JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 19, No. 3, May–June 1996

Direct Optimization Using Collocation Based on High-Order Gauss–Lobatto Quadrature Rules

Albert L. Herman* and Bruce A. Conway†
University of Illinois at Urbana–Champaign, Urbana, Illinois 61801-2935

The method of collocation and nonlinear programming has been used recently to solve a number of optimal control problems. In this method polynomials are commonly used to represent the state variable time histories over subintervals of the total time of interest. These polynomials correspond to a family of modified-Gaussian quadrature rules known as the Gauss–Lobatto rules. Presently, relatively low-order rules from the Gauss–Lobatto family, such as the trapezoid and Simpson's rule, are used to construct collocation solution schemes. In this work higher-order Gauss–Lobatto quadrature rules are formulated using collocation point selection based on a particular family of Jacobi polynomials. The advantage of using a quadrature rule of higher order is that the approximation using the higher degree polynomial may be more accurate, due to finite precision arithmetic, than a formulation based on a lower degree polynomial. In addition, the number of subintervals and, therefore, the number of nonlinear programming parameters needed to solve a problem accurately may be significantly reduced from that required if the conventional trapezoidal or Simpson's quadrature schemes are used. An optimal trajectory maximizing final energy for a low-thrust spacecraft is used to demonstrate the benefits of using the higher-order schemes.

Introduction

THIS paper describes work accomplished in developing methods for finding accurate numerical solutions to optimal low-thrust orbit transfer problems based on direct collocation. In the direct optimization technique used, the optimal control problem is converted into a nonlinear programming (NLP) problem. The solution time history is discretized, and polynomial interpolants are used to approximate solutions over subintervals of the total time history. These polynomials are computed using collocation point selection based on Jacobi polynomials. The resulting polynomial interpolants take on the form of a family of modified-Gaussian quadrature rules known as the Gauss–Lobatto rules. These quadrature rules are applied to solving an optimal orbit transfer problem where the benefits of the higher-order schemes are demonstrated.

The method of direct collocation has recently been used by a number of researchers to solve orbit transfer problems. Optimal trajectories for spacecraft using finite thrust were found by Enright and Conway[1,2] using direct collocation. The problems they solved include optimal rendezvous and transfer, and optimal multiburn orbit insertion from hyperbolic approach. They check the solution accuracy by comparing the direct collocation solution with the calculus-of-variations (COV) first-order necessary conditions. Scheel and Conway[3] demonstrate the use of a direct method for the problem of very-low-thrust transfers from planar circular orbits and do a similar a posteriori check of the COV necessary conditions. In none of these three investigations, however, are examples given of inaccurate solutions or what may be done to improve solution accuracy. Betts[4] applies a direct method to solving low-thrust orbit transfers for spacecraft that undergo a large change in orbit inclination. In addition, Betts[5] uses a direct method for finding optimal interplanetary orbit transfers. However, the numerical accuracy of the solutions presented is not demonstrated. Pierson and Kluever[6] use a direct method as part of a solution for optimal planar, Earth–moon orbit transfers. They compare the resulting solution with that obtained from solving the two-point boundary-value problem (TPBVP) that results from applying COV to the optimal control problem. They state that the solutions were identical without providing details. Tang and Conway[7] solve a low-thrust, Earth–Mars transfer problem using direct collocation (without providing estimates of accuracy). Recently, demonstration of the accuracy of computational results has become important.[8] In this work, methods for accurately determining numerical approximations to optimal low-thrust orbit transfer problems are presented.

This discussion begins by describing the COV necessary conditions for optimal control. These optimality conditions, however, are not used explicitly in the process of finding a solution, but, instead, are used in an a posteriori check of the accuracy of a solution. Next, the numerical solution method used to solve the optimal control problem is described. Then, these improved methods for numerically solving optimal control problems are applied to the example of a low-thrust orbit transfer problem.

Optimal Control Problem

In this work, the control histories that take a set of states from specified initial conditions to their desired final conditions are found. In addition to satisfying the state boundary conditions, these control histories are to minimize a function that is a function of the final values of the states. These states are governed by a system of first-order, ordinary differential equations (ODE) given by

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{1}$$

where $\boldsymbol{x}$ is an $n \times 1$ vector of states, $\boldsymbol{u}$ is an $m \times 1$ vector of controls, and $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ is an $n \times 1$ vector representing algebraic functions of the states and controls that are termed system equations. The initial conditions for the states are

$$\boldsymbol{x}(t_I) = \boldsymbol{x}_I \tag{2}$$

and the desired final conditions are

$$\boldsymbol{\Psi}[\boldsymbol{x}(T)] = \boldsymbol{0} \tag{3}$$

where $\boldsymbol{\Psi}$ is a $q \times 1$ vector of algebraic functions of the states at a fixed final time T. The control time histories are to be chosen so that a performance function is minimized while satisfying the system equations given in Eq. (1) with initial conditions given in Eq. (2) and final condition constraints given in Eq. (3). This performance function is

$$J = \phi[\boldsymbol{x}(T)] \tag{4}$$

Received Sept. 16, 1994; revision received July 21, 1995; accepted for publication Aug. 21, 1995. Copyright © 1995 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Graduate Research Assistant, Department of Aeronautical and Astronautical Engineering. Member AIAA.

†Professor, Department of Aeronautical and Astronautical Engineering. Associate Fellow AIAA.

where ϕ is a scalar function of the values of the states at the final time.

An approach[9,10] to solving this optimal control problem is to adjoin the system differential equations given in Eq. (1) along with the final condition constraints given in Eq. (3) to the performance function given in Eq. (4) (the initial conditions are satisfied directly). This formulation results in the augmented performance functional given by

$$\bar{J} = \phi + \nu^T \Psi + \int_{t_I}^{T} \left\{ \lambda^T \left[f(x, u) - \frac{\mathrm{d}x}{\mathrm{d}t} \right] \right\} \mathrm{d}t \tag{5}$$

where λ is an $n \times 1$ vector known as costates and ν is a $q \times 1$ vector of multiplier variables associated with the final condition constraints. The COV first-order necessary conditions[9] for optimality of the augmented performance functional include the costate system equations

$$\frac{\mathrm{d}\lambda}{\mathrm{d}t} = -\frac{\partial H^T}{\partial x} \tag{6}$$

where $H = \lambda^T f$ (called the system Hamiltonian[9]). Boundary conditions for the costate variables are given only at the final time. These final conditions are

$$\lambda(T) = \frac{\partial \Phi^T}{\partial x(T)} \tag{7}$$

where $\Phi = \phi + \nu^T \Psi$. An additional condition for optimality is

$$\frac{\partial H^T}{\partial u} = \mathbf{0} \tag{8}$$

The remaining optimality condition is that the solution to Eqs. (6–8) must also satisfy the system equations given in Eq. (1) with initial conditions given in Eq. (2) and final condition constraints given in Eq. (3). This system of equations thus constitutes a TPBVP and may be used in an indirect solution process for the original optimal control problem given in Eqs. (1–4).

