simply wants to use a software program to provide results so that

a particular problem of interest can be solved. If one does not wish to become an expert in the technologies associated with optimal control, it is advisable to obtain a canned software package that allows a user to input the problem in an intuitive manner. Then the software can simply be run on the problem of interest. It is always important to understand, however, that canned software can have its issues when things go wrong because the user may often not understand why.

CONCLUSIONS

A survey of numerical methods for solving optimal control problems has been given. The problem of solving optimal control problems has been decomposed into the three key components of solving differential equations and integrating functions, solving nonlinear optimization problems, and solving systems of nonlinear algebraic equations. Using these components, the two classes of indirect and direct methods for solving optimal control problems have been described. Subsequently, important computational issues have been discussed and several different software tools for solving optimal control problems have been described. Finally, a brief discussion has been given on how to choose a method.

DISCLAIMER

The views and opinions expressed in this article are those of the author and do not reflect those of the author's employer or any agencies that sponsored this research. In addition, it is noted that the citations in this article are restricted to documents that are available in the public domain (i.e., documents available upon request to *any* person or organization, without any restrictions).

ACKNOWLEDGMENTS

Support for this research from the U.S. Army Research Office under Grant 55173-CI and The Office of Naval Research under Grant N00014-08-1-1173 is gratefully acknowledged.

REFERENCES

[1] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.

[2] R. Bellman and S. Dreyfus, "Functional Approximations and Dynamic Programming," *Mathematical Tables and Other Aids to Computation*, Vol. 13, October 1959, pp. 247–251.

[3] R. Bellman, "Dynamic Programming Treatment of the Travelling Salesman Problem," *Journal of Association of Computing Machinery*, Vol. 9, No. 1, 1962, pp. 61–63.

[4] R. Bellman, R. Kalaba, and B. Kotkin, "Polynomial Approximation – A New Computational Technique in Dynamic Programming: Allocation Processes," *Mathematics of Computation*, Vol. 17, April 1963, pp. 155–161.

[5] R. Bellman, "Dynamic Programming," *Science 1*, Vol. 153, July 1966, pp. 34–37.

[6] R. E. Bellman and S. E. Dreyfus, *Applied Dynamic Programming*. Princeton University Press, 1971.

[7] G. A. Bliss, *Lectures on the Calculus of Variations*. Chicago, IL: University of Chicago Press, 1946.

[8] R. Weinstock, *Calculus of Variations*. Mineola, New York: Dover Publications, 1974.

[9] F. B. Hildebrand, *Methods of Applied Mathematics*. Mineola, New York: Dover Publications, 1992.

[10] G. Leitman, *The Calculus of Variations and Optimal Control*. New York: Springer, 1981.

[11] M. A. Athans and P. L. Falb, *Optimal Control: An Introduction to the Theory and Its Applications*. Mineola, New York: Dover Publications, 2006.

[12] A. E. Bryson and Y.-C. Ho, *Applied Optimal Control*. New York: Hemisphere Publishing, 1975.

[13] W. H. Fleming and R. W. Rishel, *Deterministic and Stochastic Optimal Control*. Heidelberg, Germany: Springer, 1982.

[14] R. F. Stengel, *Optimal Control and Estimation*. Mineola, New York: Dover Publications, 1994.

[15] R. Vintner, *Optimal Control (Systems & Control: Foundations and Applications)*. Boston: Birkhäuser, 2000.

[16] D. G. Hull, *Optimal Control Theory for Applications*. New York: Springer-Verlag, 2003.

[17] F. L. Lewis and V. L. Syrmos, *Optimal Control*. New York: John Wiley and Sons, 2 ed., 1995.

[18] D. Kraft, *On Converting Optimal Control Problems into Nonlinear Programming Codes*, Vol. F15 of *NATO ASI Series,* in *Computational Mathematical Programming*, pp. 261–280. Berlin: Springer-Verlag, 1985.

[19] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*. Wiley-Interscience, 3 ed., 2006.

[20] D. Bertsekas, *Nonlinear Programming*. Belmont, Massachusetts: Athena Scientific Publishers, 2004.

