Development of an Adaptive *hp*-Version Finite Element Method for Computational Optimal Control

Final Report
NASA Grant NAG-1-1435
Apr. 22, 1992 – Nov. 30 1994

Prof. Dewey H. Hodges, Principal Investigator
Michael S. Warner, Graduate Research Assistant
School of Aerospace Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332-0150

Research Supported by
Spacecraft Control Branch
Technical Monitor: Dr. Daniel D. Moerder
Mail Stop 161
NASA Langley Research Center
Hampton, Virginia 23681-0001

1 Introduction

In this research effort, the usefulness of *hp*-version finite elements and adaptive solution-refinement techniques in generating numerical solutions to optimal control problems has been investigated. Under NAG-939, a general FORTRAN code was developed which approximated solutions to optimal control problems with control constraints and state constraints [1, 2]. Within that methodology, to get high-order accuracy in solutions, the finite element mesh would have to be refined repeatedly through bisection of the entire mesh in a given phase. In the current research effort, the order of the shape functions in each element has been made a variable, giving more flexibility in error reduction and smoothing. Similarly, individual elements can each be subdivided into many pieces, depending on the local error indicator, while other parts of the mesh remain coarsely discretized. The problem remains to reduce and smooth the error while still keeping computational effort reasonable enough to calculate time histories in a short enough time for on-board applications.

As an aid in evaluation of the error for use in solution refinement, in optimal control problems, the integral of the boundary value problem (the Hamiltonian) can be made to equal zero. Using this information alone, a reasonable error indicator can be developed, but as is shown in later sections, it is not consistently accurate and only tends to model parts of the error well. However, the Hamiltonian can be used as a check of the more sophisticated error estimators.

The error estimator currently being developed is based on work done by Estep, *et al.*, [3], for initial value problems. This will involve the solution of a dual problem, a linear differential equation of the same order as the main problem, the behavior of which indicates the overall level of error in the main problem. *hp*-refinements will then be build around this error indicator to attempt to equidistribute the error through the time interval and attempt to provide a minimum of error for a given computational effort. It remains to be investigated if this refinement process can be accomplished in a short enough time

The rest of this paper describes the work that has already been done in developing an adaptive scheme for implementing the *hp*-version of the finite element method to solve optimatl control problems. The family of optimal control problems that can currently be solved will be defined in

section 2. Next in section 3, the variables of the problem will be modelled using shape functions, thereby establishing the algebraic equations to be solved. In section 4, some textbook-style problems will be solved using with this method and the results will be analyzed. In section 5, the Hamiltonian will be shown to be a reasonable, but inconsistent, error measure, along with some preliminary results in adaptively refining solutions to textbook problems. Following that will be the preliminary development of the residual error indicator discussed above. Finally in section 6, an outline of future research work will be given.

2 Optimal Control Problems

A FORTRAN code has been developed which uses *hp*-version finite elements to approximate solutions to a particular subset of optimal control problems. In this section this subset of problems will be defined, and the equations to be solved using finite elements will be established.

2.1 Problem formulation

Systems being studied are governed by general, nonlinear differential equations

$$\dot{x} = f(x, u, t) \qquad x \in R^{n_x}, u \in R^{n_u}, t \in [t_0, t_f] \tag{1}$$

where the state vector x describes the state of the system, u is a vector of control variables, and t is the time. Although the methodology and the code are not so restricted, for simplicity's sake, this discussion is confined to problems with only a single set of differential equations in a single time interval. Furthermore, the functions f are assumed to be differentiable with respect to their arguments, and the states are assumed to be continuous, from initial time t_0 (presumed to be zero) to final time t_f, where t_f can be fixed or free.

General boundary conditions on the states can be specified at the initial time, the final time, or some combination of both, in the form

$$\Psi[x(t_0), x(t_f), t_f] = 0 \qquad \Psi \in R^{n_{bc}} \tag{2}$$

Let J be a cost functional to be minimized that can contain both a scalar penalty on the states at the endpoints t_0 or t_f plus an integral penalty on

the states, controls, and time:

$$J = \phi[x(t_0), x(t_f), t_f] + \int_{t_0}^{t_f} L(x, u, t)\ dt \tag{3}$$

The optimal control problem is then to find the control vector time history $u(t)$ which causes the system governed by Eq. (1) to meet the boundary conditions (2) such that the given cost functional (3) is minimized. Admissible control histories are assumed to be bounded and continuous.

This formulation also allows for inequality constraints of the form:

$$g(x, u) \leq 0 \qquad g \in R^{n_g} \tag{4}$$

no more than n_u of which can be active at any one time. The function g need not be a function of the states, but it must be a function of the control vector. Otherwise it falls under the category of state constraints, which require special handling that has not been developed yet under this methodology.