This TPBVP is very sensitive to the boundary values of the problem making it difficult to solve. Even if a method for solving this TPBVP were found that provides a robust solution process, however, it still would require repeated derivation of the Euler–Lagrange equations for each new problem. The goal of this work is to solve directly the original formulation of the optimal control problem given in Eqs. (1–4) without resorting to the use of the Euler–Lagrange equations, i.e., directly, meaning without the use of the costate variables or Euler–Lagrange equations.

Numerical Solutions

The solution method employed here requires that the continuous variables be represented (or approximated) by discrete variables, so that the problem becomes one of constrained parameter optimization.[1,2,11–13] This approach discretizes the solution time history into L subintervals. The endpoints of these subintervals are denoted as $\{t_0, t_1, \ldots, t_i, t_{i+1}, \ldots, t_{L-1}, t_L\}$, where t_0 is the initial time t_I and t_L is the final time T. An example of a discretized time history with a total of eight subintervals is shown in Fig. 1. Within a given subinterval $[t_i, t_{i+1}]$ it is desired to approximate the time history of a solution including the numerical integration of the system equations $f(x, u)$.

Next, the basic rules for numerical integration are given followed by a presentation of methods for producing integration rules using collocation based on a particular family of Jacobi polynomials (which result in the Gauss–Lobatto quadrature rules). These rules are then used to formulate solution methods for solving differential equations. These formulations are then used to directly solve the optimal control problem.

Basic Methods

The solution to the equation $\mathrm{d}x/\mathrm{d}t = f(t)$ may be cast into the form

$$\int_{t_i}^{t_{i+1}} \mathrm{d}x(t) = x(t_{i+1}) - x(t_i) = \int_{t_i}^{t_{i+1}} f(t)\,\mathrm{d}t \approx \Delta t_i f(t_i) \tag{9}$$

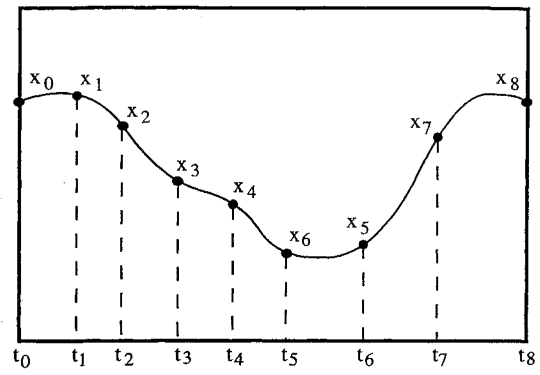

Fig. 1 Solution time history discretization.

where $\Delta t_i = t_{i+1} - t_i$, resulting in the well-known approximation called the rectangle rule. There are many other numerical integration schemes, for example the well-known trapezoidal rule

$$\int_{t_i}^{t_{i+1}} f(t)\,\mathrm{d}t \approx \frac{\Delta t_i}{2}[f(t_i) + f(t_{i+1})] \tag{10}$$

In the numerical approximation given in Eq. (10), the integrand is approximated with a linear function. Another method of approximating $f(t)$ over a subinterval is to use a polynomial interpolant. One method that uses a polynomial interpolant is the well-known Simpson's rule. In this rule, the integrand is approximated using a quadratic polynomial that evaluates to the integrand at the endpoints t_i and t_{i+1} of the subinterval and at the center point of the subinterval $t_c = (t_i + t_{i+1})/2$. This quadratic interpolant is integrated once, resulting in the approximation

$$\int_{t_i}^{t_{i+1}} f(t)\,\mathrm{d}t \approx \frac{\Delta t_i}{6}[f(t_i) + 4f(t_c) + f(t_{i+1})] \tag{11}$$

The method of Simpson's rule uses three discrete values of the integrand. Two of these discrete values are arbitrarily chosen to correspond to the endpoints of the subinterval $[t_i, t_{i+1}]$. The remaining value is determined using the center point of the interval t_c. These three evaluation points are termed the collocation points, i.e., the slope of the approximating polynomial is collocated with the integrand at these points. A question arises as to where to locate additional collocation points to be used in the case when the interpolant is of higher degree than the quadratic interpolant used in Simpson's rule. It can be shown that this choice is related to the error in the approximation.

Error Analysis

It is desired that the error of the approximate integration be of the highest order possible. By this it is meant that the function that represents the error in the approximate integration is a function of the subinterval length Δt_i to an integer power that is as large as possible. Let $p_m(t)$ be a polynomial in time with the highest power of t being m. This polynomial is constructed by choosing the coefficients associated with each term so that the resulting polynomial evaluates to the integrand, $f(t)$, at selected points in the interval $[t_i, t_{i+1}]$. Thus, the integration of $p_m(t)$ is an approximation to the integral of $f(t)$. Let the error in the integral of $p_m(t)$ over the interval $[t_i, t_{i+1}]$ be defined by

$$E(f) = \int_{t_i}^{t_{i+1}} f(t)\,\mathrm{d}t - \int_{t_i}^{t_{i+1}} p_m(t)\,\mathrm{d}t \tag{12}$$

which is called the local discretization error (or, more precisely, called the local truncation error). The local truncation errors for the rectangle and trapezoid rules are proportional to Δt_i^2 and Δt_i^3, respectively. The derivation of local truncation error may be found in several references.[14–17] The power on the subinterval length Δt_i of the trapezoid rule error is higher than that of the error for the rectangle rule. Thus, the trapezoid rule is of higher order than the rectangle rule. As Δt_i is reduced (thus increasing the number of subintervals in a time history discretization), the trapezoid rule approximation improves cubically, whereas rectangle rule improves only quadratically. The local truncation error for Simpson's rule is proportional to Δt_i^5.

The order of accuracy is that power of the stepsize Δt_i to which the truncation error is proportional. The order of accuracy is important in determining the accuracy of a method due to finite precision arithmetic used in computer solutions. Skeel and Keiper[18] state that "the order of accuracy is the first and foremost measure of a method's power to approximate." In addition, concerning the choice of stepsize, Conte and de Boor[14] state the following:

> The accuracy of a numerical integration will depend upon the discretization error and the accumulated rounding error. To keep the discretization error small, we will normally choose the step size h small. On the other hand, the smaller h is taken, the more integration steps we shall have to perform, and the greater the rounding error is likely to be. There is, therefore, an optimum value of the step size h which for a given machine and a given problem will result in the best accuracy. This optimum is in practice very difficult to find without the use of extensive amounts of computer time. The existence of such an optimum does show, however, that there is some **danger** in taking too small a step size.