[21] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, United Kingdom: Cambridge University Press, 2004.

[22] J. Nocedal and S. Wright, *Numerical Optimization*. New York: Springer-Verlag, 2 ed., 2006.

[23] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, *User's Guide for NPSOL (Version 4.0): A FORTRAN Package for Nonlinear Programming*. Department of Operations Research, Stanford University, January 1986.

[24] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Review*, Vol. 47, January 2002, pp. 99–131.

[25] P. E. Gill, W. Murray, and M. A. Saunders, *User's Guide for SNOPT Version 7: Software for Large Scale Nonlinear Programming*, February 2006.

[26] R. H. Byrd, J. Nocedal, and R. A. Waltz, "KNITRO: An Integrated Package for Nonlinear Optimization," *Large Scale Nonlinear Optimization*, Springer Verlag, 2006, pp. 35–59.

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

[28] J. T. Betts, *Practical Methods for Optimal Control Using Nonlinear Programming*. Philadelphia: SIAM Press, 2001.

[29] U. M. Ascher, R. M. Mattheij, and R. D. Russell, *Numerical Solution of Boundary-Value Problems in Ordinary Differential Equations*. Philadelphia: SIAM Press, 1996.

[30] E. Polak, "An Historical Survey of Computational Methods in Optimal Control," *SIAM Review*, Vol. 15, April 1973, pp. 553–584.

[31] A. Miele, "Recent Advances in Gradient Algorithms for Optimal Control Problems," *Journal of Optimization Theory and Applications*, Vol. 17, December 1975, pp. 361–430.

[32] A. E. Bryson, "Optimal Control – 1950 to 1985," *IEEE Control Systems Magazine*, Vol. 16, No. 3, 1996, pp. 26–33.

[33] H. H. Goldstine, *A History of the Calculus of Variations from the 17th to the 19th Centuries*. Heidelberg, Germany: Springer-Verlag, 1980.

[34] O. v. Stryk and R. Bulirsch, "Direct and Indirect Methods for Trajectory Optimization," *Annals of Operations Research*, Vol. 37, 1992, pp. 357–373.

[35] D. G. Hull, "Conversion of Optimal Control Problems into Parameter Optimization Problems," *Journal of Guidance, Control, and Dynamics*, Vol. 20, No. 1, 1997, pp. 57–60.

[36] J. T. Betts, "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidnance, Control, and Dynamics*, Vol. 21, March–April 1998, pp. 193–207.

[37] W. C. Gear, *Numerical Initial-Value Problems in Ordinary Differential Equations*. Englewood Cliffs, New Jersey: Prentice-Hall, 1971.

[38] G. Dahlquist and A. Björck, *Numerical Methods*. Mineola, New York: Dover Publications, 2003.

[39] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*. Springer-Verlag, 2002.

[40] J. C. Butcher, "Implicit Runge-Kutta Processes," *Mathematics of Computation*, Vol. 18, No. 85, 1964, pp. 50–64.

[41] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*. New York: John Wiley and Sons, 2008.

[42] E. Hairer, S. P. Norsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*. New York: Springer-Verlag, 1993.

[43] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff Differential-Algebraic Problems*. New York: Springer-Verlag, 1996.

[44] C. d. Boor and B. Schwartz, "Collocation at Gaussian Points," *SIAM Journal on Numerical Analysis*, Vol. 10, September 1973, pp. 582–606.

[45] P. J. Davis and P. Rabinowitz, *Methods of Numerical Integration*. New York: Academic Press, 1984.

[46] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. London: Academic Press, 1981.

[47] C. G. Broyden, "The Convergence of a Class of Double-Rank Minimization Algorithms," *Journal of the Institute of Mathematics and Its Applications*, Vol. 6, No. 1, 1970, pp. 76–90.

[48] R. Fletcher, "A New Approach to Variable Metric Algorithms," *Computer Journal*, Vol. 13, No. 3, 1970, pp. 317–322.

[49] D. Goldfarb, "A Family of Variable Metric Updates Derived by Variational Means," *Mathematics of Computation*, Vol. 24, No. 3, 1970, pp. 23–26.