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

$$g_i(x, u) + k_i^2 = 0 \qquad i = 1 \ldots n_g \tag{5}$$

To simplify notation, a vector of these squared slack variables is defined such that

$$K_i = k_i^2 \qquad i = 1 \ldots n_g \tag{6}$$

2.2 Calculus of variations

In using calculus of variations to minimize a cost functional subject to constraints [4, 5], 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

$$\begin{aligned} J' = {} & \phi[x(t_0), x(t_f), t_f] + \nu^T \Psi[x(t_0), x(t_f), t_f] \\ & + \int_{t_0}^{t_f} \left\{ L(x, u, t) + \lambda^T [f(x, u, t) - \dot{x}] + \mu^T [g(x, u) + K] \right\} dt \end{aligned} \tag{7}$$

which equals J if all the constraints are satisfied. The functions $\lambda(t)$ are known as the costates.

Next, to simplify notation, define quantities

$$H \equiv L + \lambda^T f + \mu^T (g + K) \tag{8}$$

$$\Phi \equiv \phi + \nu^T \Psi \tag{9}$$

where H is known as the Hamiltonian of the system. Also define $x_f \equiv x(t_f)$ and $x_0 \equiv x(t_0)$, thus reducing J' to

$$J' = \Phi(x_0, x_f, t_f) + \int_{t_0}^{t_f} \left[H(x, u, t) - \lambda^T \dot{x} \right] \, dt \tag{10}$$

To satisfy the first-order necessary conditions for a local minimum, the authors of Ref. [5] find an extremal of J' by varying the control vector, with the variation in the state vector (and other variables) being a result of the variation in the control vector. Here instead, the development in [1] is followed in which the first variation of J' is taken, allowing independent variations in the states, state rates, controls, Lagrange multipliers, slack variables, and final time, yielding

$$\begin{aligned} \delta J' = {} & \delta \nu^T \Psi + \frac{\partial \Phi}{\partial x_0} dx(t_0) + \frac{\partial \Phi}{\partial x_f} dx(t_f) + \left[\frac{\partial \Phi}{\partial t} + H - \lambda^T \dot{x} \right]_{t_f} dt_f \\ & + \int_{t_0}^{t_f} \left[\frac{\partial H}{\partial x} \delta x + \frac{\partial H}{\partial u} \delta u - \lambda^T \delta \dot{x} + \delta \lambda^T (f - \dot{x}) \right. \\ & \left. + \delta \mu^T (g + K) + \mu^T \delta K \right] \, dt \end{aligned} \tag{11}$$

where

$$\delta K_i = 2 k_i \delta k_i \qquad i = 1 \ldots n_g \tag{12}$$

This results in a weaker form of the equations, though as will be shown, the equations developed in [5] are all still satisfied.

The variations denoted by δ are variations holding time fixed (appropriate for variations of quantities under an integral), while those with a d are total variations or differentials, which allow time to vary (appropriate for varying quantities at an end point). These variations are related at the end points by

$$dx(t_0) = \delta x(t_0) \tag{13}$$

$$dx(t_f) = \delta x(t_f) + \dot{x}(t_f) dt_f \tag{14}$$

since t_f is allowed to vary while t_0 is not.

In an attempt to group terms by their variational coefficient, one can remove the time derivatives of variational parameters by integrating $\lambda^T \delta\dot{x}$ by parts and expand the total variations at the end points. Eq. (11) then becomes

$$\begin{aligned}\delta J' = {} & \delta\nu^T \Psi + \frac{\partial \Phi}{\partial x_0}\delta x(t_0) + \frac{\partial \Phi}{\partial x_f}\delta x(t_f) - \lambda^T \delta x\Big|_{t_0}^{t_f} \\ & + \left[\frac{\partial \Phi}{\partial x_f} - \lambda^T\right]_{t_f} \dot{x}(t_f)dt_f + \left(\frac{\partial \Phi}{\partial t} + H\right)_{t_f} dt_f \\ & + \int_{t_0}^{t_f} \left[\frac{\partial H}{\partial x}\delta x + \frac{\partial H}{\partial u}\delta u + \dot{\lambda}^T \delta x + \delta\lambda^T (f - \dot{x})\right. \\ & \left. + \delta\mu^T (g + K) + \mu^T \delta K\right] \, dt\end{aligned} \tag{15}$$