For ODEs the discretization error of concern is the global truncation error that is a measure of the error in propagating a set of states from an initial state to the resulting final state using some discretization scheme. The global truncation error results from using a method to integrate a function over an interval that has been subdivided. Thus, the global truncation error results from the accumulation of local truncation errors. Because the stepsize and the number of local truncation error values to be summed are both proportional to the number of subintervals, the global truncation error is of order one less than the local truncation error. For example, if one were to use Simpson's rule for integrating a function with L total intervals of equal length $\Delta t_i = (T - t_I)/L$ or $L = (T - t_I)/\Delta t_i$. Then, the global truncation error is the sum from 1 to L of local truncation errors for each subinterval, resulting in a sum of functions that are $\mathcal{O}(h^5)$. Hence, this global error is proportional to $L\Delta t_i^5 = (T - t_I)\Delta t_i^4$, making the order of accuracy of Simpson's rule equal to 4. The greater is the order of accuracy, the greater is the reduction in error if the stepsize is made smaller. Therefore, as the order of accuracy increases, a specified accuracy may be achieved with larger stepsizes, reducing the accumulation of rounding errors. As a rule of thumb one should choose the order of the scheme to equal the number of digits of accuracy desired.[19]

As mentioned previously, a question arises as to the choice of additional collocation points to be used to derive additional integration rules with orders of accuracy higher than Simpson's rule. In this work, the interpolating polynomial will always use discrete values of the integrand at the endpoints of the subintervals. Thus, only the collocation points internal to a subinterval must be determined.

Collocation Point Determination

The collocation points internal to a subinterval used to formulate an integration rule are chosen to increase the order of accuracy of the resulting integration rule to the highest order possible. The resulting integration rules are a family of modified Gaussian integration rules known as the Gauss–Lobatto rules.[15–17] The collocation points that maximize the power of Δt_i in the local truncation error are the roots of the corresponding Jacobi polynomial. Jacobi polynomials are the family of polynomials that are orthogonal on the interval $[-1, +1]$ with respect to the weight function $w(s) = (1-s)^\alpha(1+s)^\beta$. In the Gauss–Lobatto rules, $\alpha = \beta = 1$. A subinterval with endpoints $[t_i, t_{i+1}]$ may be transformed to the interval $[-1, +1]$ using the transformation $s = 2(t - t_i)/\Delta t_i - 1$. The interpolating polynomial may then be formulated by interpolating $f(t)$ at the endpoints of the interval $[-1, +1]$ and at the zeros of the corresponding Jacobi polynomial. For a quadratic polynomial interpolant, the corresponding Jacobi polynomial has one root with a value of zero, which is the center point of the subinterval. This interpolant results in Simpson's rule. Thus, the Gauss–Lobatto family of integration rules begins with the trapezoid and Simpson's rules and continues with higher order rules. For the Gauss–Lobatto integration rules, the degree of the integrated polynomial matches the number of discrete values of the integrand used to formulate the interpolating polynomial. Thus, the trapezoid rule is the second-degree rule and Simpson's rule is the third-degree Gauss–Lobatto integration rule. For the fourth-degree Gauss–Lobatto integration rule the roots of the corresponding Jacobi polynomial are $\{-\sqrt{(1/5)}, \sqrt{(1/5)}\}$, yielding the approximate integration rule

$$\int_{t_i}^{t_{i+1}} f(t)\,\mathrm{d}t \approx \frac{\Delta t_i}{12}\Big[f(t_i) + 5f\Big(t_c - \sqrt{1/5}\Delta t_i\Big) + 5f\Big(t_c + \sqrt{1/5}\Delta t_i\Big) + f(t_{i+1})\Big] \tag{13}$$

which has an order of accuracy of 6. Thus, the fourth-degree integration rule may result in a more accurate solution than Simpson's rule when considering finite precision arithmetic. For the Gauss–Lobatto fifth-degree integration rule, the collocation points are $\{-\sqrt{(3/7)}, 0, \sqrt{(3/7)}\}$ yielding the approximate integration rule

$$\int_{t_i}^{t_{i+1}} f(t)\,\mathrm{d}t \approx \frac{\Delta t_i}{180}\Big[9f(t_i) + 49f\Big(t_c - \sqrt{3/7}\Delta t_i\Big) + 64f(t_c) + 49f\Big(t_c + \sqrt{3/7}\Delta t_i\Big) + 9f(t_{i+1})\Big] \tag{14}$$

which has an order of accuracy of 8.

Application to Differential Equations

If one wishes to find a solution to differential equation of the form $\mathrm{d}x/\mathrm{d}t = f(x)$, it is straightforward to apply the rectangle rule of Eq. (9). With an initial value for the state, the formula $x_{i+1} = x_i + \Delta t_i f(x_i)$ may be applied iteratively to propagate x for specified values of Δt_i. This form of the rectangle rule applied to a differential equation is the well-known Euler's method. For known values x_i, t_i, and t_{i+1}, a value for x_{i+1} may be computed. This method is explicit in that the value x_{i+1} is not used to compute a discrete value of the integrand $f(x)$. Applying the trapezoid rule to the differential equation results in the implicit method

$$x_{i+1} = x_i + (\Delta t_i/2)[f(x_i) + f(x_{i+1})] \tag{15}$$

Here, x_{i+1} is used to compute a discrete value of the differential equation, thus making this method an implicit method. An approach to using this method is to solve Eq. (15) analytically for x_{i+1} and then use the resulting explicit formula for x_{i+1} to propagate x. It may not be possible, however, to solve Eq. (15) for x_{i+1} analytically, and even if it is possible for a particular problem, the resulting formula would not be generally applicable. Therefore, an alternative method for solving Eq. (15) is to formulate it as the constraint

$$C_T(x_i, x_{i+1}) = x_i - x_{i+1} + (\Delta t_i/2)[f_i + f_{i+1}] = 0 \tag{16}$$

termed the trapezoid system constraint (explicit functional dependencies are omitted). This equation may be solved for x_{i+1} by some numerical technique resulting in a general method.

For Simpson's rule, the derivation of an analogous constraint is not straightforward. Remembering that Simpson's rule is formulated assuming a quadratic approximation of the integrand $f(x)$, the state time history must be approximated by a cubic polynomial. This polynomial represents an integration of the polynomial used to interpolate $f(x)$ at the endpoints and center points of a subinterval. In this case parameters representing the state at the endpoints are used to formulate a constraint. No parameters, however, are used to represent the state internal to the interval (as is the case when Simpson's rule is used as the corrector step in a multistep method). The endpoint information available is the parameter values of the states x_i and x_{i+1} and the system equation values f_i and f_{i+1}. Using these four pieces of information, a Hermite-cubic polynomial representing the state time history between the endpoint times t_i and t_{i+1} may be constructed,[11,12] as shown in Fig. 2, and used to compute a value of the state at the center point of the subinterval, yielding

$$x_c = \tfrac{1}{2}(x_i + x_{i+1}) + (\Delta t_i/8)(f_i - f_{i+1}) \tag{17}$$

where x_c is a discrete approximation for $x(t)$ at t_c. Simpson's system constraint is then formulated using x_c to evaluate the system