[50] D. F. Shanno, "Conditioning of Quasi-Newton Methods for Function Minimization," *Mathematics of Computation*, Vol. 24, 1970, pp. 647–656.

[51] J. T. Betts and W. P. Huffman, "A Sparse Nonlinear Optimization Algorithm," *Journal of Optimization Theory and Applications*, Vol. 82, September 1994, pp. 519–541.

[52] R. J. Vanderbei and D. F. Shanno, "An Interior-Point Algorithm for Nonconvex Nonlinear Programming," *Computational Optimization and Applications*, Vol. 13, 1999, pp. 231–252.

[53] R. H. Byrd, M. E. Hribar, and J. Nocedal, "An Interior Point Algorithm for Large-Scale Nonlinear Programming," *SIAM Journal on Control and Optimization*, Vol. 9, No. 4, 1999, pp. 877–900.

[54] A. Wächter, *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering*. PhD thesis, Department of Chemical Engineering, Carnegie-Mellon University, Pittsburgh, PA, 2002.

[55] A. Wächter and L. T. Biegler, "On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming," *Mathematical Programming*, Vol. 106, No. 1, 2006, pp. 25–57.

[56] L. T. Biegler and V. M. Zavala, "Large-Scale Nonlinear Programming Using IPOPT: An Integrating Framework for Enterprise-Wide Optimization," *Computers and Chemical Engineering*, Vol. 33, March 2008, pp. 575–582.

[57] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press/Brooks-Cole Publishing, 2002.

[58] R. Horst, P. M. Pardalos, and N. V. Thoai, *Introduction to Global Optimization*. Norwell, MA: Kluwer Academic Publishers, 2000.

[59] K. A. DeJong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, Ann Arbor, Michigan, 1975.

[60] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, Massachusetts: Addison-Wesley, 1989.

[61] L. Davis, *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold, 1991.

[62] Z. Michalewicz, *Genetic Algorihms + Data Structures = Evolutionary Programs*. Berlin: Springer-Verlag, 1992.

[63] M. Srinivas and L. M. Patnaik, "Genetic Algorithms: A Survey," *Computer*, Vol. 27, June 1994, pp. 17–26.

[64] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 220, No. 4598, 1983, pp. 671–681.

[65] V. Cerny, "A Thermodynamic Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm," *Journal of Optimization Theory and Applications*, Vol. 45, 1985, pp. 41–51.

[66] R. C. Eberhardt and J. A. Kennedy, "A New Optimizer Using Particle Swarm Theory," *Proceedings of the Sixth International Symposium on Micromachine and Human Science*, Nagoya, Japan, 1995, pp. 39–43.

[67] R. C. Eberhardt and J. A. Kennedy, "Particle Swarm Optimization," *Proceedings of the IEEE International Conference on Neural Networks*, Piscataway, NJ, 1995, pp. 1942–1948.

[68] J. Kennedy and R. C. Eberhardt, *Swarm Intelligence*. Morgan Kaufmann, 2001.

[69] W. H. Press, B. P. Flannery, S. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 1990.

[70] H. Kwakernaak and R. Sivan, *Linear Optimal Control Systems*. New York: John Wiley and Sons, 1972.

[71] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*. Englewood Cliffs, NJ: Prentice-Hall, 1990.

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

[73] L. S. Pontryagin, *Mathematical Theory of Optimal Processes*. New York: John Wiley & Sons, 1962.

[74] N.-X. Vinh, *Optimal Trajectories in Atmospheric Flight*. New York: Elsevier Science, 1981.

[75] H. B. Keller, *Numerical Solution of Two Point Boundary Value Problems*. SIAM, 1976.

[76] A. V. Rao, *Extension of the Computational Singular Perturbation Method to Optimal Control*. PhD thesis, Princeton University, 1996.

[77] A. V. Rao and K. D. Mease, "Dichotomic Basis Approach to solving Hyper-Sensitive Optimal Control Problems," *Automatica*, Vol. 35, April 1999, pp. 633–642.