Defining subscripts on H to denote partial derivatives and rearranging terms gives:

$$\begin{aligned}\delta J' = {} & \delta\nu^T \Psi + \left(\frac{\partial \Phi}{\partial t} + H\right)_{t_f} dt_f + \left(\frac{\partial \Phi}{\partial x_0} - \lambda^T\right)_{t_0} \delta x(t_0) \\ & + \left(\frac{\partial \Phi}{\partial x_f} - \lambda^T\right)_{t_f} \dot{x}(t_f)dt_f + \left(\frac{\partial \Phi}{\partial x_f} - \lambda^T\right)_{t_f} \delta x(t_f) \\ & + \int_{t_0}^{t_f} \left[H_u \delta u + H_x \delta x + \dot{\lambda}^T \delta x + \delta\lambda^T (f - \dot{x})\right. \\ & \left. + \delta\mu^T (g + K) + \mu^T \delta K\right] \, dt\end{aligned} \tag{16}$$

Now, defining $\hat{H}$ and $\hat{\lambda}$ as

$$\hat{\lambda}^T \equiv \begin{cases} -\dfrac{\partial \Phi}{\partial x_0} & t = t_0 \\ \dfrac{\partial \Phi}{\partial x_f} & t = t_f \end{cases} \tag{17}$$

$$\hat{H} \equiv \frac{\partial \Phi}{\partial t} \tag{18}$$

and combining terms yields:

$$\begin{aligned}\delta J' = {} & \delta\nu^T\Psi + \left(\hat{H} + H\right)_{t_f} dt_f + \left[\left(\hat{\lambda} - \lambda\right)^T dx\right]_{t_0}^{t_f} \\ & + \int_{t_0}^{t_f} \left[H_u\delta u + H_x\delta x + \dot{\lambda}^T\delta x + \delta\lambda^T f + \delta\lambda^T\dot{x}\right. \\ & \left. + \delta\mu^T(g + K) + \mu^T\delta K\right] dt\end{aligned} \tag{19}$$

In an analagous way to finding the extremum of a scalar function of many variables, the first order necessary condition for a extremal solution to the functional J' is that $\delta J' = 0$ for arbitrary independent variations in the states, costates, time, slack variables and other Lagrange multipliers. Since this includes the case where all the variations except any one are zero, this condition implies that for this cost functional to have a minimum, all the quantities multiplied by each of the variations must be zero. At the boundaries, this results in the following equations:

$$\lambda(t_0) + \hat{\lambda}(t_0) = 0 \tag{20}$$

$$\lambda(t_f) + \hat{\lambda}(t_f) = 0 \tag{21}$$

$$\Psi = 0. \tag{22}$$

Also, if the final time is indeed free to vary, then this relation must also hold:

$$H + \hat{H} = 0 \tag{23}$$

otherwise, $dt_f = 0$ and Eq. (23) can be ignored. In appendix A the difference between these two cases will be discussed.

The integral in Eq. (19) is made equal to zero in continuous time by enforcing the following equations:

$$\dot{\lambda}^T(t) + H_x(x, \lambda, u, t) = 0 \tag{24}$$

$$\dot{x}(t) - f(x, u, t) = 0 \tag{25}$$

$$H_u = 0 \tag{26}$$

(hereby refered to as the costate equations, state equations, and the optimality condition respectively), and the control constraint equations:

$$g + K = 0 \tag{27}$$

$$2\mu_i k_i = 0 \qquad i = 1 \ldots n_g \tag{28}$$

These equations are equivalent to the Euler-Lagrange equations developed in [5] and define a two-point boundary value problem in x and λ.

To approximately solve this problem using *hp*-version finite elements, a form of these equations will be developed such that all the boundary conditions are enforced weakly through the use of Lagrange multipliers. This will allow shape functions to be chosen that do not need to meet the boundary conditions, thus allowing the same set of shape functions to be used to solve a wide class of optimal control problems. This way the number of equations to be solved is not affected by the boundary conditions, except for fixed vs. free t_f.

To look at the equations on the interior of the time interval, first assume the boundary conditions are satisfied. Eq. (19) then becomes:

$$\begin{aligned} \delta J' = \int_{t_0}^{t_f} \Big[& H_u \delta u + H_x \delta x - \delta \dot{x}^T \lambda + \delta \lambda^T f + \delta \dot{\lambda}^T x \\ & + \delta \mu^T (g + K) + \mu^T \delta K \Big] \; dt \end{aligned} \tag{29}$$

At this stage, the analogy can be drawn between variational methods and Galerkin methods. Interpreting the variations in the variables as test functions, the equation $\delta J' = 0$ becomes the representation of the residuals in meeting Eqs. (24 – 28) being orthogonal to the test functions $(\delta x, \delta u, \ldots)$. This observation will become important when we later look at *a posteriori* error estimators.

2.3 Discretization of the problem

Again looking at Eq. (29) as a variational problem, to solve $\delta J' = 0$, the infinite dimensional problem is simplified by projecting the true unknown solution onto a finite set of piecewise polynomials. The variations are then approximated using piecewise polynomials of appropriate order to generate a set of nonlinear equations.

