JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 23, No. 1, January–February 2000

Solving Optimal Control Problems Using *hp*-Version Finite Elements in Time

Michael S. Warner* and Dewey H. Hodges†
Georgia Institute of Technology, Atlanta, Georgia 30332-0150

A previously published temporal finite element method for optimal control problems is updated to include higher-order hierarchical shape functions. The accuracy and power of the analysis and corresponding code are illustrated on optimal control problems with nonlinear system dynamics, using interfaces based on symbolic mathematics to automatically generate the necessary derivative expressions and nonlinear algebraic equations to be solved. The class of problems includes those that can have free or fixed final time, multiple phases, nonlinear/periodic boundary conditions, and control constraints. This code has been tested on a variety of problems, culminating in a three-phase, seven-state, two-control missile problem with nonlinear system dynamics, where accuracy was found to be significantly improved over *h*-version results, with potential order-of-magnitude reductions in CPU time and numbers of parameters for a given level of error.

Introduction

IN Ref. 1 a new method for solving optimal control problems using finite elements in time was proposed by Bless and Hodges. In this formulation, the boundary conditions are all enforced weakly through use of Lagrange multipliers. In this way, the same set of shape functions can be used to approximate solutions to general problems because the shape functions do not have to meet any strong boundary conditions. Another feature of their work is that the simplest possible shape functions (piecewise constants) are used in each case, eliminating the need for numerical quadrature in the resulting equations. Other contrasts between this methodology and direct methods for solving optimal control problems are summarized in their paper.[1]

In their work,[1] a general FORTRAN code was developed that could approximate solutions to optimal control problems with constraints on the controls and states.[2–4] They were able to solve nominal trajectory optimization problems for a complex advanced launch vehicle model in remarkably little CPU time, on the order of a few seconds. Likewise, engineers at Lockheed Martin Vought Systems Corporation used a version of this code to compute optimal trajectory updates of a missile model in 1 s (Ref. 5), showing great potential for real-time applications.

The errors using this scheme were reasonable, reducing proportionally to the square of the element lengths. However, using the Bless and Hodges[1] methodology to increase the order of accuracy in solutions, the finite element mesh would have to be refined excessively at a prohibitive computational cost. Similarly, in their work, all mesh refinement was done by bisecting all elements in a given phase (one stage of a rocket or a period of being on a state constraint, for example).

In the current effort, two methods are proposed for reducing the CPU times and number of parameters necessary to achieve a given level of error in finite element solutions to optimal control problems, including those with control constraints. High-order shape functions are implemented in the finite element method in an effort to increase the order of accuracy of approximate solutions. Likewise, nonuniform finite element meshes and nonuniform mesh refinement are explored, providing a much larger set from which to choose an optimal mesh.

The set of optimal-control problems that is defined can be solved using the current implementation of high-order finite element shape functions, including problems in which the states or the state equations are discontinuous, a class of problems called multiphase problems. Then results are shown for a single-phase problem from Ref. 6 that has been solved using this method, followed by a discussion of how h-version adaptive solution refinement techniques apply to multiphase problems, specifically a seven-state, two-control, three-phase missile targeting problem with nonlinear system dynamics.

Optimal Control Problems

The systems being studied are governed by a set of n_x general, nonlinear state differential equations that are discontinuous at a single point in time t_1

$$\dot{x}(t) = \begin{cases} f_1(x, u, t), & t \in [t_0, t_1) \\ f_2(x, u, t), & t \in (t_1, t_f] \end{cases}$$
$$x \in R^{n_x}, \quad u \in R^{n_u}, \quad t \in [0, t_f] \qquad (1)$$

The function $f(\,)$ is assumed to be twice differentiable with respect to its arguments, except perhaps at t_1, and the states and the controls are assumed to be piecewise continuous, again perhaps except for at t_1. The final time t_f and t_1 can be specified or not, whereas the initial time is assumed to be zero. This formulation generalizes to the case of many intermediate times, though the equations will be developed only for the case with two phases.

For this situation, general, nonlinear boundary conditions on the states can be specified at the initial time, the final time, just before the intermediate time t_1^-, just after the intermediate time t_1^+, or some combination of those times,

$$\Psi\big[x(0), x(t_1^-), x(t_1^+), t_1, x(t_f), t_f\big] = 0, \qquad \Psi \in R^{n_{\rm bc}} \qquad (2)$$

thus allowing for periodic boundary conditions, for example.

Let J be a cost functional to be minimized for this problem in terms of a scalar penalty on the states at the endpoints of the time interval (or either side of the intermediate time) and/or an integral penalty on the states, controls, and time, which can also be distinct in the two phases,

$$J = \phi\big[x_0, x(t_1^-), x(t_1^+), t_1, x_f, t_f\big] + \int_0^{t_1^-} L_1(x, u, t)\,\mathrm{d}t + \int_{t_1^+}^{t_f} L_2(x, u, t)\,\mathrm{d}t \qquad (3)$$

where $x_0 \equiv x(t_0)$ and $x_f \equiv x(t_f)$.

The optimal control problem is to find the control time history $u(t)$ that causes the system governed by Eq. (1) to meet the boundary conditions (2) such that the given cost functional (3) is minimized.

Received 15 September 1997; revision received 3 May 1999; accepted for publication 9 June 1999. Copyright © 1999 by Michael S. Warner and Dewey H. Hodges. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

*Graduate Research Assistant, School of Aerospace Engineering; currently Space Systems Engineer, NAIC/DXSS, First Lieutenant, U.S. Air Force, Wright–Patterson Air Force Base, Ohio. Member AIAA.

†Professor, School of Aerospace Engineering. Fellow AIAA.

Admissible control histories are assumed to be bounded and continuous within each phase. Further restrictions on the control history are also allowed in the form of inequality constraints, which can be different in the two phases,

$$G(x,u) = \begin{cases} g_1(x,u) \le 0, & g_1 \in R^{n_{g1}}, \quad t \in [0, t_1) \\ g_2(x,u) \le 0, & g_2 \in R^{n_{g2}}, \quad t \in (t_1, t_f] \end{cases} \tag{4}$$

including the case where either n_{g_1} or n_{g_2} is zero. No more than n_u of these can be equal to zero at any one time in either phase. The functions $g_i(\,)$ are assumed to be at least a function of the controls, though they may also be a function of the states. Otherwise, $g_i(\,)$ is considered a state constraint that can be handled in optimal control problems[6] and in particular in the finite element methodology,[4] but this is a complex subclass of optimal control problems that has not been studied in this work. Control constraints are dealt with in more detail in Ref. 7.