[78] A. V. Rao and K. D. Mease, "Eigenvector Approximate Dichotomic Basis Method for solving Hyper-Sensitive optimal Control Problems," *Optimal Control Applications and Methods*, Vol. 21, January–February 2000, pp. 1–19.

[79] A. V. Rao, "Application of a Dichotomic Basis Method to Performance Optimization of Supersonic Aircraft," *Journal of Guidance, Control, and Dynamics*, Vol. 23, May–June 2000, pp. 570–573.

[80] A. V. Rao, "Riccati Dichotomic Basis Method for solving Hyper-Sensitive optimal Control Problems," *Journal of Guidance, Control, and Dynamics*, Vol. 26, January–February 2003, pp. 185–189.

[81] W. Grimm and A. Markl, "Adjoint Estimation from a Multiple-Shooting Method," *Journal of Optimization Theory and Applications*, Vol. 92, February 1997, pp. 263–283.

[82] C. J. Goh and K. L. Teo, "Control Parameterization: A Unified Approach to Optimal Control Problems with General Constraints," *Automatica*, Vol. 24, January 1988, pp. 3–18.

[83] J. S. Logsdon and L. T. Biegler, "Accurate Solution of Differential-Algebraic Optimization Problems," *Industrial and Engineering Chemistry Research*, Vol. 28, 1989, pp. 1628–1639.

[84] A. Schwartz, *Theory and Implementation of Numerical Methods Based on Runge-Kutta Integration for Solving Optimal Control Problems*. PhD thesis, Department of Electrical Engineering, University of California, Berkeley, 1996.

[85] A. Schwartz and E. Polak, "Consistent Approximations for Optimal Control Problems Based on Runge-Kutta Integration," *SIAM Journal on Control and Optimization*, Vol. 34, July 1996, pp. 1235–1269.

[86] W. W. Hager, "Runge-Kutta Methods in Optimal Control and the Transformed Adjoint System," *Numerische Mathematik*, Vol. 87, 2000, pp. 247–282.

[87] A. L. Dontchev, W. W. Hager, and K. Malanowski, "Error Bounds for the Euler Approximation and Control Constrained Optimal Control Problem," *Numerical Functional Analysis and Applications*, Vol. 21, 2000, pp. 653–682.

[88] A. L. Dontchev, W. W. Hager, and V. M. Veliov, "Second-Order Runge-Kutta Approximations In Constrained Optimal Control," *SIAM Journal on Numerical Analysis*, Vol. 38, 2000, pp. 202–226.

[89] A. L. Dontchev, W. W. Hager, and V. M. Veliov, "Uniform Convergence and Mesh independence of Newton's method for Discretized Variational Problems," *SIAM Journal on Control and Optimization*, Vol. 39, 2000, pp. 961–980.

[90] A. L. Dontchev and W. W. Hager, "The Euler Approximation in State Constrained Optimal Control," *Mathematics of Computation*, Vol. 70, 2001, pp. 173–203.

[91] A. L. Dontchev and W. W. Hager, "A New Approach to Lipschitz Continuity in State Constrained Optimal Control," *Systems and Control Letters*, Vol. 35, 1998, pp. 137–143.

[92] A. L. Dontchev and W. W. Hager, "Lipschitzian Stability for State Constrained nonlinear Optimal Control," *SIAM Journal on Control and Optimization*, Vol. 36, 1998, pp. 696–718.

[93] G. W. Reddien, "Collocation at Gauss Points as a Discretization in Optimal Control," *SIAM Journal on Control and Optimization*, Vol. 17, March 1979, pp. 298–306.

[94] J. E. Cuthrell and L. T. Biegler, "On the Optimization of Differential-Algebraic Processes," *AIChe Journal*, Vol. 33, August 1987, pp. 1257–1270.

[95] J. E. Cuthrell and L. T. Biegler, "Simultaneous Optimization and Solution Methods for Batch Reactor Control Profiles," *Computers and Chemical Engineering*, Vol. 13, No. 1/2, 1989, pp. 49–62.

[96] P. J. Enright, *Optimal Finite-Thrust Spacecraft Trajectories Using Direct Transcription and Nonlinear Programming*. PhD thesis, Department of Aerospace Engineering, University of Illinois at Urbana-Champaign, 1991.