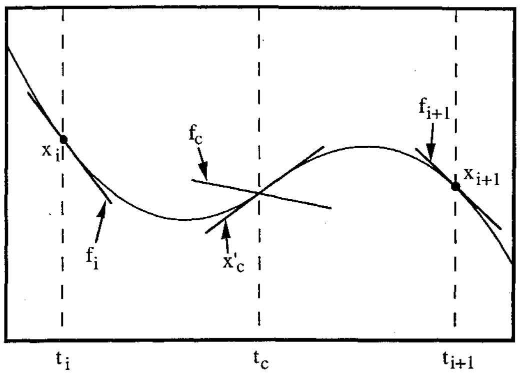

Fig. 2 Simpson's system constraint formulation.

equation resulting in a discrete value of the system equation at the center point of the subinterval, $f_c = f(x_c)$. Then equating f_c with the first derivative in time of the Hermite-cubic polynomial evaluated at the center point of the subinterval x'_c yields the system constraint

$$C_S(x_i, x_{i+1}) = x_i - x_{i+1} + (\Delta t_i/6)[f_i + 4f_c + f_{i+1}] = 0 \quad (18)$$

This constraint has been termed the Hermite–Simpson system constraint.[20]

For the fourth-degree Gauss–Lobatto integration rule, the process of transforming the integration rule to yield a constraint is analogous to the process just described for the Simpson system constraint but is more complex and yields two system constraints per subinterval. In this case the resulting state time history is approximated by a fourth-degree polynomial requiring five pieces of information. Four pieces of information are available at the subinterval endpoints $(x_i, f_i, x_{i+1}, f_{i+1})$ as shown in Fig. 3; thus, one additional parameter per subinterval is required in the discretization of the problem. This parameter is chosen to be the value of the state at the center point of the interval x_c. This polynomial interpolant, as shown in Fig. 3, is then used to compute discrete values for the state at the two interval collocation points, t_1 and t_2, for this integration rule. These values are

$$x_1 = \tfrac{1}{50}\{(7\sqrt{5}+9)x_i + 32x_c + (-7\sqrt{5}+9)x_{i+1} + \Delta t_i[(\sqrt{5}+1)f_i + (\sqrt{5}-1)f_{i+1}]\} \quad (19a)$$

$$x_2 = \tfrac{1}{50}\{(-7\sqrt{5}+9)x_i + 32x_c + (7\sqrt{5}+9)x_{i+1} + \Delta t_i[(-\sqrt{5}+1)f_i + (-\sqrt{5}-1)f_{i+1}]\} \quad (19b)$$

where x_1 and x_2 are discrete approximations for $x(t)$ at $t_c - \sqrt{(1/5)}\frac{1}{2}\Delta t_i$ and $t_c + \sqrt{(1/5)}\frac{1}{2}\Delta t_i$, respectively. The fourth-degree Gauss–Lobatto system constraints are then formulated at the internal collocation points using x_1 and x_2 to evaluate the system equation resulting in the discrete values f_1 and f_2, respectively, as illustrated in Fig. 3. These discrete values for the system equation are then equated with the first derivative in time of the fourth-degree polynomial representing the state time history evaluated at the same points, x'_1 and x'_2 shown in Fig. 3. The resulting system constraints are

$$C_{4,1}(x_i, x_{i+1}) = \tfrac{1}{120}\{(32\sqrt{5}+60)x_i - 72\sqrt{5}x_c + (32\sqrt{5}-60)x_{i+1} + \Delta t_i[(5+3\sqrt{5})f_i + 50f_1 + (5-3\sqrt{5})f_{i+1}]\} = 0 \quad (20a)$$

$$C_{4,2}(x_i, x_{i+1}) = \tfrac{1}{120}\{(-32\sqrt{5}+60)x_i + 72\sqrt{5}x_c + (-32\sqrt{5}-60)x_{i+1} + \Delta t_i[(5-3\sqrt{5})f_i + 50f_2 + (5+3\sqrt{5})f_{i+1}]\} = 0 \quad (20b)$$

where $f_1 = f(x_1)$ and $f_2 = f(x_2)$. Neither of these constraints takes on the expected form for the fourth-degree integrator. If added together (providing both are satisfied), however, they do recover

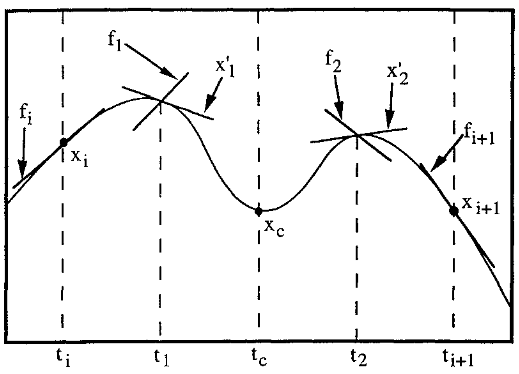

Fig. 3 Fourth-degree Gauss–Lobatto system constraint formulation.

the fourth-degree Gauss–Lobatto integration rule given in Eq. (13). Therefore, the constraints given in Eq. (20) are termed the fourth-degree Gauss–Lobatto system constraints.

A one-system-constraint-per-subinterval formulation for the fourth-degree Gauss–Lobatto rule can be constructed. Such a formulation, however, results in an NLP problem with inadequate properties. This inadequacy occurs because the parameters representing a particular state internal to an interval do not explicitly appear in the resulting system constraint. Thus, if a particular state variable is not included in any of the system equations, the resulting NLP problem will not include the associated internal subinterval state parameter x_c in the problem formulation. Therefore, no information is available for correctly choosing these parameters.

For the fifth-degree Gauss–Lobatto integration rule, the process of transforming the integration rule is similar to that for the fourth-degree integration rule. In this case, the resulting state time history approximation is a fifth-degree polynomial requiring six pieces of information. Again, a parameter x_c is added, representing the discrete value of the state at the center point of the subinterval (which is one of the three collocation points internal to the subinterval). In addition, this value is used to compute a discrete value of the integrand at the center point of the interval f_c, which becomes the sixth parameter. The resulting polynomial interpolant is used to compute values for the state at the remaining collocation points, which are

$$x_1 = \tfrac{1}{686}\{(39\sqrt{21}+231)x_i + 224x_c + (-39\sqrt{21}+231)x_{i+1} + \Delta t_i[(3\sqrt{21}+21)f_i - 16\sqrt{21}f_c + (3\sqrt{21}-21)f_{i+1}]\} \quad (21a)$$

$$x_3 = \tfrac{1}{686}\{(-39\sqrt{21}+231)x_i + 224x_c + (39\sqrt{21}+231)x_{i+1} + \Delta t_i[(-3\sqrt{21}+21)f_i + 16\sqrt{21}f_c + (-3\sqrt{21}-21)f_{i+1}]\} \quad (21b)$$