Obviously, different choices for these polynomials yield different sets of equations. Previous work [2] used discontinuous, piecewise-constant polynomials for the main variables. This choice is prudent for problems which have discontinuities in the states or costates, which many optimal control problems do, so 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. (29) to be done by inspection rather than by numerical quadrature. Even the simplest continuous polynomial approximations will require numerical quadrature over an element.

As Eq. (29) stands though, with 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, adding to the complexity (specifically, the dimension) of the problem. To avoid this, continuous approximations for δx and $\delta\lambda$ are used, and the terms $\dot{\lambda}^T\delta x$ and $\delta\lambda^T\dot{x}$ are integrated by parts. Eq. (29) then becomes:

$$\begin{aligned}\delta J' = \lambda^T\delta x\Big|_{t_0}^{t_f} - \delta\lambda^T x\Big|_{t_0}^{t_f} + \int_{t_0}^{t_f} \Big[H_u\delta u + H_x\delta x - \lambda^T\delta\dot{x} + \delta\lambda^T f \\ + \delta\dot{\lambda}^T x + \delta\mu^T(g+K) + \mu^T\delta K\Big]\ dt\end{aligned} \tag{30}$$

Next, the time interval is broken up into N not necessarily equal length time elements, Δt_i, such that the time at each element boundary (or *node*) $\hat{t}_i$ is calculated as:

$$\hat{t}_1 = t_0 \tag{31}$$

$$\hat{t}_i = \hat{t}_{i-1} + \Delta t_i \qquad i = 2,\ldots,N+1 \tag{32}$$

and the states and controls at these nodes are defined to be

$$\hat{x}_i = x(\hat{t}_i) \qquad i = 1,\ldots,N+1 \tag{33}$$

$$\hat{u}_i = u(\hat{t}_i) \qquad i = 1,\ldots,N+1 \tag{34}$$

The time within the i^{th} element, t_i is expressed as $t_i = \hat{t}_{i-1} + \tau\Delta t_i$ where $0 \le \tau \le 1$ and

$$\tau = \frac{t_i - \hat{t}_{i-1}}{\Delta t_i} \tag{35}$$

so that

$$\frac{d(\cdot)}{dt} = \frac{1}{\Delta t_i}\frac{d(\cdot)}{d\tau} \equiv \frac{1}{\Delta t_i}(\cdot)' \tag{36}$$

Substituting these relationships into Eq. (30) gives:

$$\begin{aligned}\delta J' = & \left.\hat{\lambda}^T \delta x\right|_{t_0}^{t_f} - \left.\hat{x}^T \delta\lambda\right|_{t_0}^{t_f} \\ & + \sum_i^N \Delta t_i \int_0^1 \Big[(H_u)_i \, \delta u_i + (H_x)_i \, \delta x_i + \frac{\delta\lambda_i'^T}{\Delta t_i} x_i - \frac{\delta x_i'^T}{\Delta t_i} \lambda_i \\ & + \delta\lambda_i^T f_i + \delta\mu_i^T (g_i + K_i) + \mu_i^T \delta K_i \Big] \, d\tau \end{aligned} \tag{37}$$

In this equation, a subscript i on a variable refers to the value of that variable within the i^{th} element, while the subscript on a function indicates the value of that function evaluated using variables within the i^{th} element.

3 Shape Functions

In this section, the specific shape functions alluded to in the previous section will be defined and used to solve Eq. (37). What will result is a set of nonlinear algebraic equations to be solved numerically.

Following the work of [8], to avoid jump-terms while also keeping the shape functions as simple as possible, C^0 shape functions for the variational quantities are implemented in each element. These are in terms of nodal values ($\hat{\cdot}$) and polynomial coefficients on element interiors ($\bar{\cdot}$).

$$\delta\lambda_i = \delta\hat{\lambda}_i(1-\tau) + \delta\hat{\lambda}_{i+1}\tau + \sum_{j=1}^{n_b-1} (1-\tau)\tau\beta_j(\tau)\delta\bar{\lambda}_{ij} \tag{38}$$

$$\delta x_i = \delta\hat{x}_i(1-\tau) + \delta\hat{x}_{i+1}\tau + \sum_{j=1}^{n_b-1} (1-\tau)\tau\beta_j(\tau)\delta\bar{x}_{ij} \tag{39}$$

Here n_b is the order of the shape function polynomial being used, with $n_b = 1$ the summation would be ignored, and this represents what is used in the h-version development. The functions $\beta_j(\tau)$ are Jacobi polynomials of order $(j-1)$ as detailed in [9].