[97] A. L. Herman, *Improved Collocation Methods with Application to Direct Trajectory Optimization*. PhD thesis, Department of Aerospace Engineering, University of Illinois at Urbana-Champaign, 1995.

[98] 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, May-June 1996, pp. 592–599.

[99] 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. 19, July–August 1996, pp. 994–1002.

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

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

[102] C. G. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods: Evolution to Complex Geometries and Applications to Fluid Dynamics*. Heidelberg, Germany: Spinger-Verlag, 2007.

[103] J. Vlassenbroeck and R. V. Dooren, "A Chebyshev Technique for Solving Nonlinear Optimal Control Problems," *IEEE Transactions on Automatic Control*, Vol. 33, No. 4, 1988, pp. 333–340.

[104] J. Vlassenbroeck, "A Chebyshev Polynomial Method for Optimal Control with State Constraints," *Automatica*, Vol. 24, No. 4, 1988, pp. 499–506.

[105] R. V. Dooren and J. Vlassenbroeck, "A New Look at the Brachistochrone Problem," *Zeitschrift für Angewandte Mathematik und Physik*, Vol. 31, No. 6, 1980, pp. 785–790.

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

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

[108] G. Elnagar and M. Kazemi, "Pseudospectral Chebyshev Optimal Control of Constrained Nonlinear Dynamical Systems," *Computational Optimization and Applications*, Vol. 11, No. 2, 1998, pp. 195–217.

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

[110] P. Williams, "Jacobi Pseudospectral Method for Solving Optimal Control Problems," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 2, 2004, pp. 293–297.

[111] P. Williams, "A Gauss-Lobatto Quadrature Method for Solving Optimal Control Problems," *Australian and New Zealand Industrial and Applied Mathematics Journal*, Vol. 47, 2006, pp. 101–115.

[112] Q. Gong, W. Kang, and I. M. Ross, "A Pseudospectral Method for the Optimal Control of Feedback Linearizable Systems," *IEEE Transactions on Automatic Control*, Vol. 51, July 2006, pp. 1115–1129.

[113] D. A. Benson, *A Gauss Pseudospectral Transcription for Optimal Control*. PhD thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2004.

[114] D. A. Benson, G. T. Huntington, T. P. Thorvaldsen, and A. V. Rao, "Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method," *Journal of Guidance, Control, and Dynamics*, Vol. 29, November-December 2006, pp. 1435–1440.

[115] G. T. Huntington, *Advancement and Analysis of a Gauss Pseudospectral Transcription for Optimal Control*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2007.

[116] A. V. Rao, D. A. Benson, C. L. Darby, C. Francolin, M. A. Patterson, I. Sanders, and G. T. Huntington, "Algorithm: GPOPS, A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using the Gauss Pseudospectral Method," *ACM Transactions on Mathematical Software,* Accepted for Publication, June 2009.

[117] F. Fahroo and I. M. Ross, "Pseudospectral Methods for Infinite-Horizon Nonlinear Optimal Control Problems," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 4, 2008, pp. 927–936.

[118] D. Garg, M. A. Patterson, C. L. Darby, C. Francolin, G. T. Huntington, W. W. Hager, and A. V. Rao, "Direct Trajectory Optimization and Costate Estimation of Finite-Horizon and Infinite-Horizon Optimal Control Problems via a Radau Pseudospectral Method," *Computational Optimization and Applications,* Conditionally Accepted for Publication, June 2009.

[119] O. Axelsson, "Global Integration of Differential Equations Through Lobatto Quadrature," *BIT 4*, Vol. 4, June 1064, pp. 69–86.

[120] I. M. Ross and F. Fahroo, "Convergence of the Costates Does Not Imply Convergence of the Controls," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 4, 2008, pp. 1492–1496.

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

[122] G. N. Elnagar and M. A. Razzaghi, "Pseudospectral Legendre-Based Optimal Computation of Nonlinear Constrained Variational Problems," *Journal of Computational and Applied Mathematics*, Vol. 88, March 1998, pp. 363–375.