The constraints in Eq. (4) are enforced through use of slack variables K, such that Eq. (4) is replaced by the equality constraints

$$(g_j)_i + (K_j)_i = 0, \qquad i \in [1, n_{g_j}], \quad j \in [1, 2] \tag{5}$$

in the jth phase, where K_j is a column matrix of squared variables

$$(K_j)_i = \left(k_j^2\right)_i, \qquad i \in \left[1, n_{g_j}\right], \quad j \in [1, 2] \tag{6}$$

Calculus of Variations

In using the calculus of variations to minimize a cost functional subject to constraints,[6,8] the residuals of the differential equations, control constraints, and boundary conditions are adjoined to the original cost function by means of Lagrange multipliers $\lambda(t)$, $\mu(t)$, and ν, respectively. This yields a new cost function J' such that

$$J' = \Phi\left[x_0, x(t_1^-), x(t_1^+), t_1, x_1, t_f\right] + \int_0^{t_1^-} (H_1 - \lambda^T \dot{x})\,\mathrm{d}t + \int_{t_1^+}^{t_f} (H_2 - \lambda^T \dot{x})\,\mathrm{d}t \tag{7}$$

where the Hamiltonian (distinct within each phase) and Φ are defined as

$$H_1 \equiv L_1 + \lambda^T f_1 + \mu^T (g_1 + K_1)$$

$$H_2 \equiv L_2 + \lambda^T f_2 + \mu^T (g_2 + K_2), \qquad \Phi \equiv \phi + \nu^T \Psi$$

To find the local minimum of J', independent variations in the states, state rates, controls, Lagrange multipliers, slack variables, intermediate time, and final time are taken. Ignoring the boundary conditions as before, the expression for the variation can be found in Ref. 3 as

$$\delta J' = \int_0^{t_1^-} \left\{(H_1)_u \delta u + \left[(H_1)_x + \dot{\lambda}^T\right]\delta x + \delta\lambda^T (f_1 - \dot{x}) + \delta\mu^T (g_1 + K) + \mu^T \delta K\right\}\mathrm{d}t + \int_{t_1^+}^{t_f} \left\{(H_2)_u \delta u + \left[(H_2)_x + \dot{\lambda}^T\right]\delta x + \delta\lambda^T (f_2 - \dot{x}) + \delta\mu^T (g_2 + K) + \mu^T \delta K\right\}\mathrm{d}t \tag{8}$$

When the corresponding variations in the cost function are each set equal to zero, the resulting equations are the standard ones outlined in Ref. 6, to include internal and endpoint boundary conditions on the costates and Hamiltonian.

Discretization of the Problem

Unfortunately, direct solution of those equations can only be done in relatively trivial problems, and as such the solutions to complex problems must be approximated. Using Eq. (8), to solve $\delta J' = 0$, the infinite-dimensional problem can be simplified by approximating the true unknown solution (states, controls, etc.) as an element of a finite set of piecewise polynomials. The variations are then also approximated using piecewise polynomials of appropriate order to generate a set of nonlinear algebraic equations (with the polynomial coefficients as unknowns) that can be solved much more readily than the two-point boundary value problem indicated in Eq. (8). With the problem cast in the weakest form,[1] the polynomials do not have to meet any boundary conditions, and the same set of polynomials can be used to solve a wide class of optimal control problems.

Previous work[1,3] used discontinuous, piecewise-constant polynomials for the main variables. This choice is prudent for problems that have discontinuities in the states or costates, which many optimal-control problems do; therefore, in this effort, discontinuous shape functions will be used.

Another advantage is that in using discontinuous polynomials, the simplest version of those (piecewise constants) allows the integrals in Eq. (8) to be done by inspection rather than by numerical quadrature. Even the simplest continuous polynomial approximations will require numerical quadrature over an element.

However, because Eq. (8) has the $\dot{x}$ and $\dot{\lambda}$ terms under the integral, using discontinuous shape functions results in jump terms at each node point when the time interval is discretized. To avoid these jump terms while maintaining discontinuous shape functions, the expressions $\dot{\lambda}^T \delta x$ and $\delta\lambda^T \dot{x}$ are integrated by parts, and continuous approximations for δx and $\delta\lambda$ are used.

Likewise, for simplicity, a single phase is assumed, with the equations being generalized after the development. Equation (8) then becomes

$$\delta J' = \lambda^T \delta x\big|_0^{t_f} - \delta\lambda^T x\big|_0^{t_f} + \int_0^{t_f} \left[H_u \delta u + H_x \delta x - \lambda^T \delta\dot{x} + \delta\lambda^T f + \delta\dot{\lambda}^T x + \delta\mu^T (g + K) + \mu^T \delta K\right]\mathrm{d}t \tag{9}$$

The integral is then broken up into N nonoverlapping, not necessarily equally spaced elements, with the time within the ith element, t_i, expressed in terms of a nondimensional variable τ as

$$t_i(\tau) = \hat{t}_{i-1} + \tau \Delta t_i, \qquad 0 \le \tau \le 1 \tag{10}$$

Solving for τ gives

$$\tau = (t_i - \hat{t}_{i-1})/\Delta t_i \tag{11}$$

such that $\mathrm{d}t = \Delta t_i\,\mathrm{d}\tau$, and τ is the normalized time in each element.

Substituting for the nondimensional time parameter yields

$$\delta J' = \lambda^T \delta x\big|_0^{t_f} - x^T \delta\lambda\big|_0^{t_f} + \sum_{i=1}^{N} \Delta t_i \int_0^1 \left[(H_x)_i \delta x_i + \frac{\delta\lambda_i'^T}{\Delta t_i} x_i - \frac{\delta x_i'^T}{\Delta t_i}\lambda_i + \delta\lambda_i^T f_i + (H_u)_i \delta u_i + \delta\mu_i^T (g_i + K_i) + \mu_i^T \delta K_i\right]\mathrm{d}\tau \tag{12}$$

In this equation, a subscript i on a variable refers to the value of that variable within the ith element, whereas the subscript on a function indicates the value of that function evaluated using variables within the ith element.

Shape functions for the variational quantities for the states[9] are chosen from the space C^0, the space of continuous functions with no derivatives constrained to be continuous. These shape functions are in terms of nodal values ($\hat{\ }$) and polynomial coefficients on the element interiors ($\bar{\ }$)

$$\delta x_i(\tau) = \delta\hat{x}_i(1-\tau) + \delta\hat{x}_{i+1}\tau + \sum_{j=1}^{p-1} (1-\tau)\tau\beta_j(\tau)\delta\bar{x}_{ij} \tag{13}$$

Here $(p-1)$ is the order of the shape function polynomial being used, with the summation being ignored if $p = 1$. The functions $\beta_j(\tau)$ are the hierarchical set of Jacobi polynomials of order $(j-1)$, as given in detail in Ref. 10. The costate expressions are found by replacing x with λ throughout Eq. (13). This choice of shape functions provides well-conditioned coefficient matrices, as well as approximate solutions to linear dynamics problems that are both unconditionally stable and accurate.

The corresponding shape functions for the nonvariational quantities are only continuous within the element but discontinuous elsewhere, and they are represented as

$$x_i(\tau) = \begin{cases} \hat{x}_i, & \tau = 0 \\ \sum_{j=1}^{p} \alpha_j(\tau)\bar{x}_{ij}, & 0 < \tau < 1 \\ \hat{x}_{i+1}, & \tau = 1 \end{cases} \tag{14}$$

where the functions $\alpha_j(\tau)$ are a hierarchical set of polynomials of order $(j-1)$ as given in Ref. 9. Again, the λ_i expressions are of the same form, as are those for u_i, μ_i, and K_i, with $\hat{x}_i$ (and corresponding nodal values for the other variables) being discrete values that are distinct from the values of the corresponding function within the element. Because the nodal values do not appear in the main finite element equations for the element interiors (as will be seen), they provide the anchor between the element interior values and the boundary values of the states and costates that are defined by the nodal values. As discussed later, these nodal values are the reported solution, not the interior values. The enforcement of both interior and exterior boundary conditions through the nodal values that are weakly linked to the interior values results in solutions that automatically tend to become more nearly continuous in the appropriate places as either h or p refinements are undertaken.