where x_1 and x_3 are discrete approximations for $x(t)$ at $t_c - \sqrt{(3/7)}\frac{1}{2}\Delta t_i$ and $t_c + \sqrt{(3/7)}\frac{1}{2}\Delta t_i$, respectively. The fifth-degree Gauss–Lobatto system constraints are then formulated using x_1 and x_3 to evaluate the system equation, resulting in the discrete values f_1 and f_3, respectively. These discrete values are then the values of the system equations that correspond to the two remaining internal collocation points. The resulting system constraints are

$$C_{5,1}(x_i, x_{i+1}) = \tfrac{1}{360}\{(32\sqrt{21}+180)x_i - 64\sqrt{21}x_c + (32\sqrt{21}-180)x_{i+1} + \Delta t_i[(9+\sqrt{21})f_i + 98f_1 + 64f_c + (9-\sqrt{21})f_{i+1}]\} = 0 \quad (22a)$$

$$C_{5,3}(x_i, x_{i+1}) = \tfrac{1}{360}\{(-32\sqrt{21}+180)x_i + 64\sqrt{21}x_c + (-32\sqrt{21}-180)x_{i+1} + \Delta t_i[(9-\sqrt{21})f_i + 98f_3 + 64f_c + (9+\sqrt{21})f_{i+1}]\} = 0 \quad (22b)$$

where $f_1 = f(x_1)$ and $f_3 = f(x_3)$. The constraints given in Eq. (22) are termed the fifth-degree Gauss–Lobatto system constraints.

Note that with the fourth- and fifth-degree methods, the interpolating polynomial is a Hermite polynomial where the states and system equations are used to form the polynomials (analogous to multistep methods). The resulting polynomials are optimal in the sense that they are the best approximation possible to the integral of the system equations given the a priori selection of the endpoints as evaluation points. This result is accomplished by the use of orthogonal polynomials, similar to those methods used for constructing the best interpolating polynomials of functions.

Methods for numerically solving ODEs calculate a sequence of approximations $\{x_0, x_1, \ldots, x_L\}$ to the solution of Eq. (1) on a mesh of time values $\{t_0, t_1, \ldots, t_L\}$. Most methods (e.g., a four-stage Runge–Kutta integration algorithm) compute this sequence of approximations serially, using the past r points $\{(t_{n-r}, x_{n-r}), \ldots, (t_{n-1}, x_{n-1})\}$ to calculate the next s approximations $\{(t_n, x_n), \ldots, (t_{n+s-1}, x_{n+s-1})\}$. This recursion may diverge unbounded from the correct solution of the ODEs as the sequence progresses because of the instability of the recursion formula that results from the application of a numerical solver. In this work no recursion formula is used. Instead, the sequence of approximations $\{x_1, x_2, \ldots, x_n\}$ is related through a set of constraints, solving for the entire sequence in parallel. In addition, the set of collocation methods used in this work for numerically solving the ODEs have symmetric interpolation points in each subinterval. This set of methods is stable at least in a finite interval for a general class of ODEs, thus providing good stability characteristics.[21,22] Therefore, these new methods have high-order accuracy as well as good stability properties, providing for good approximations for the states over an interval of integration.[23]

In each of these four differential equation integration methods, there are n more state variable parameters than constraints. For the trapezoid and Simpson's methods, there are $2n$ state variable parameters and n system constraints per subinterval, and for the fourth- and fifth-degree methods there are $3n$ state variable parameters and $2n$ constraints per subinterval. If the parameters that represent the states at the left endpoint t_i are constrained, then each formulation will have the same number of variables and constraints. This is analogous to the initial conditions given in Eq. (2) for the system differential equations given in Eq. (1). If a time history is subdivided into a total of L subintervals, there will yet be only n state variable parameters more than there are system constraints, again allowing for the application of constraints to specify initial conditions, final conditions, or a mix of n boundary conditions.

Note that the derivations of the fourth- and fifth-degree formulations were accomplished using the symbolic mathematics computer program Maple.[24]

Direct Solutions

To find an optimal control history, the system equations given in Eq. (1) are numerically solved using one of the four sets of Gauss–Lobatto system constraints while applying the boundary conditions given in Eqs. (2) and (3). The control variables are discretized by providing parameters representing discrete values for the controls for each discrete time at which the system equations are evaluated. Such controls are termed free controls[20] because the only relationship between parameters used to discretize a control variable is in the system constraints used to solve a problem, e.g., the control variable representation is not constrained to be piecewise cubic. For example, using the fourth-degree Gauss–Lobatto system constraints given in Eq. (20) to solve the system (1–3), four sets of control parameters are provided in each subinterval, $\boldsymbol{u}_i, \boldsymbol{u}_{i,1}, \boldsymbol{u}_{i,2}$, and $\boldsymbol{u}_{i+1}$, representing discrete values of the control variables at the subinterval endpoints and at the two internal collocation points. These parameters are optimally chosen to minimize the performance function given in Eq. (4) while satisfying the chosen system constraints along with the initial conditions given in Eq. (2) and the final condition constraints given in Eq. (3). Such a formulation takes on the following form of an NLP problem.

Minimize:

$$\phi[\boldsymbol{x}_L]$$

Subject to:

$$C_{4,1}(\boldsymbol{x}_{i-1}, \boldsymbol{x}_{i,c}, \boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{u}_{i,1}, \boldsymbol{u}_{i+1}) = 0, \qquad \text{for} \quad i = 1, 2, 3, \ldots, L$$

$$C_{4,2}(\boldsymbol{x}_{i-1}, \boldsymbol{x}_{i,c}, \boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{u}_{i,2}, \boldsymbol{u}_{i+1}) = 0, \qquad \text{for} \quad i = 1, 2, 3, \ldots, L$$

$$\boldsymbol{x}_0 - \boldsymbol{x}_I = 0 \qquad \Psi[\boldsymbol{x}_L] = 0$$

The algorithm used in this work to solve this NLP problem is NZOPT.[25] NZOPT is a set of Fortran subroutines, based on a sequential quadratic programming algorithm, for minimizing a smooth function subject to constraints, that may include simple bounds on the parameters, linear constraints, and smooth nonlinear constraints. All work was performed on a Cray C90. The partial derivatives for this method are computed analytically, and the constraint satisfaction tolerance is chosen by the algorithm to be the square root of the relative machine precision, which is approximately 7.11×10^{-15} (which gives a constraint tolerance approximately 8.43×10^{-8}).

Once a solution to the NLP problem is found, an approximation to the solution of the original problem has been computed. The accuracy of this numerical solution may be checked by integrating the system Euler–Lagrange equations, i.e., the costate differential equations (6) coupled with the system differential equations (1), backward from the final time T to the initial time t_I. The final conditions for the states $\boldsymbol{x}_L$ are taken from the NLP solution as are any needed multiplier variables $\boldsymbol{\nu}$ (Ref. 20). These values may be used to compute final values for the costates $\boldsymbol{\lambda}(T)$ using Eq. (7). Then, integrating background from the final time T to the initial time t_I and comparing state values recovered from this backward integration with the initial state values given in Eq. (2), the accuracy of the NLP solution may be checked.