The time derivative of these expressions is of the form

$$\frac{d\delta x_i}{d\tau} \equiv \delta x_i' = \Delta t_i \delta\dot{x}_i = -\delta\hat{x}_i + \delta\hat{x}_{i+1} + \sum_{j=1}^{n_b-1} \gamma_j(\tau)\delta\bar{x}_{ij} \tag{40}$$

where

$$\gamma_j(\tau) \equiv [(1-\tau)\tau\beta_j(\tau)]' \tag{41}$$

Similar expressions hold for the costates.

For the states and costates shape functions are chosen that are only continuous within the element:

$$x_i = \begin{cases} \hat{x}_i & \tau = 0 \\ \sum_{j=1}^{n_b} \alpha_j(\tau)\bar{x}_{ij} & 0 < \tau < 1 \\ \hat{x}_{i+1} & \tau = 1 \end{cases} \qquad \lambda_i = \begin{cases} \hat{\lambda}_i & \tau = 0 \\ \sum_{j=1}^{n_b} \alpha_j(\tau)\bar{\lambda}_{ij} & 0 < \tau < 1 \\ \hat{\lambda}_{i+1} & \tau = 1 \end{cases} \tag{42}$$

where the functions $\alpha_j(\tau)$ are polynomials of order $j-1$ as derived in [8] to simplify the algebraic equations later on. Note that $\hat{x}_i$ and $\hat{\lambda}_i$ are discrete values, distinct from the shape functions within the element. The boundary conditions on the states and costates will be enforced weakly through these nodal values.

For the controls, control constraint Lagrange multipliers, slack variables, and their variations, again no time derivatives exist in Eq. (37), so the same shape functions are used:

$$\delta u_i = \sum_{j=1}^{n_b} \alpha_j(\tau)\delta\bar{u}_{ij} \tag{43}$$

$$\delta\mu_i = \sum_{j=1}^{n_b} \alpha_j(\tau)\delta\bar{\mu}_{ij} \qquad \delta k_i = \sum_{j=1}^{n_b} \alpha_j(\tau)\delta\bar{k}_{ij} \tag{44}$$

$$u_i = \sum_{j=1}^{n_b} \alpha_j(\tau)\bar{u}_{ij} \qquad 0 < \tau < 1 \tag{45}$$

$$\mu_i = \sum_{j=1}^{n_b} \alpha_j(\tau)\bar{\mu}_{ij} \qquad 0 < \tau < 1 \qquad k_i = \sum_{j=1}^{n_b} \alpha_j(\tau)\bar{k}_{ij} \qquad 0 < \tau < 1 \tag{46}$$

No nodal values of the controls, slack variables, or control constraint Lagrange mulipliers are needed in this problem formulation. They can be calculated as needed after the fact through the optimality condition and the control constraint equations of the previous section, Eqs. (26) – (28).

Substituting the shape functions from Eqs. (42) – (46) into the integral which remains in the cost function (37) gives

$$
\begin{aligned}
\delta J' = {} & \hat{\lambda}_{N+1}^T \delta x_{N+1} - \hat{\lambda}_1^T \delta x_1 - \hat{x}_{N+1}^T \delta \lambda_{N+1} + \hat{x}_1^T \delta \lambda_1 \\
& + \sum_i^N \Delta t_i \int_0^1 \left\{ (H_u)_i \left(\sum_{j=1}^{n_b} \alpha_j(\tau) \delta \bar{u}_{ij} \right) \right. \\
& + (H_x)_i \left[\delta \hat{x}_i (1-\tau) + \delta \hat{x}_{i+1} \tau + \sum_{j=1}^{n_b - 1} \epsilon_j(\tau) \delta \bar{x}_{ij} \right] \\
& + f_i^T \left[\delta \hat{\lambda}_i (1-\tau) + \delta \hat{\lambda}_{i+1} \tau + \sum_{j=1}^{n_b - 1} \epsilon_j(\tau) \delta \bar{\lambda}_{ij} \right] \\
& + \left[\sum_{j=1}^{n_b} \alpha_j(\tau) \delta \bar{\mu}_{ij} \right]^T \left[g_i + \left(\sum_{j=1}^{n_b} \alpha_j(\tau) \bar{k}_{ij} \right)^2 \right] \\
& \left. + 2 \left(\sum_{j=1}^{n_b} \alpha_j(\tau) \bar{\mu}_{ij} \right)^T \left(\sum_{j=1}^{n_b} \alpha_j(\tau) \bar{k}_{ij} \right) \sum_{j=1}^{n_b} \alpha_j(\tau) \delta \bar{k}_{ij} \right\} d\tau \\
& + \sum_i^N \int_0^1 \left[-\delta \hat{\lambda}_i + \delta \hat{\lambda}_{i+1} + \sum_{j=1}^{n_b - 1} \gamma_j(\tau) \delta \bar{\lambda}_{ij} \right]^T \sum_{j=1}^{n_b} \alpha_j(\tau) \bar{x}_{ij} \, d\tau \\
& - \sum_i^N \int_0^1 \left[-\delta \hat{x}_i + \delta \hat{x}_{i+1} + \sum_{j=1}^{n_b - 1} \gamma_j(\tau) \delta \bar{x}_{ij} \right]^T \sum_{j=1}^{n_b} \alpha_j(\tau) \bar{\lambda}_{ij} \, d\tau
\end{aligned}
\tag{47}
$$