Substituting the shape functions from Eqs. (14) into the integral that remains in the cost function (12), enforcing the orthogonality of polynomials $\alpha(\tau)$ and $\epsilon(\tau) \equiv (1-\tau)\tau\beta_j(\tau)$, and arranging terms by their variational coefficient give

$$\begin{aligned}
\delta J' &= \delta\hat{\lambda}_1^T\left[-\hat{x}_1 + \bar{x}_{1,1} - \Delta t_1\int_0^1 (1-\tau)f_1\,\mathrm{d}\tau\right] \\
&+ \sum_{i=2}^{N}\delta\hat{\lambda}_i^T\left[-\bar{x}_{i-1,1} - \Delta t_{i-1}\int_0^1 \tau f_{i-1}\,\mathrm{d}\tau\right. \\
&\left.+ \bar{x}_{i,1} - \Delta t_i\int_0^1 (1-\tau)f_i\,\mathrm{d}\tau\right] \\
&+ \sum_{i=1}^{N}\sum_{j=1}^{p-1}\delta\bar{\lambda}_{ij}^T\left[-\bar{x}_{i,j+1} + \Delta t_i\int_0^1 \epsilon_j(\tau)f_i\,\mathrm{d}\tau\right] \\
&+ \delta\hat{\lambda}_{N+1}^T\left[-\bar{x}_{N,1} - \Delta t_N\int_0^1 \tau f_N\,\mathrm{d}\tau + \hat{x}_{N+1}\right] \\
&+ \delta\hat{x}_1^T\left[\hat{\lambda}_1 - \bar{\lambda}_{1,1} - \Delta t_i\int_0^1 (1-\tau)(H_x)_1\,\mathrm{d}\tau\right] \\
&+ \sum_{i=2}^{N}\delta\hat{x}_i^T\left[-\bar{\lambda}_{i,1} - \Delta t_i\int_0^1 (1-\tau)(H_x)_i\,\mathrm{d}\tau\right. \\
&\left.+ \bar{\lambda}_{i-1,1} - \Delta t_{i-1}\int_0^1 \tau(H_x)_{i-1}\,\mathrm{d}\tau\right] \\
&+ \sum_{i=1}^{N}\sum_{j=1}^{p-1}\delta\bar{x}_{ij}^T\left[\bar{\lambda}_{i,j+1} + \Delta t_i\int_0^1 \epsilon_j(\tau)(H_x)_i\,\mathrm{d}\tau\right] \\
&+ \delta\hat{x}_{N+1}^T\left[-\hat{\lambda}_{N+1} + \bar{\lambda}_{N,1} - \Delta t_N\int_0^1 \tau(H_x)_N\,\mathrm{d}\tau\right] \\
&+ \sum_{i=1}^{N}\sum_{j=1}^{p}\delta\bar{u}_{ij}\left[\Delta t_i\int_0^1 (H_u)_i\alpha_j(\tau)\,\mathrm{d}\tau\right] \\
&+ \sum_{i=1}^{N}\sum_{j=1}^{p}\delta\bar{\mu}_{ij}\left[\Delta t_i\int_0^1 \alpha_j(\tau)\left\{g_i + \left(\sum_{j=1}^{p}\alpha_j(\tau)\bar{k}_{ij}\right)^2\right\}\mathrm{d}\tau\right] \\
&+ \sum_{i=1}^{N}\sum_{j=1}^{p}\delta\bar{k}_{ij}\left[\Delta t_i\int_0^1 2\alpha_j(\tau)\left(\sum_{j=1}^{p}\alpha_j(\tau)\bar{\mu}_{ij}\right)^T\right. \\
&\left.\times\left(\sum_{j=1}^{p}\alpha_j(\tau)\bar{k}_{ij}\right)\mathrm{d}\tau\right]
\end{aligned} \tag{15}$$

The variational coefficients are independent and arbitrary due to the weak form of the equations. Therefore, for the variation of J' to be zero, the expressions multiplied by the coefficients must be identically zero. That is, if all of the expressions in brackets in the Eq. (15) are zero, the first-order necessary conditions for optimal control are approximated. To supplement the equations derived from setting Eq. (15) equal to zero, the following boundary conditions need to be enforced[6]

$$\lambda_1^T + \frac{\partial\Phi}{\partial x_0} = 0, \qquad \lambda_{N+1}^T - \frac{\partial\Phi}{\partial x_f} = 0$$
$$H(t_f) + \frac{\partial\Phi}{\partial t_f} = 0, \qquad \Psi = 0 \tag{16}$$

Note that the only nodal values of the states and costates that appear in this formulation are those at the endpoints of the time interval. The nodal values for the states and costates on the interior of the phase can be generated from a relationship similar to the equation for the value of the final node, that is,

$$\hat{x}_{i+1} = \bar{x}_{i,1} + \Delta t_i\int_0^1 \tau f_i\,\mathrm{d}\tau \tag{17}$$

using the other nodal and interior values that have been solved for already. Once this is done for the states and costates, the interior nodal values for the controls, slack variables, and control constraint Lagrange multipliers are calculated as a local problem by using Newton's method on the optimality condition on the controls ($H_u = 0$) along with the control constraints. These values are then reported as the approximate solution.

For problems with $M > 1$ phases, the expressions in brackets in Eq. (15) simply hold over each phase, taking care that the appropriate system equations, constraints, and cost functions are used for each phase. The boundary conditions are then supplemented with equations for the discontinuities in the Hamiltonian and costates at the ith boundary ($i \in [1, M-1]$) (Ref. 6):

$$\lambda^T(t_i^-) = \frac{\partial\Phi}{\partial x(t_i^-)}, \qquad \lambda^T(t_i^+) = -\frac{\partial\Phi}{\partial x(t_i^+)}$$
$$H_{i+1}(t_i^+) = H_i(t_i^-) + \frac{\partial\Phi}{\partial t_i} \tag{18}$$

Thus, the two-point boundary value problem is approximated by the set of nonlinear algebraic equations derived from Eqs. (15) and (16). To solve an eight-state, two-control problem, such as the one described in a later section, using third-order shape functions and 64 elements, this results in over 3000 equations.

Implementation and Results

The FORTRAN code, called GENCODE, that was originally developed by Bless and Hodges (as described in Ref. 2) to solve these nonlinear equations has been expanded and revised. In essence, GENCODE is a restricted-step Newton–Raphson code, specialized for the case of the equations derived from Eqs. (15) and (16). Most of the code consists of setting up the Jacobian matrix in a certain structure, depending on the order of shape functions used and the finite element mesh.

GENCODE previously was written to solve these equations when the order of shape functions was zero uniformly across all elements. Now, GENCODE can handle shape functions of arbitrary order (within memory limitations of the computer) that vary arbitrarily among the elements, though all variables in an element are represented by polynomials of the same order.

Similarly, in the previous version of GENCODE, the elements in each phase were of equal length Δt, and thus, mesh refinement consisted of a uniform bisection of each element. The current version allows for an arbitrary distribution of elements, assuming, of course, that each point in the time interval is either a part of exactly one element or on an element boundary.

A restricted-step Newton–Raphson iteration differs from the standard iteration in that if the full Newton step yields a value of the target function that is an insufficient improvement over that of the starting point, perhaps even yielding a higher value, the step size is halved, and the cost is evaluated at the new point. This process is repeated until a sufficiently improved point is found with the step size then being reset to unity.