Low-Thrust Spacecraft Orbit Transfer

As an example of the use of the optimization methods described, an orbit transfer problem in which the rocket engine imparts a constant thrust acceleration to the spacecraft is considered. Motion is confined to a single plane. The spacecraft's position is described with polar coordinates $\{r, \theta\}$ that have their origin located at the center of mass of the attracting body, as illustrated in Fig. 4. In addition, the radial and tangential velocities v_r and v_t, respectively, are shown. Thus, the spacecraft's velocity vector $\boldsymbol{V} = \{v_r, v_t\}^T$. The only control variable is the thrust angle β that is measured relative to the local horizontal. Finally, the thrust acceleration A is the thrust divided by the mass of the spacecraft and is considered to be of constant magnitude for this investigation. The state variables are then $\boldsymbol{x} = \{r, \theta, v_r, v_t\}^T$ resulting in the system equations

$$\frac{\mathrm{d}r}{\mathrm{d}t} = v_r \tag{23}$$

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = \frac{v_t}{r} \tag{24}$$

$$\frac{\mathrm{d}v_r}{\mathrm{d}t} = \frac{v_t^2}{r} - \frac{\mu}{r^2} + A\sin(\beta) \tag{25}$$

$$\frac{\mathrm{d}v_t}{\mathrm{d}t} = -\frac{v_r v_t}{r} + A\cos(\beta) \tag{26}$$

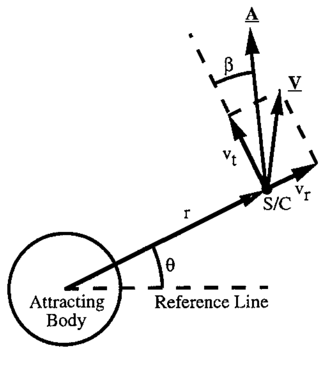

Fig. 4 Two-dimensional polar coordinates describing spacecraft trajectory: S/C, spacecraft; V, spacecraft velocity; A, thrust acceleration; and β, thrust direction.

The length units (LU) and time units (TU) are chosen such that the gravitational constant μ is unity. (In these canonical variables, 2π time units would represent the period of a circular orbit at the surface of the attracting body.) The constant acceleration magnitude A, the initial time t_I, and the final time T are arbitrarily chosen to be 0.01, 0.0, and 50.0, respectively. For this problem, the performance function is the total specific energy at the final time T that is to be maximized. Thus,

$$\phi[x(T)] = -\left\{\tfrac{1}{2}\left[v_r(T)^2 + v_t(T)^2\right] - [1/r(T)]\right\} \tag{27}$$

Also, the initial conditions for the states correspond to a circular orbit at a radius of 1.1 LU, resulting in

$$x_I = \left\{1.1, 0, 0, 1/\sqrt{1.1}\right\}^T \tag{28}$$

There are no final condition constraints and, therefore, $\Psi[x(T)]$ is identically equal to the zero vector (thus, $\nu \equiv 0$).

A Posteriori Verification of the Euler–Lagrange Equations

As described in the preceding section, neither the system costate variables, defined in Eqs. (5–7), nor the optimality condition (8) are explicitly used in the NLP solution for the optimal control. Backward integration of the Euler–Lagrange equations from final values of the states and costates given by the NLP problem solver, however, may be used as an a posteriori check on the optimality of the solution.

For this system, the costate variables are $\lambda = \{\lambda_r, \lambda_\theta, \lambda_{v_r}, \lambda_{v_t}\}^T$ and the system Hamiltonian becomes

$$\begin{aligned} H &= \lambda_r(v_r) + \lambda_\theta(v_t/r) + \lambda_{v_r}\left[\left(v_t^2/r\right) - (1/r^2) + A\sin(\beta)\right] \\ &\quad + \lambda_{v_t}[-(v_r v_t/r) + A\cos(\beta)] \end{aligned} \tag{29}$$

The costate system equations given by Eq. (6) become

$$\frac{\mathrm{d}\lambda_r}{\mathrm{d}t} = \lambda_\theta\left(\frac{v_t}{r^2}\right) + \lambda_{v_r}\left(\frac{v_t^2}{r^2} - \frac{2}{r^3}\right) - \lambda_{v_t}\left(\frac{v_r v_t}{r^2}\right) \tag{30}$$

$$\frac{\mathrm{d}\lambda_\theta}{\mathrm{d}t} = 0 \tag{31}$$

$$\frac{\mathrm{d}\lambda_{v_r}}{\mathrm{d}t} = -\lambda_r + \lambda_{v_t}\left(\frac{v_t}{r}\right) \tag{32}$$

$$\frac{\mathrm{d}\lambda_{v_t}}{\mathrm{d}t} = -\lambda_\theta\left(\frac{1}{r}\right) - \lambda_{v_r}\left(\frac{2v_t}{r}\right) + \lambda_{v_t}\left(\frac{v_r}{r}\right) \tag{33}$$

and the final conditions for the costates given by Eq. (7) become

$$\lambda(T) = \left\{-1/r(T)^2, 0, -v_r(T), -v_t(T)\right\}^T \tag{34}$$

The optimality condition given in Eq. (8) becomes

$$\tan(\beta) = -\lambda_{v_r}/-\lambda_{v_t} \tag{35}$$

Equations (30–33) may thus be integrated simultaneously with the system (23–26) using the optimal control given in Eq. (35) and the final values for the states that result from the NLP solution and the final values for the costates given in Eq. (34). The accuracy with which the initial states are recovered is then a measure of the accuracy of the discrete (NLP) solution for the optimal control problem.

Results

The resulting optimal trajectory when using the fifth-degree system constraint with 30 subintervals is shown in Fig. 5. As can be seen, the trajectory is a multirevolution spiral away from the attracting body. The optimal thrust pointing angle is shown in Fig. 6. The results from the direct and COV methods are both shown in these figures; differences are too small to be seen in the figures. The optimal final values of the states are $x^*(T = 50.0) = \{4.316, 20.09, 0.1566, 0.4986\}^T$ and the optimal final energy resulting from this case is -9.512×10^{-2} (which is not sufficient for escape). Using Eq. (34), the final values of the costates are then computed to be $\lambda^*(T = 50.0) = \{-5.367 \times 10^{-2}, 0.0, -0.1566, -0.4986\}^T$.

To perform the backward integration, the package ODE is used.[26] This package uses a modified divided difference form of the Adams predictor–corrector integration formulas and local extrapolation. It adjusts the order of the integrator and the stepsize to control the local error. We have used it extensively with good results.