where

$$
\epsilon_j(\tau) \equiv (1-\tau)\tau\beta_j(\tau) \tag{48}
$$

Note that the only nodal values of the states and costates that appear here are at the endpoints of the time interval. The other internal nodal values can be generated from the state equations once the other nodal and interior values have been solved for.

Because the polynomials $\alpha(\tau)$ and $\gamma(\tau)$ were chosen to be orthogonal, Eq. (47) reduces to

$$
\begin{aligned}
\delta J' = {} & \hat{\lambda}_{N+1}^T \delta x_{N+1} - \hat{\lambda}_1^T \delta x_1 - \hat{x}_{N+1}^T \delta \lambda_{N+1} + \hat{x}_1^T \delta \lambda_1 \\
& + \sum_i^N \Delta t_i \int_0^1 \left\{ (H_u)_i \left(\sum_{j=1}^{n_b} \alpha_j(\tau) \delta \bar{u}_{ij} \right) \right. \\
& + \delta \hat{x}_i^T (H_x)_i (1-\tau) + \delta \hat{x}_{i+1}^T (H_x)_i \tau + \delta \hat{\lambda}_i^T f_i (1-\tau) + \delta \hat{\lambda}_{i+1}^T f_i \tau \\
& + \sum_{j=1}^{n_b-1} \left[\epsilon_j(\tau) \delta \bar{x}_{ij}^T (H_x)_i + \epsilon_j(\tau) \delta \bar{\lambda}_{ij}^T f_i \right] \\
& + \left(\sum_{j=1}^{n_b} \alpha_j(\tau) \delta \bar{\mu}_{ij} \right)^T \left[g_i + \left(\sum_{j=1}^{n_b} \alpha_j(\tau) \bar{k}_{ij} \right)^2 \right] \\
& \left. + 2 \left(\sum_{j=1}^{n_b} \alpha_j(\tau) \bar{\mu}_{ij} \right)^T \left(\sum_{j=1}^{n_b} \alpha_j(\tau) \bar{k}_{ij} \right) \sum_{j=1}^{n_b} \alpha_j(\tau) \delta \bar{k}_{ij} \right\} d\tau \\
& + \sum_i^N \left(-\delta \hat{\lambda}_i^T \bar{x}_{i,1} + \delta \hat{\lambda}_{i+1}^T \bar{x}_{i,1} - \sum_{j=2}^{n_b} \delta \bar{\lambda}_{i,j-1}^T \bar{x}_{ij} \right) \\
& + \sum_i^N \left(\delta \hat{x}_i^T \bar{\lambda}_{i,1} - \delta \hat{x}_{i+1}^T \bar{\lambda}_{i,1} + \sum_{j=2}^{n_b} \delta \bar{x}_{i,j-1} \bar{\lambda}_{ij} \right)
\end{aligned}
\tag{49}
$$