The determination of the step direction requires the solution of the linear system of equations. For optimal control problems, each equation that is not a boundary condition depends at most on values in two neighboring elements, so that the Jacobian ends up being quite sparsely populated. The sparsity is exploited by the use of sparse linear systems solvers from the Harwell subroutine library.[11]

The Jacobian matrix for optimal control problems involves partial derivatives of the system equations, boundary conditions, and control constraints with respect to all relevant states and controls. Likewise, the Jacobian also requires expressions for the $\alpha(\tau)$ and $\epsilon(\tau)$ polynomials from Eq. (15) that come from a recursion formula involving derivatives and integrals of polynomials, as developed in Ref. 9. The symbolic mathematics computer package MACSYMA[12] was used to generate analytical expressions for all of these quantities. GENCODE then assembles all of the analytical expressions to calculate the Jacobian matrix and the residual column matrix for Newton's method. Note that these same routines are also available in MAPLE[13] format. This leaves the user to supply the initial guess for the Newton iteration, which includes all of the polynomial coefficients for the shape functions plus the values at nodes for all variables. Depending on the problem, supplying the initial guess can be anywhere from a trivial to a complex task, and no new insights into this selection process have been gained in this work.

The integrals in Eq. (15) are approximated using Gauss–Legendre integration,[14] with the user selecting the number of Gauss points, which at this time is constant for all of the integrations.

Single-Phase Problem

In perhaps the simplest representation of an aerospace problem with nonlinear system dynamics, this first problem involves the maximum velocity transfer to a particle of mass m to a specified horizontal flight path in a fixed time (see Ref. 6, p. 59). The mass is acted on by a force of constant magnitude ma and variable heading $\beta(t)$. The particle begins at the origin ($x = y = 0$) with zero velocity ($u = v = 0$), and after a certain length of time t_f, the particle will be in horizontal flight at a given altitude.

The final horizontal position $x(t_f)$ is unspecified, and the final horizontal velocity $u(t_f)$ is to be maximized. The cost function is then

$$J = u(t_f) \tag{19}$$

with the boundary conditions being supplemented by two conditions from the costates: $\lambda_x(t_f) = 0$ due to $x(t_f)$ being free, whereas $\lambda_u(t_f) = 1$ because $u(t_f)$ is being maximized.

Reference 6 gives the analytic solution in terms of the initial force heading angle, the final time, and the final altitude in unspecified units. These values were chosen to be 75 deg, unity, and unity, respectively.

This problem was set up in GENCODE by the user inputting the differential equations, constraints, and cost; then MACSYMA generated the necessary FORTRAN expressions. The FORTRAN code was then compiled and run, using the initial guess supplied by the user for each shape function coefficient for each variable. It was run for a variety of combinations of finite element parameters.

For any significant refinement of the mesh or increase in the shape function order, the plots of the approximations all lie atop one another and the exact solution, and so error plots are used to show the accuracy of each approximate solution. The L^2 norm of the error on the interval $[0, t_f]$ was chosen inasmuch as it takes into account errors through the entire time interval.

Because the exact solutions for this problem are available, Fig. 1 shows the L^2 norm of the error time histories relative to the exact solution for the states and costates combined as a function of the CPU time (on an Hewlett Packard-UX workstation) involved in calculating solutions. Each line represents a different order of approximation polynomial, and the data points start with a four-element solution and move by powers of two. The initial guess for each case was interpolated from the converged solution for the case of half as many elements (or for one fewer shape function coefficient if only two elements). The CPU times are cumulative along each line, with the first data point representing the time necessary to converge from the lower-order solution. Figure 2 shows the same error data vs the number of free parameters in the problem, which equals the dimension of the Jacobian. Note that the zero-order curves are equivalent to the work of Hodges and Bless.[1]

Once an initial solution was obtained for low-order shape functions and a small number of elements, the code converged easily as these parameters were changed. Not surprisingly, the overall errors reduced as the order of shape functions increased and as the number of elements increased. The higher-order shape functions

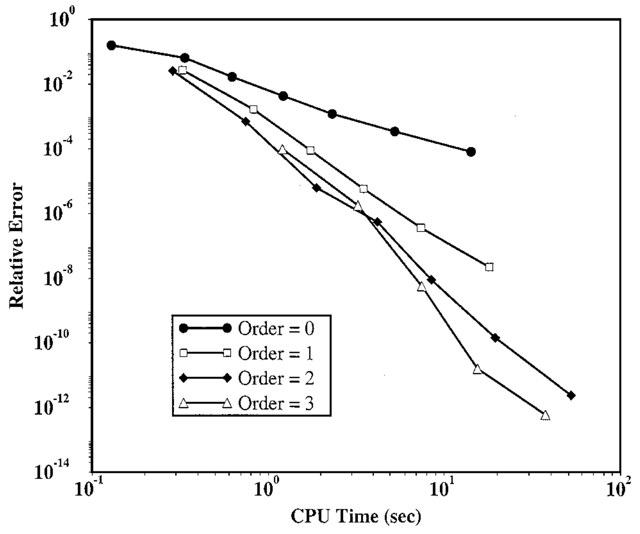

Fig. 1 Relative error in states and costates vs CPU time.

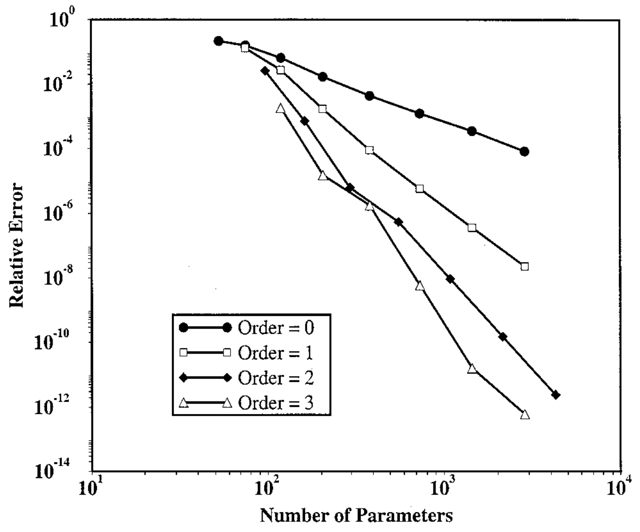

Fig. 2 Relative error in states and costates vs number of parameters.

also performed better using the CPU time as a criterion. Likewise, depending on how much CPU time was available, different orders of shape functions proved to provide the most accuracy: the second order for CPU times less than 3 s and the third order for times higher than that.

However, not all means have been exhausted to improve the h-version solutions to these problems. Beyond simply uniformly doubling the meshes and increasing the shape function orders, more prudent selection of the element mesh and shape function orders could bring down the computational effort while still reducing and smoothing errors. Starting with a coarse discretization, the error could be estimated in each element, and a new distribution of elements and higher-order shape functions could be determined to smooth and reduce this error estimator, ideally smoothing and reducing the error itself. Some indicators available to gauge error for use in such a process are examined in Ref. 15.

Multiphase Problem

The example multiphase problem to be used is a missile intercept problem maximizing impact velocity, using a generic missile model developed by Lockheed Martin Vought Systems Corporation (LMVS), then the LTV Aerospace and Defense Company. The equations of motion for this seven-state model were developed by Hodges and Johnson[16] and are summarized here. Note that three frames of reference are used: an inertial frame in which to represent the gravity and position states, a wind frame to conveniently determine the magnitudes of lift and drag, and a body frame to represent the orientation of the thrust vector and aerodynamic forces.