[123] F. Fahroo and I. M. Ross, "A Spectral Patching Method for Direct Trajectory Optimization," *Journal of Astronautical Sciences*, Vol. 48, Apr.-Sept. 2000, pp. 269–286.

[124] A. V. Rao, "Extension of a Pseudospectral Legendre Method for Solving Non-Sequential Multiple-Phase Optimal Control Problems," *2003 AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2003-5634, Austin, TX, August 11–14, 2003.

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

[126] F. Fahroo and I. M. Ross, "Costate Estimation by a Legendre Pseudospectral Method," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 2, 2001, pp. 270–277.

[127] I. M. Ross and F. Fahroo, "Legendre Pseudospectral Approximations of Optimal Control Problems," *Lecture Notes in Control and Information Sciences*, pp. 327–342, Springer-Verlag, 2003.

[128] Q. Gong, I. M. Ross, W. Kang, and F. Fahroo, "Connections Between the Covector Mapping Theorem and Convergence of Pseudospectral Methods," *Computational Optimization and Applications*, Vol. 41, December 2008, pp. 307–335.

[129] P. D. Lax and R. D. Richtmyer, "Survey of the Stability of Linear Finite Difference Equations," *Communications on Pure and Applied Mathematics*, Vol. 9, 1956, pp. 267–293.

[130] R. D. Richtmyer and K. W. Morton, *Difference Methods for Initial-Value Problems*. New York: John Wiley & Sons, 1967.

[131] E. Polak, *Optimization: Algorithms and Consistent Approximations*. Heidelberg, Germany: Springer-Verlag, 1997.

[132] A. V. Rao, *Dynamics of Particles and Rigid Bodies: A Systematic Approach*. Cambridge University Press, 2006.

[133] S. Hughes, "GMAT – Generalized Mission Analysis Tool," tech. rep., NASA Goddard Space Flight Center, http://sourceforge.net/projects/gmat, 2008.

[134] W. Squire and G. Trapp, "Using Complex Variables to Estimate Derivatives of Real Functions," *SIAM Review*, Vol. 40, March 1998, pp. 110–112.

[135] J. R. R. Martins, P. Sturdza, and J. J. Alonso, "The Complex-Step Derivative Approximation," *ACM Transactions on Mathematical Software*, Vol. 29, September 2003, pp. 245–262.

[136] J. W. Brown and R. V. Churchill, *Complex Variables and Applications*. McGraw-Hill, 1995.

[137] C. H. Bischof, A. Carle, G. F. Corliss, A. Griewank, and P. D. Hovland, "ADIFOR: Generating Derivative Codes from Fortran Programs," *Scientific Programming*, Vol. 1, No. 1, 1992, pp. 11–29.

[138] C. H. Bischof, A. Carle, P. Khademi, and A. Mauer, "ADIFOR 2.0: Automatic Differentiation of Fortran 77 Programs," *IEEE Computational Science & Engineering*, Vol. 3, No. 3, 1996, pp. 18–32.

[139] C. H. Bischof, H. M. Bücker, B. Lang, A. Rasch, and A. Vehreschild, "Combining Source Transformation and Operator Overloading Techniques to Compute Derivatives for MATLAB Programs," *Proceedings of the Second IEEE International Workshop on Source Code Analysis and Manipulation (SCAM 2002)*, Los Alamitos, CA, USA, IEEE Computer Society, 2002, pp. 65–72.

[140] L. Hascoët and V. Pascual, "TAPENADE 2.1 User's Guide," Rapport technique 300, INRIA, Sophia Antipolis, 2004.

[141] A. Griewank, D. Juedes, and J. Utke, "Algorithm 755: ADOL-C: A Package for the Automatic Differentiation of Algorithms Written in C/C++," *ACM Transactions on Mathematical Software*, Vol. 22, No. 2, 1996, pp. 131–167.

[142] T. F. Coleman and A. Verma, "ADMAT: An Automatic Differentiation Toolbox for MATLAB," tech. rep., Computer Science Department, Cornell University, 1998.