The results of an investigation of the accuracy of the solution of this problem using each of the four system constraints are shown in Tables 1 and 2. These results are for the cases with subintervals of equal length with a total of 10, 20, 30, 40, 50, 60, 70, and 80 subintervals. Shown in Table 1 is the relative error in the values of the radius at the initial time that results from the backward integration of the system Euler–Lagrange equations. Similarly, Table 2 shows the relative error in the velocity obtained from backward integration. As can be seen, the error for each of the methods decreases as the stepsize Δt_i decreases with the higher order methods' errors decreasing more rapidly. With the fourth- and fifth-degree methods the errors begin increasing as the number of subintervals increases past 70 and 60, respectively. This behavior is a clear example of the effects of finite precision arithmetic, of which Conte and de Boor warned,[14] where an accumulation of rounding errors begin to dominate the

Table 1 Error in the initial position

Number of	System constraint			
subintervals	Trapezoid	Simpson's	Fourth degree	Fifth degree
10	2.529×10^{-1}	4.641×10^{-2}	5.533×10^{-3}	3.475×10^{-4}
20	2.112×10^{-1}	4.455×10^{-3}	1.136×10^{-4}	4.237×10^{-6}
30	6.415×10^{-2}	9.764×10^{-4}	1.222×10^{-5}	9.109×10^{-7}
40	2.823×10^{-2}	3.248×10^{-4}	6.283×10^{-6}	8.753×10^{-7}
50	1.595×10^{-2}	1.353×10^{-4}	2.748×10^{-6}	6.799×10^{-7}
60	1.036×10^{-2}	6.583×10^{-5}	1.884×10^{-6}	2.427×10^{-7}
70	7.323×10^{-3}	3.548×10^{-5}	1.118×10^{-6}	4.973×10^{-7}
80	5.478×10^{-3}	2.123×10^{-5}	1.154×10^{-6}	9.576×10^{-7}

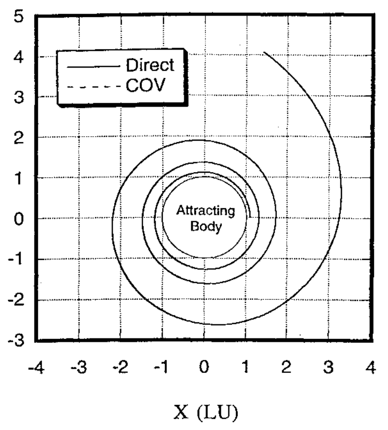

Fig. 5 Trajectory optimizing final orbit energy.

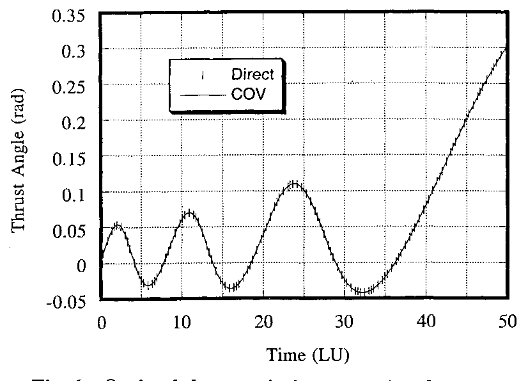

Fig. 6 Optimal thrust pointing angle time history.

Table 2 Error in the initial velocity

Number of subintervals	System constraint			
	Trapezoid	Simpson's	Fourth degree	Fifth degree
10	1.296×10^{-1}	3.501×10^{-2}	4.947×10^{-3}	2.576×10^{-4}
20	7.710×10^{-2}	3.804×10^{-3}	9.365×10^{-5}	4.057×10^{-6}
30	3.780×10^{-2}	8.011×10^{-4}	1.115×10^{-5}	3.698×10^{-7}
40	2.146×10^{-2}	2.660×10^{-4}	2.263×10^{-6}	3.376×10^{-7}
50	1.371×10^{-2}	1.104×10^{-4}	1.376×10^{-6}	2.580×10^{-7}
60	9.487×10^{-3}	5.411×10^{-5}	9.000×10^{-7}	1.896×10^{-7}
70	6.948×10^{-3}	2.919×10^{-5}	5.646×10^{-7}	4.766×10^{-7}
80	5.309×10^{-3}	1.730×10^{-5}	5.863×10^{-7}	9.433×10^{-7}

Table 3 Results from equal error investigation with minimum relative error of 10^{-2}

System constraint	Position error	Number of subintervals	Number of parameters	CPU time, s
Trapezoid	0.9977×10^{-2}	61	310	12.6
Simpson's	0.9230×10^{-2}	16	101	3.6
Fourth degree	0.8785×10^{-2}	9	104	3.6
Fifth degree	0.7004×10^{-2}	7	89	3.4

Table 4 Results from equal error investigtion with minimum relative error of 10^{-3}

System constraint	Position error	Number of subintervals	Number of parameters	CPU time, s
Trapezoid	0.9912×10^{-3}	183	920	283.2
Simpson's	0.9764×10^{-3}	30	185	6.2
Fourth degree	0.8929×10^{-3}	15	170	5.6
Fifth degree	0.8643×10^{-3}	9	113	3.9

Table 5 Results from equal error investigation with minimum relative error of 10^{-4}

System constraint	Position error	Number of subintervals	Number of parameters	CPU time, s
Trapezoid	*	*	*	*
Simpson's	0.9324×10^{-4}	55	335	19.1
Fourth degree	0.8668×10^{-4}	21	236	9.2
Fifth degree	0.7226×10^{-4}	13	161	5.5

Table 6 Results from equal error investigation with minimum relative error of 10^{-5}

System constraint	Position error	Number of subintervals	Number of parameters	CPU time, s
Trapezoid	*	*	*	*
Simpson's	0.9920×10^{-5}	98	593	45.9
Fourth degree	0.9687×10^{-5}	31	346	20.4
Fifth degree	0.9152×10^{-5}	18	221	9.3

solution accuracy. The minimum error obtained for the fifth-degree method is slightly greater than 10^{-7}. The constraint tolerance is slightly less than 10^{-7}. The accumulation of the rounding errors in the interpolating polynomial at the constraint points has affected the solution accuracy to the point that it begins to dominate. With the trapezoid and Simpson's methods the rounding error introduced by constraint tolerance will cause the minimum error achieved to be greater than that of the higher-order methods.

In a subsequent investigation, the number of subintervals was adjusted for each of the four system constraints until the magnitude of the error in the initial value of the radius vector just reaches a specified accuracy. The results are shown in Tables 3–6. In our experience, there exists a practical upper limit of approximately 1000 parameters in the size of the NLP problem that may be solved by NZOPT. Thus, for those cases in which the trapezoid rule failed to achieve the desired accuracy with 1000 or fewer parameters, an asterisk replaces the expected data. As expected, the higher-order integrator requires fewer subintervals to achieve a given accuracy. In addition, the fifth-degree system constraint requires the fewest number of parameters in each case to achieve the desired accuracy. For the cases shown in Table 3, each of the solution methods achieves approximately the same accuracy using relatively few parameters and little CPU time. The benefit of using a higher-degree system constraint is illustrated in Table 4 where the number of parameters and the CPU time required increase substantially for the trapezoid rule. In addition, the trapezoid rule could not achieve an accuracy of 10^{-4} as shown in Tables 5 and 6.