Let the orientation of the missile be defined by Rodrigues parameters,[17] expressed as a column matrix θ, so that the direction cosine matrix between the wind frame and the inertial frame can be expressed as

$$C = \frac{\left(1 - \frac{1}{4}\theta^T\theta\right)I + \frac{1}{2}\theta\theta^T - \tilde{\theta}}{1 + \frac{1}{4}\theta^T\theta} \tag{20}$$

with

$$\theta = \begin{Bmatrix}\theta_1\\ \theta_2\\ \theta_3\end{Bmatrix}, \qquad \tilde{\theta} \equiv \begin{bmatrix} 0 & -\theta_3 & \theta_2\\ \theta_3 & 0 & -\theta_1\\ -\theta_2 & \theta_1 & 0\end{bmatrix} \tag{21}$$

Rodrigues parameters are chosen because any possible singularities in the orientation angles between the wind frame and the inertial frame are moved to plus or minus 180 deg, which in this case means the problem will be free of singularities. It is noted that there is an excess of configuration variables used here, and an additional constraint is needed. This extra constraint is chosen to be a nonholonomic constraint on the angular velocity vector, namely, that the inertial wind-frame angular velocity component along the wind direction is equal to zero. This has the effect of yielding simpler equations, as can easily be verified.

The kinematic equations can then be derived from Newton's second law in terms of three Cartesian coordinates in the inertial frame $x = \{x_1, x_2, x_3\}^T$ plus the magnitude of the velocity vector V

$$\dot{x} = VC^Te_1 \tag{22}$$

$$\dot{V} = (1/m)e_1^TF_W \tag{23}$$

Note that x_1 is defined positive north, x_2 is positive east, $-x_3$ is the altitude, m is the mass, and $e_1 = \{1, 0, 0\}^T$.

From the angular velocities comes an expression for derivatives of the Rodrigues parameters,

$$\dot{\theta} = (1/mV)\left(I + \tfrac{1}{2}\tilde{\theta} + \tfrac{1}{4}\theta\theta^T\right)\tilde{e}_1F_W \tag{24}$$

as outlined in Ref. 16, with $\tilde{e}_1$ defined similarly as $\tilde{\theta}$ earlier.

In the preceding expressions, F_W is a column matrix containing the measure numbers of the resultant force on the missile, expressed in the wind frame basis, due to the lift L, the drag D, the thrust T, and the weight mg (assuming a flat earth as the inertial frame). Thrust and drag are assumed to be along the axis of the missile, which are related to the wind axis by means of an angle of attack α and a bank angle ϕ. Meanwhile the weight is in the e_3 direction, with the mass of the missile as fuel is burned being given as a function of time in tabular form by LMVS.

The lift and drag were assumed to have the standard forms of

$$L = \tfrac{1}{2}C_L\rho V^2S, \qquad D = \tfrac{1}{2}C_D\rho V^2S \tag{25}$$

where the atmospheric density ρ is derived from an exponential atmospheric model, S is a given reference area, whereas the lift and drag coefficients, C_L and C_D, are given in tabular form for various Mach numbers and angles of attack. The thrust T is given as

$$T = T_0 - A_ep_a \tag{26}$$

where A_e is nozzle exit area, p_a is the ambient pressure, and T_0 is the thrust produced by the missile, given by LMVS as constants for each of three flight regimes.

Here α is always considered positive, which allows ϕ to be between $-\pi$ and $+\pi$. However, due to symmetry in the missile, specifying ϕ between $-\pi/2$ and $+\pi/2$ adequately describes all possible aerodynamic states of the missile. Unfortunately, this formulation leads to singularities when α approaches 0. Therefore, a different control formulation is used in this work.

In Ref. 16, a control formulation is postulated based on Lagrangian equinoctial variables[18] to avoid singularities when $\alpha = 0$. In this formulation, α and ϕ are replaced by β_2 and β_3, which are defined as

$$\beta_2 = \cos\phi\tan\alpha, \qquad \beta_3 = \sin\phi\tan\alpha \tag{27}$$

with the auxiliary variable β defined by

$$\beta^2 = \beta_2^2 + \beta_3^2 = \tan^2\alpha \tag{28}$$

always taking the positive square root.

With this transformation, when α should be zero, both β_2 and β_3 are zero, and indeed they are continuous as α approaches zero. Back calculating ϕ inevitably results in discontinuities as inverse sines, cosines, and tangents are taken, especially when β_2 and β_3 move through zero. In particular, the bank angles shift between $\pi/2$ and $-\pi/2$. However, by missile symmetry, these orientations are equivalent, and the discontinuity does not affect Newton's method because the direction cosines of the body frame in the wind frame vary smoothly.

In the resulting transformed problem, the lift and drag coefficients, C_L and C_D, are replaced by even functions of β, C_l, and C_d, such that

$$C_l = \frac{C_L}{\beta\sqrt{1+\beta^2}}, \qquad C_d = \frac{C_D}{\sqrt{1+\beta^2}} \tag{29}$$

These new quantities are then splined in the same way the old aerodynamic data were, only this time in terms of β and the Mach number.

Note that in Newton's method, derivatives with respect to β_2 and β_3 will result in singularities near $\alpha = \beta = 0$, and so special handling of these derivatives has been done as outlined in Ref. 16.

Thus, the missile problem has seven states (three position states, three orientation states, and the magnitude of the velocity) plus two controls (β_2 and β_3). The problem is to find the control time histories for the missile that maximize the magnitude of the velocity at the final time, $V(t_f)$ [thus, $J = -V(t_f)$], while meeting both the initial conditions and terminal conditions based on those specified by LMVS, which are as follows:

$$\begin{aligned}
x_1(t_0) &= 0.0\text{ m} & x_1(t_f) &= 15{,}000\text{ m}\\
x_2(t_0) &= 0.0\text{ m} & x_2(t_f) &= 15{,}000\text{ m}\\
x_3(t_0) &= -3.0\text{ m} & x_3(t_f) &= -12{,}000\text{ m}\\
\theta_1(t_0) &= 0.0 & \theta_1(t_f) &= 0\\
\theta_2(t_0) &= 0.68865 & \theta_2(t_f) &= 2(\sqrt{2}-1)\\
\theta_3(t_0) &= 0.0 & \theta_3(t_f) &= 0\\
V(t_0) &= 19.6673\text{ m/s}
\end{aligned} \tag{30}$$

where the final time t_f is 25 s.

The equations are singular when the velocity is zero, and so it was assumed that the missile was being fired out of a tube, and then the exit point from the tube was used as the initial condition. Note that x_3 is the negative of the altitude, so that the problem begins with the missile 3 m above the ground. The initial azimuth angle Ψ is zero, indicating that the launch tube is pointed north and the initial flight-path angle γ is 38 deg. The final constraints on the Rodrigues parameters are equivalent to specifying that the missile again must be pointed north, pitched up 45 deg from horizontal. Though the missile is pointed north at the beginning and the end of the time interval, because the final value for x_2 is nonzero, the missile will not remain in a single plane.

Along with these endpoint boundary conditions, interior boundary conditions are provided because this is a multiphase problem with the engine thrust specified as separate constants over three given time intervals. In this case, these conditions are merely continuity equations across the time nodes at the two intermediate times, which are 7 and 15 s, respectively.

Generating initial guesses for this problem was much more complicated than for the previous problems where a guess of unity for all variables would be sufficiently close for Newton's method to converge to the actual solution. Between using largely out-of-scale variables plus nonanalytic expressions for the mass, atmospheric parameters, and aerodynamic data, much more accurate initial guesses were needed.