[143] T. F. Coleman and A. Verma, "ADMIT-1: Automatic Differentiation and MATLAB Interface Toolbox," *ACM Transactions on Mathematical Software*, Vol. 26, No. 1, 2000, pp. 150–175.

[144] A. Verma, "ADMAT: Automatic Differentiation in MATLAB Using Object Oriented Methods," *Object Oriented Methods for Interoperable Scientific and Engineering Computing: Proceedings of the 1998 SIAM Workshop* (M. E. Henderson, C. R. Anderson, and S. L. Lyons, eds.), Philadelphia, SIAM, 1999, pp. 174–183.

[145] S. M. Rump, "INTLAB – INTerval LABoratory," *Developments in Reliable Computing* (T. Csendes, ed.), pp. 77–104, Dordrecht: Kluwer Academic Publishers, 1999.

[146] S. A. Forth, "An Efficient Overloaded Implementation of Forward Mode Automatic Differentiation in MATLAB," *ACM Transactions on Mathematical Software*, Vol. 32, jun 2006, pp. 195–222.

[147] H. J. Oberle and W. Grimm, "BNDSCO: A Program for the Numerical Solution of Optimal Control Problems," tech. rep., Institute of Flight Systems Dynamics, German Aerospace Research Establishment DLR, IB 515-89/22, Oberpfaffenhofen, Germany, 1990.

[148] G. L. Brauer, D. E. Cornick, and R. Stevenson, "Capabilities and Applications of the Program to Optimize and Simulate Trajectories," Tech. Rep. NASA-CR-2770, National Aeronautics and Space Administration, 1977.

[149] W. G. Vlases, S. W. Paris, R. M. Lajoie, M. J. Martens, and C. R. Hargraves, "Optimal Trajectories by Implicit Simulation," Tech. Rep. WRDC-TR-90-3056, Boeing Aerospace and Electronics, Wright-Patterson Air Force Base, Ohio, 1990.

[150] C. R. Hargraves and S. W. Paris, "Direct Trajectory Optimization Using Nonlinear Programming Techniques," *Journal of Guidance, Control, and Dynamics*, Vol. 10, July–August 1987, pp. 338–342.

[151] J. T. Betts and W. P. Huffman, "Sparse Optimal Control Software – SOCS," Tech. Rep. MEA-LR-085, Boeing Information and Support Services, Seattle, Washington, July 1997.

[152] J. T. Betts, "Using Sparse Nonlinear Programming to Compute Low Thrust Orbit Transfers," *The Journal of the Astronautical Sciences*, Vol. 41, 1993, pp. 349–371.

[153] J. T. Betts, "Optimal Interplanetary Orbit Transfers by Direct Transcription," *The Journal of the Astronautical Sciences*, Vol. 42, 1994, pp. 247–268.

[154] J. T. Betts, "Very Low Thrust Trajectory Optimization Using a Direct SQP Method," *Journal of Computational and Applied Mathematics*, Vol. 120, 2000, pp. 27–40.

[155] A. V. Rao, S. Tang, and W. P. Hallman, "Numerical Optimization Study of Multiple-Pass Aeroassisted Orbital Transfer," *Optimal Control Applications and Methods*, Vol. 23, July–August 2002, pp. 215–238.

[156] J. T. Betts, "Optimal Lunar Swingby Trajectories," *The Journal of the Astronautical Sciences*, Vol. 55, 2007, pp. 349–371.

[157] C. J. Goh and K. L. Teo, "MISER: A FORTRAN Program for Solving Optimal Control Problems," *Advances in Engineering Software*, Vol. 10, April 1988, pp. 90–99.

[158] O. v. Stryk, *User's Guide for DIRCOL Version 2.1: A Direct Collocation Method for the Numerical Solution of Optimal Control Problems*. Technische Universitat Darmstadt, Darmstadt, Germany, 1999.

[159] "GESOP & ASTOS - The New Generation of Optimization Software," Institute of Flight Mechanics and Control of Stuttgart University, 2003.

[160] M. B. Milam, *Real-Time Optimal Trajectory Generation for Constrained Dynamical Systems*. PhD thesis, California Institute of Technology, Pasadena, California, May 2003.