Arranging terms by their variational coefficient gives

$$
\begin{aligned}
\delta J' = {} & \delta\hat{\lambda}_1^T \left[-\hat{x}_1 + \bar{x}_{1,1} - \Delta t_1 \int_0^1 (1-\tau) f_1 \, d\tau \right] \\
& + \sum_{i=2}^{N} \delta\hat{\lambda}_i^T \left[-\bar{x}_{i-1,1} - \Delta t_{i-1} \int_0^1 \tau f_{i-1} \, d\tau \right. \\
& \qquad \left. + \bar{x}_{i,1} - \Delta t_i \int_0^1 (1-\tau) f_i \, d\tau \right] \\
& + \sum_{i=1}^{N} \sum_{j=1}^{n_b-1} \delta\bar{\lambda}_{ij}^T \left[-\bar{x}_{i,j+1} + \Delta t_i \int_0^1 \epsilon_j(\tau) f_i \, d\tau \right] \\
& + \delta\hat{\lambda}_{N+1}^T \left[-\bar{x}_{N,1} - \Delta t_N \int_0^1 \tau f_N \, d\tau + \hat{x}_{N+1} \right] \\
& + \delta\hat{x}_1^T \left[\hat{\lambda}_1 - \bar{\lambda}_{1,1} - \Delta t_i \int_0^1 (1-\tau)(H_x)_1 \, d\tau \right] \\
& + \sum_{i=2}^{N} \delta\hat{x}_i^T \left[-\bar{\lambda}_{i,1} - \Delta t_i \int_0^1 (1-\tau)(H_x)_i \, d\tau \right. \\
& \qquad \left. + \bar{\lambda}_{i-1,1} - \Delta t_{i-1} \int_0^1 \tau (H_x)_{i-1} \, d\tau \right] \\
& + \sum_{i=1}^{N} \sum_{j=1}^{n_b-1} \delta\bar{x}_{ij}^T \left[\bar{\lambda}_{i,j+1} + \Delta t_i \int_0^1 \epsilon_j(\tau)(H_x)_i \, d\tau \right] \\
& + \delta\hat{x}_{N+1}^T \left[-\hat{\lambda}_{N+1} + \bar{\lambda}_{N,1} - \Delta t_N \int_0^1 \tau (H_x)_N \, d\tau \right] \\
& + \sum_{i}^{N} \sum_{j=1}^{n_b} \delta\bar{u}_{ij} \Delta t_i \int_0^1 (H_u)_i \alpha_j(\tau) \, d\tau \\
& + \sum_{i}^{N} \sum_{j=1}^{n_b} \delta\bar{\mu}_{ij} \Delta t_i \int_0^1 \alpha_j(\tau) \left[g_i + \left(\sum_{j=1}^{n_b} \alpha_j(\tau) \bar{k}_{ij} \right)^2 \right] d\tau \\
& + \sum_{i}^{N} \sum_{j=1}^{n_b} \delta\bar{k}_{ij} \Delta t_i \int_0^1 2\alpha_j(\tau) \left[\sum_{j=1}^{n_b} \alpha_j(\tau) \bar{\mu}_{ij} \right]^T \left[\sum_{j=1}^{n_b} \alpha_j(\tau) \bar{k}_{ij} \right] d\tau
\end{aligned}
\tag{50}
$$

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. In this way, the first-order necessary conditions for optimal control are approx-

imated. To supplement the equations derived from setting Eq. (50) equal to zero, the boundary conditions need to be enforced, as provided in the previous section:

$$\hat{\lambda}_1^T + \frac{\partial \phi}{\partial x_0} + \nu^T \frac{\partial \Psi}{\partial x_0} = 0 \tag{51}$$

$$\hat{\lambda}_{N+1}^T - \frac{\partial \phi}{\partial x_f} - \nu^T \frac{\partial \Psi}{\partial x_f} = 0 \tag{52}$$

$$H(t_f) + \frac{\partial \phi}{\partial t_f} + \nu^T \frac{\partial \Psi}{\partial t_f} = 0 \tag{53}$$

$$\Psi(\hat{x}_0, \hat{x}_{N+1}, t_f) = 0 \tag{54}$$

Thus, the two-point boundary value problem (20) – (28) is approximated by the set of nonlinear algebraic equations (50) – (54). When a problem has multiple phases (whether by discontinuity of the states or the differential equations, or both), the equations are similar and can also be handled by the code. The differences are that the user has to specify extra boundary conditions, including whatever will trigger the discontinuity, and then the code handles the corresponding jump conditions for the costates and Hamiltonian automatically.

In the following sections, how these equations are implemented will be explained, and example problems will be shown which demonstrate the use and performance of *hp*-version finite elements. Following that will be a discussion of how to improve the solution adaptively by gauging the error in each interval, followed by some preliminary results using this refinement technique

4 Implementation and Results

A numerical solution which makes the expressions from Eq. (50) equal to zero is found using a restricted-step Newton-Raphson method, implemented in a FORTRAN code. A restricted-step Newton-Raphson method differs from a standard Newton-Raphson method in that if the full Newton step yields a higher value of the objective function than the starting point, the step size is halved repeatedly until an improving step is achieved.

The determination of the Newton step requires the solution of a linear system of equations. The Jacobian of these equations can be very large as

shown in Figure 1, which is for the problem defined in section 4.1. The equations for the most part only depend on neighboring elements, so the Jacobian ends up being quite sparse, as seen in Figure 2 for the same problem. That feature is exploited by the use of sparse linear systems solvers from the Harwell subroutine library [11]. These routines use a special sparse version of Gaussian elimination with partial pivoting, where information is retained between calls to the routine, so subsequent linear systems to be solved using the same Jacobian structure (as is common in determining different Newton steps) or simply different right-hand sides (as is common in repeatedly step halving) take considerably less time.