Appropriate guesses were generated by marching from the simplified missile model discussed earlier to the final missile model, one step at a time. Larger thrust was added, then higher mass, then more distant target point, and so on, until the full missile model was being used. At each stage, the initial guess from the previous model was used, and scaling coefficients were used to ease into each new model like an ad hoc homotopy method. Obviously, the discretization method cannot save one from the underlying shortcomings of the nonlinear algebraic equation solver, for example, a Newton–Raphson method. (Note, however, that the discretization method does provide a very sparse system that is quite economical to solve and for which initial guesses are generally easy to obtain. Moreover, the solutions are sufficiently accurate in their own right and certainly useful to be used as initial guesses for exact methods.)

Ultimately a solution to the complete three-phase problem was achieved for the h-version finite elements. This nominal solution had 23 total elements, 9 in the first phase, 2 in the second, and 12 in the third, denoted as the 9:2:12 mesh. The mesh is uniform in the second and third phases, and the nine elements in the first phase resulted from repeatedly bisecting the first element of an original four. Numerical problems with the lift and drag created this need for smaller elements in the first phase. This problem is further discussed later in this section.

With a nominal solution in place, the code then easily converged the first time for a wide variety of element distributions (both with a finer and coarser mesh) and increases in the order of shape functions. The approximate solution using piecewise linear shape functions over 81 elements (27 per phase, uniformly distributed) are given in Figs. 3–7. This solution involved over 117,000 nonzero Jacobian elements, essentially the maximum number of variables the code could handle on the Hewlett Packard-9000 workstation before running out of memory. This solution is indistinguishable on a graph from the one using piecewise-quadratic or higher-order shape functions, and it was chosen to maximize the number of data points along the approximate solution, providing the smoothest plots.

Figure 3 shows the position states, all of which vary smoothly. Next is the velocity in Fig. 4, which is smooth in each phase but changes slope at each phase boundary because the thrust is throttled back. Figure 5 highlights the Rodrigues parameters. Note the steep climb in each of the parameters within the first second. Similarly, Figs. 6 and 7 highlight the rapid changes in the control variables (both the Lagrangian equinoctial variables and the corresponding orientation angles) over the first second. These changes necessitated extra elements to accurately capture this behavior. Because of space limitations, plots of the costates are not shown. However, they are

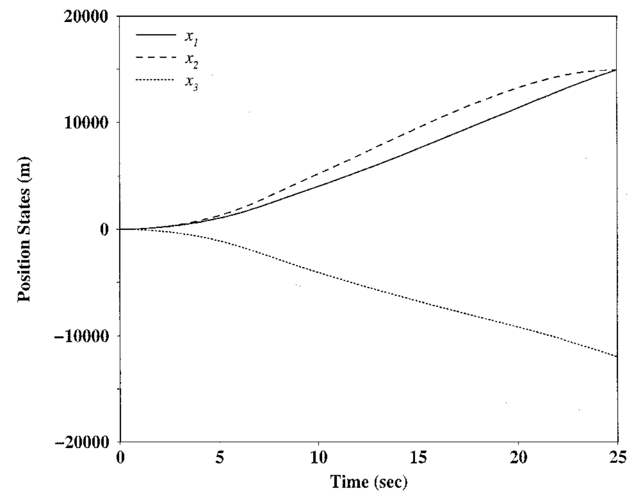

Fig. 3 Position states for missile problem.

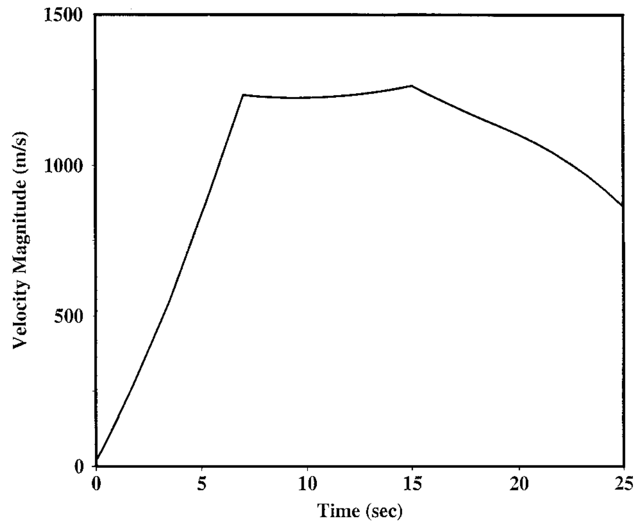

Fig. 4 Velocity for missile problem.

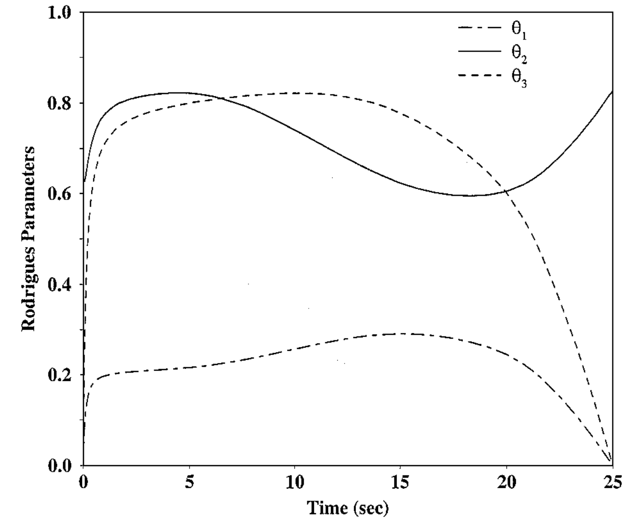

Fig. 5 Rodrigues parameters for missile problem.

calculated with the same degree of accuracy as all other variables in the current methodology.

The accuracy of the solutions in the first couple of elements is limited by the data given by LMVS for the lift and drag coefficients. Subsonic data were only given for a Mach number of 0.2. Likewise, no data points were given above 40-deg angle of attack. In Figs. 3–7, this only affects the first 0.05 s, but the effects become more far reaching when the initial element is longer.

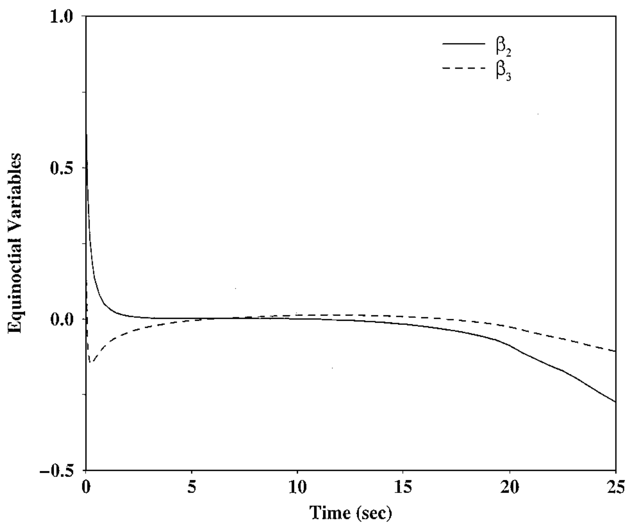

Fig. 6 Equinoctial variables for missile problem.