The Simpson's rule system constraint also uses significantly more NLP parameters and CPU time than are required for the constraints corresponding to higher-order quadrature rules, as shown in Table 6. In addition to the problem of solution accuracy, solving NLP problems using Simpson's system constraint with large numbers of parameters can become quite difficult.[15] Thus, the use of a higher-order system constraint may be needed to reduce the total number of parameters used so that a solution to the NLP problem may be found.

Other investigations using higher-order rules are being conducted. A direct collocation solution to the problem of optimal interception of Earth-orbit-crossing asteroids, using the fifth-degree Gauss–Lobatto system constraints, has been found.[27] In addition, a direct collocation method based on a seventh-degree Gauss–Lobatto quadrature rule has been used to find minimum time, very-low-thrust, three-dimensional, Earth–moon orbit transfers incorporating third-body effects where the total transfer time is in excess of 30 days.[28] In these problems, both of which are more substantial than the example problem presented in this paper, the methods have proven to be robust, exhibiting no problems with stability.

Conclusions

A method for solving the system differential equations of a discretized optimal control problem, using constraints corresponding to higher-order Gauss–Lobatto numerical integration rules, has been described and its application to a low-thrust orbit transfer problem given. The results of this investigation show that the higher-order integrators do provide accurate solutions with fewer total parameters than lower-order integration rules such as the trapezoid rule require. These results also show that as the desired accuracy of a solution increases, the number of parameters and execution time for a lower-order system constraint increase faster than those for a higher-order system constraint. Thus, higher-order system constraints are recommended for accurate solutions of large nonlinear programming problems resulting from the discretization of trajectory optimization problems. Examples of such large problems where these new constraints would be beneficial are many-revolution or long-duration transfers for vehicles using very low-thrust propulsion.

Acknowledgments

We wish to thank McDonnell Douglas Aerospace Co. for giving us the use of their NLP package NZOPT and the Pittsburgh Supercomputing Center for the use of their Cray C90.

References

[1]Enright, P. J., and Conway, B. A., "Optimal Finite-Thrust Spacecraft Trajectories Using Collocation and Nonlinear Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 14, No. 5, 1991, pp. 981–985.

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

[3]Scheel, W. A., and Conway, B. A., "Optimization of Very-Low-Thrust, Many-Revolution Spacecraft Trajectories," *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 6, 1994, pp. 1275–1282.

[4]Betts, J. T., "Using Sparse Nonlinear Programming to Compute Low Thrust Orbit Transfers," *Journal of the Astronautical Sciences*, Vol. 41, No. 3, 1993, pp. 349–371.

[5]Betts, J. T., "Optimal Interplanetary Orbit Transfers by Direct Transcription," *Journal of the Astronautical Sciences*, Vol. 42, No. 3, 1994, pp. 247–268.

[6]Pierson, B. L., and Kluever, C. A., "Three-Stage Approach to Optimal Low-Thrust Earth–moon Trajectories," *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 6, 1994, pp. 1185–1192.

[7]Tang, S., and Conway, B. A., "Optimization of Low-Thrust Interplanetary Trajectories Using Direct Collocation and Nonlinear Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 3, 1995, pp. 599–604.

[8]Anon., "Editorial Policy Statement on Numerical Accuracy and Experimental Uncertainty," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 1, 1995, p. 7.

[9]Bryson, A. E., and Ho, Y.-C., *Applied Optimal Control*, Hemisphere, New York, 1975, pp. 87–89.

[10]Kirk, D. E., *Optimal Control Theory: An Introduction*, Electrical Engineering Series, Prentice–Hall, Englewood Cliffs, NJ, 1970, Chap. 5.

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

[12]Dickmanns, E. D., and Well, K. H., "Approximate Solution of Optimal Control Problems Using Third-Order Hermite Polynomial Functions," *Proceedings of the 6th Technical Conference on Optimization Techniques*, International Federation for Information Processing—Technical Conf. 7, Springer–Verlag, New York, 1975.

[13]Russell, R. D., and Shampine, L. F., "A Collocation Method for Boundary Value Problems," *Numerical Mathematics*, Vol. 19, 1972, pp. 1–28.

[14]Conte, S. D., and de Boor, C., *Elementary Numerical Analysis: An Algorithmic Approach*, McGraw–Hill, New York, 1980, Chap. 8.

[15]Hammerlin, G., and Hoffmann, K.-H., *Numerical Mathematics*, translated by L. Schumaker, Springer–Verlag, New York, 1991, Chap. 7.

[16]Davis, P. J., and Rabinowitz, P., *Methods of Numerical Integration*, 2nd ed., Academic, New York, 1984.

[17]Hildebrand, F. B., *Introduction to Numerical Analysis*, 2nd ed., Dover, New York, 1974.

[18]Skeel, R. D., and Keiper, J. B., *Elementary Numerical Computing with Mathematica®*, McGraw–Hill, New York, 1993, Chaps. 7 and 8.

[19]Skeel, R. D., personal communication, March 1995.

[20]Enright, P. J., "Optimal Finite-Thrust Spacecraft Trajectories Using Direct Transcription and Nonlinear Programming," Ph.D. Thesis, Dept. of Aeronautical and Astronautical Engineering, Univ. of Illinois, Urbana–Champaign, IL, Jan. 1991.

[21]Link, B. D., "Numerical Solution of Stiff Ordinary Differential Equations Using Collocation Methods," M.S. Thesis, Dept. of Computer Science, Univ. of Illinois, Urbana–Champaign, IL, 1976.

[22]Kramarz, L., "Stability of Collocation Methods for the Numerical Solution of $y'' = f(x, y)$," Institutionen for Informationsbehandling, Lunds Universitet, Lund, Sweden, Bind 20, Hefte Nv. 2, 1980, pp. 215–222.

[23]Johnson, C., *Numerical Solution of Partial Differential Equations by the Finite Element Method*, Cambridge Univ. Press, New York, 1987, Chap. 9.

[24]Char, B. W., Geddes, K. O., Gonnet, G. H., Leong, B. L., Monagan, M. B., and Watt, S. M., *Maple V Language Reference Manual*, Springer–Verlag and Waterloo Maple, New York, 1991.

[25]Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H., *User's Guide for NZOPT 1.0: A Fortran Package for Nonlinear Programming*, McDonnell Douglas Aerospace, Huntington Beach, CA, April 1993.

[26]Shampine, L. F., and Gordon, M. K., *Computer Solutions of Ordinary Differential Equations*, W. H. Freeman, San Francisco, CA, 1975.

[27]Conway, B. A., "Optimal Low-Thrust Interception of Earth-Crossing Asteroids," AAS/AIAA Space Flight Mechanics Meeting, Austin, TX, AAS Paper 96-195, Feb. 1996.

[28]Herman, A. L., "Improved Collocation Methods Used for Direct Trajectory Optimization," Ph.D. Thesis, Dept. of Aeronautical and Astronautical Engineering, Univ. of Illinois, Urbana–Champaign, IL, Sept. 1995.