[161] C. G. Sauer, "MIDAS – Mission Design and Analysis Software for the Optimization of Ballistic Interplanetary Trajectories," *The Journal of the Astronautical Sciences*, Vol. 37, No. 3, 1989, pp. 251–259.

[162] C. Ocampo, "An Architecture for a Generalized Spacecraft Trajectory Design and Optimization System," *Proceedings of the International Conference on Libration Point Missions and Applications*, August 2003.

[163] C. Ocampo, "Finite Burn Maneuver Modeling for a Generalized Spacecraft Trajectory Design and Optimization System," *Annals of the New York Academy of Sciences*, Vol. 1017, pp. 210–233, March 2004.

[164] K. Holmström, "New Optimization Algorithms and Software," *Theory of Stochastic Processes*, Vol. 1–2, 1999, pp. 55–63.

[165] K. Holmström, "The Tomlab Optimization Environment in Matlab," *Advanced Modeling and Simulation*, Vol. 1, 1999, pp. 47–69.

[166] K. Holmström and M. Björkman, "The Tomlab NLPLIB Toolbox for Nonlinear Programming," *Advanced Modeling and Simulation*, Vol. 1, 1999, pp. 70–86.

[167] K. Holmström, M. Björkman, and E. Dotzauer, "The Tomlab OPERA Toolbox for Linear and Discrete Optimization," *Advanced Modeling and Simulation*, Vol. 2, 1999, pp. 1–8.

[168] E. Dotzauer and K. Holmström, "The Tomlab Graphical User Interface for Nonlinear Programming," *Advanced Modeling and Simulation*, Vol. 2, 1999, pp. 9–16.

[169] M. Björkman and K. Holmström, "Global Optimization with the DIRECT User Interface for Nonlinear Programming," *Advanced Modeling and Simulation*, Vol. 2, 1999, pp. 17–37.

[170] A. Schwartz, E. Polak, and Y. Chen, *Recursive Integration Optimal Trajectory Solver (RIOTS_95)*, 1997.

[171] I. M. Ross and F. Fahroo, *User's Manual for DIDO 2001 α: A MATLAB Application for Solving Optimal Control Problems*. Department of Aeronautics and Astronautics, Naval Postgraduate School, Technical Report AAS-01-03, Monterey, California, 2001.

[172] P. Williams, *User's Guide for DIRECT 2.0*. Royal Melbourne Institute of Technology, Melbourne, Australia, 2008.

[173] P. Rutquist and M. Edvall, *PROPT: MATLAB Optimal Control Software*. Tomlab Optimization, Inc, Pullman, Washington, 2008.

[174] T. Jockenhovel, "OPTCONTROLCENTRE, Software Package for Dynamic Optimization," http://OptControlCentre.com/, 2002.

[175] P. J. Gage, R. D. Braun, and I. M. Kroo, "Interplanetary Trajectory Optimization Using a Genetic Algorithm," *Journal of the Astronautical Sciences*, Vol. 43, January 1995, pp. 59–75.

[176] J. W. Hartmann and V. L. Coverstone-Carroll, "Optimal Interplanetary Spacecraft Trajectories via a Pareto Genetic Algorithm," *Journal of the Astronautical Sciences*, Vol. 46, July–September 1998, pp. 267–282.

[177] G. A. Rauwolf and V. L. Coverstone-Carroll, "Near-Optimal Low-Thrust Orbit Transfers Generated by a Genetic Algorithm," *Journal of Spacecraft and Rockets*, Vol. 33, November–December 1996.

[178] V. L. Coverstone-Carroll, J. W. Hartmann, and W. J. Mason, "Optimal Multi-Objective Low-Thrust Spacecraft Trajectories," *Computer Methods in Applied Mechanics and Engineering*, Vol. 186, June 2000.

[179] A. Wuerl, T. Crain, and E. Braden, "Genetic Algorithm and Calculus of Variations-Based Trajectory Optimization Technique," *Journal of Spacecraft and Rockets*, Vol. 40, November–December 2003, pp. 882–888.