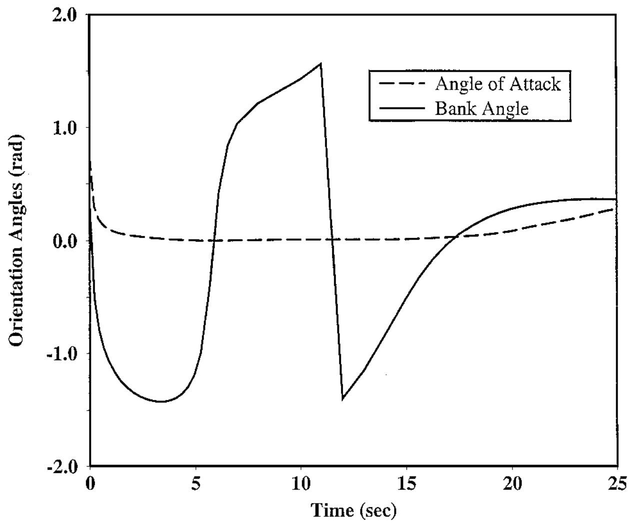

Fig. 7 Orientation angles for missile problem.

Error Analysis

Just by looking at Figs. 3–7, very little difference could be discerned between solutions for different orders of shape functions, except inasmuch as the increase in order of shape functions limits the number of elements possible with a finite amount of computer memory.

In attempting to analyze the error, two main problems arose. First, error plots are difficult to produce because the exact solution is not known for this problem. Likewise, with so many states and a limited amount of computer memory, reference solutions could not be calculated on arbitrarily fine meshes, even with an unlimited amount of CPU time. Thus, to provide a balance between needed accuracy and the desire to have an approximation to the exact solution available at as many nodes as possible, a piecewise-quadratic solution on an 18:8:12 mesh is used as the representative exact solution. The 18 elements are not uniformly distributed, though the 8 and 12 are. Obviously this approach has limitations; the accuracy of the reference solution is simply not knowable. However, under the current computer limitations, the present approach combines accuracy with having a reasonably coarse mesh.

A second problem in error analysis for this problem is one of scaling. Whereas errors can be studied for each state, in this work, each state is considered equivalent, and the 2-norms are used for the error at each element boundary. From Figs. 3–6, clearly the states in this problem are of a wide range of scales. The maximum Rodrigues parameter is less than unity, whereas the position states range to over 10,000. It is clear that the Rodrigues parameters change least smoothly, especially in the first second. These errors are completely lost relative to the other errors when taking the 2-norms of unscaled variables.

To alleviate this, the state variables were all multiplied by scaling factors to balance the large and small variables. The state differential equations were scaled similarly. The optimal Lagrange multipliers are then the previous values divided by the same factors as their corresponding states. GENCODE has been modified to expect scalings for the states and costates. After trying different scalings to achieve a good balance, the final values are given in the Table 1.

With these scalings, the maxima of each state and its corresponding costate are all within 20% of each other, with the overall maxima in the eight states and eight costates ranging from 7 to 75. Time and its costate are the largest, whereas θ_1 and its costate are the smallest. The overall range of maxima cannot be reduced significantly with this particular, admittedly simple, variable scaling.

With the variables now more reasonably scaled and with the exact solution approximated, the absolute error in each state, each costate, and each control were calculated at each element boundary for which the exact solution was available. These errors were calculated for a range of element distributions and shape function orders, at which point the L^2 norm of these error time histories were then calculated. These values were then plotted vs the number of unknowns and the CPU time necessary solve the problem in Figs. 8 and 9.

To compare with results from the earlier section, uniform meshes and uniform refinements were desired. The smallest mesh on which the code would converge for an equal number of evenly distributed elements was four elements per phase (4:4:4). However, from that starting point, the code would not converge when splitting to eight elements per phase. Thus, six uniformly spaced elements per phase (6:6:6) were tried, inasmuch as that provided the most opportunities for uniform splits before running out of computer memory.

As the parameters plot shows (Fig. 8), the errors can be significantly reduced by changing to higher-order shape functions rather than continuously uniformly refining the mesh, with reductions demonstrated up to 60%. However, the CPU time plot (Fig. 9) highlights the cost of these particular parameters: Doubling the number of elements for the first-order solution was no more efficient than

Table 1 Scaling factors for variables in the missile problem

Variable	Scaling
x_1	0.0017
x_2	0.002
x_3	0.003
V	0.04
θ_1	23
θ_2	13
θ_3	20
t	3

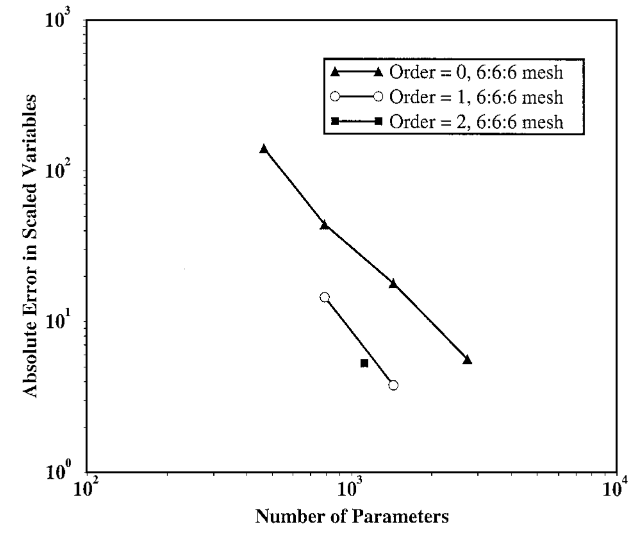

Fig. 8 Absolute error in states and costates vs number of parameters.

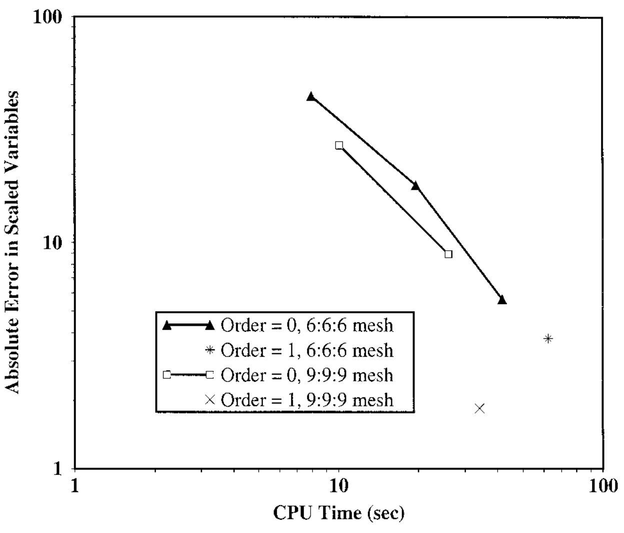

Fig. 9 Absolute error in states and costates vs CPU time.

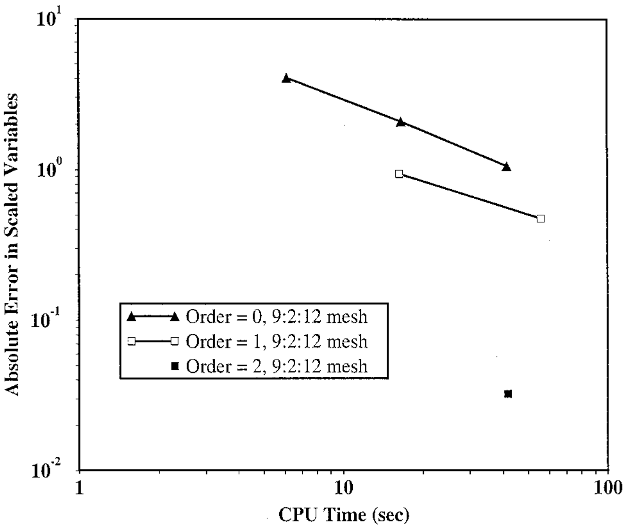

Fig. 10 Absolute error in states and costates vs CPU time.

simply using zeroth-order elements, though the results were better for the 9:9:9 solution because the code converges more easily with more elements in the first phase.

Note, however, that in determining the usefulness of higher-order shape functions, in some sense these CPU time plots are not as important as the parameter plots. All of the CPU plots presented in this work are cumulative times for moving from an initial solution, uniformly refining until the computer memory is exhausted. This is very important in comparison with adaptive schemes for determining the best mesh because this determines how quickly the code can get to a mesh with a certain level of accuracy. However, in laboratory simulations, this is not as important as simply having a solution with the best accuracy for a given number of parameters. In that case, the CPU time that will subsequently be involved in analysis of that trajectory is simply proportional to the number of elements.

That having been said, the next two plots are rather ironic. Discarding the notion of starting with a uniform mesh, Figs. 10 and 11 show how the errors reduce as the mesh is uniformly refined beginning with the 9:2:12 solution, which was the first converged solution to be discovered. Figures 10 and 11 highlight how important starting with a good solution is: The 6:6:6 solution was on the edge of those solutions that could converge, and as such the code had great difficulty in advancing that solution; hence, the CPU time plots for the 9:2:12 mesh are much better.

However, the number of parameters is not so clearly brought down by higher-order shape functions in this case, perhaps reflecting the nature of how the error is not calculated from an actual exact

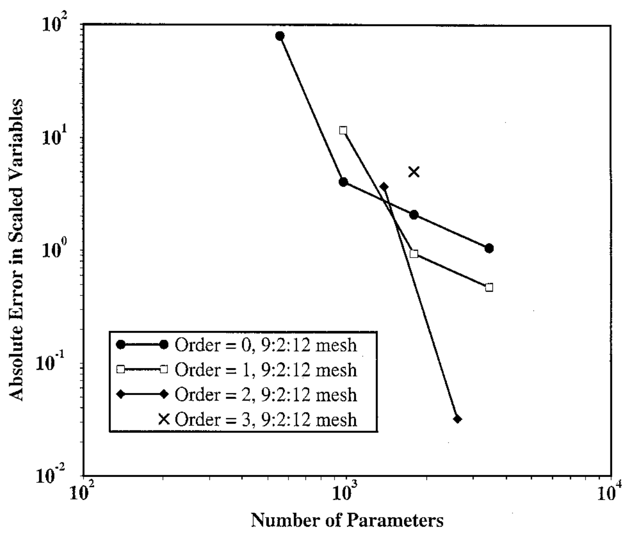

Fig. 11 Absolute error in states and costates vs number of parameters.

solution. This is especially likely in the third-order solution results. Likewise, by ignoring the first data point in each line, which is clearly driven by the errors in the time costate, the lines look more as expected. The CPU plots (Figs. 9 and 10) ignore these first points because this is the nominal solution for which no consistent way of comparing CPU time exists.

What remains to be seen is how the accuracy of solutions could be improved through a more sophisticated method of refining the mesh. This question is left to a future paper.

Conclusions

Hierarchical higher-order shape functions were included in the finite elements in time formulation for solving optimal control problems. The code is currently in place to use these shape functions to solve optimal control problems with nonlinear system dynamics, using interfaces based on symbolic mathematics to automatically generate the necessary derivative expressions. Problems can have a free or fixed final time, multiple phases, nonlinear/periodic boundary conditions, and control constraints. This code has been tested on a variety of problems, culminating in a three-phase, seven-state, two-control missile problem with full nonlinear system dynamics, where accuracy was found to be significantly improved over h-version results, with potential order-of-magnitude reductions in CPU time and numbers of parameters for a given level of error.

Acknowledgments

This work was supported primarily by NASA Grant NAG-1-1435 from NASA Langley Research Center, Daniel D. Moerder, Technical Monitor. Auxiliary funding for the first author came from a National Science Foundation fellowship. The authors thank J.-H. Yen for his help in recoding GENCODE to work with MAPLE.[13]

References

[1]Hodges, D. H., and Bless, R. R., "A Weak Hamiltonian Finite Element Formulation for Optimal Control Problems," *Journal of Guidance, Control, and Dynamics*, Vol. 14, No. 1, 1991, pp. 148–156.

[2]Bless, R. R., "Time-Domain Finite Elements in Optimal Control with Application to Launch Vehicle Guidance," NASA TR CR 4376, May 1991.

[3]Bless, R. R., and Hodges, D. H., "Finite Element Solution of Optimal Control Problems with State-Control Inequality Constraints," *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 4, 1992, pp. 1029–1032.

[4]Bless, R. R., Hodges, D. H., and Seywald, H., "A Finite Element Method for the Solution of State-Constrained Optimal Control Problems," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 5, 1995, pp. 1036–1043.

[5]Johnson, M. G., Jr., Haskins, G. T., and Hodges, D. H., "Real Time Missile Guidance System," U.S. Patent 5,435,503, July 1995.

[6]Bryson, A. E., Jr., and Ho, Y.-C., *Applied Optimal Control*, Blaisdell, Waltham, MA, 1975, Chaps. 2 and 3.

[7]Warner, M. S., and Hodges, D. H., "Treatment of Control Constraints as Zeroth-Order State Constraints in Finite Element Solution of Optimal Control Problems," *Journal of Guidance, Control, and Dynamics*, Vol. 22, No. 2, 1999, pp. 358–360.

[8]Gelfand, I. M., and Fomin, S. V., *Calculus of Variations*, Prentice–Hall, Upper Saddle River, NJ, 1963, Chap. 2.

[9]Hodges, D. H., and Hou, L.-J., "Shape Functions for Mixed *p*-version Finite Elements in the Time Domain," *Journal of Sound and Vibration*, Vol. 145, No. 2, 1991, pp. 169–178.

[10]Abramowitz, M., and Stegun, I. (eds.), *Handbook of Mathematical Functions*, National Bureau of Standards, Washington, DC, 1970.

[11]Duff, I. S., *Harwell Subroutine Library*, Computer Science and System Div., Harwell Lab., Oxfordshire, England, UK, Feb. 1988.

[12]*MACSYMA Reference Manual*, Symbolics, Inc., Burlington, MA, 1988.

[13]*Maple V Mathematics Programming Guide*, Springer–Verlag, Waterloo, Belgium, 1995.

[14]Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., *Numerical Recipes: The Art of Scientific Computing*, Cambridge Univ. Press, Cambridge, England, UK, 1986, Chap. 16.

[15]Estep, D. J., Hodges, D. H., and Warner, M. S., "Computational Error Estimation for Finite Element Solution of Missile Trajectory Optimization Problems," *SIAM Journal on Scientific Computing* (to be published).

[16]Hodges, D. H., and Johnson, M. G., Jr., "Control Variables for Finite Element Solution of Missile Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 5, 1995, pp. 1208–1211.

[17]Hodges, D. H., "Finite Rotation and Nonlinear Beam Kinematics," *Vertica*, Vol. 11, No. 1/2, 1987, pp. 297–307.

[18]Battin, R. H., *An Introduction to the Mathematics and Methods of Astrodynamics*, AIAA, Washington, DC, 1987, Sec. 10.4.