The symbolic-manipulation package MACSYMA developed by Symbolics [12] is used to generate analytical partial derivatives of the system equations, boundary conditions, and control constraints for use in the Jacobian. MACSYMA is also used to generate the $\alpha(\tau)$ and $\epsilon(\tau)$ polynomials from Eq. (50), which come from a recursion formula involving derivatives and integrals of polynomials, as developed in [8]. The user specifies the order of the polynomials, and MACSYMA generates the necessary expressions, but the user does have to supply initial guesses at the polynomial coefficients.

The integrals in the equations are approximated using Gauss-Legendre integration [10], with the user selecting the number of Gauss points, which at this time is constant for all of the integrations. No fewer Gauss quadrature points should be chosen than the order of the shape functions being used plus one. This is because the zeros of the shape functions correspond to zeros of polynomials used in Gauss-Legendre integration. More Gauss points may be used, and that has shown to be more accurate in some problems, but not all, nor has the value of this extra computational effort been established. In Refs. [6] and [7], it is established that the minimum error (or most accurate stresses) in finite element approximations occur at the Gauss points.

4.1 Unconstrained 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 [5], pg. 59). The mass is acted on by a force of constant magnitude ma and variable heading $\beta(t)$. The states for this problem are position of the particle x (horizontal) and y (vertical) and the corresponding velocity

components u and v. The differential equations for this system are then:

$$\begin{aligned} \dot{x} &= u \\ \dot{y} &= v \\ \dot{u} &= a\cos\beta \\ \dot{v} &= a\sin\beta \end{aligned} \tag{55}$$

with initial conditions corresponding to a zero velocity at the origin and with terminal constraints that the particle is in horizontal flight at a given height (assumed here to be 1), yielding a boundary condition vector,

$$\begin{aligned} \Psi_1 &= u(0) \\ \Psi_2 &= v(0) \\ \Psi_3 &= x(0) \\ \Psi_4 &= y(0) \\ \Psi_5 &= y(t_f) - 1 \\ \Psi_6 &= v(t_f) \end{aligned} \tag{56}$$

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

$$J = u(t_f) \tag{57}$$

with the boundary conditions enforced by setting $\Psi = 0$.

Reference [5] 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°, 1, and 1 respectively.

The code was run for this problem for a variety of combinations of finite-element parameters. Figures 3 and 4 show the square root of the integral of the squares (i.e. the two-norm) of the relative error time histories for the states and controls as a function of the CPU time involved in calculating solutions. The data points correspond to the 2-, 4-, 8-, 16-, and 32-element solutions. The initial guess for each case was extrapolated from the converged solution for the case of half as many elements (or for one fewer shape function coefficient if only 2 elements). All of the specified boundary conditions were met to within 10^{-12} for the case when all the elements are evenly

spaced. Figures 5 and 6 show the the same error data vs. the number of free parameters in the problem, which equals the dimension of the Jacobian.

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

What is surprising is the particularly good performance of the first-order shape function through the higher error regions. It isn't until errors get below 10^{-6} in the states that the higher shape function orders get significantly better, with each of the other shape function orders being best for each subsequent 2 orders of magnitude error. That pattern gets pushed even further down and wider with the control variables. First-order is best all the way down to 10^{-8} with each subsequent higher-order shape function being best seemingly for another 3 or 4 orders of magnitude, until the code runs into problems with round-off error.

Later these results will be compared with those when the mesh is refined using an adaptive scheme. For now these results show the increase in accuracy that is possible using higher-order shape functions in solving optimal control problems.

4.2 Example with control constraint

Next a problem with a control constraint was studied, again from Ref. [5]. The problem is to minimize the cost function

$$J = \frac{1}{2}x(t_f)^2 + \frac{1}{2}\int_0^{t_f} u^2\,dt \tag{58}$$

where x and u are scalars. The system dynamics are governed by

$$\dot{x} = h(t)u \tag{59}$$

for some function $h(t)$, subject to two control inequality constraints,

$$\begin{aligned} g_1 &= u - 1 \leq 0, \qquad \text{and} \\ g_2 &= -(u+1) \leq 0 \end{aligned} \tag{60}$$

An exact solution is available [2] if the final time is chosen to be 10, the initial condition is a given constant, and

$$h(t) = 1 + t - \frac{3}{17}t^2 \tag{61}$$

The exact value for the state at the final time is $x(t_f) = -17/39$ and the optimal control is

$$u(t) = \begin{cases} -x(t_f)h(t) & 0 \le t \le 2 \\ 1 & 2 \le t \le 11/3 \\ -x(t_f)h(t) & 11/3 \le t \le 8 \\ -1 & 8 \le t \le 10 \end{cases} \tag{62}$$

This problem is not as well-behaved numerically as the previous problem, and so solutions are not readily available for the sequences of distributions of elements. Errors do reduce significantly as the mesh is refined and the shape function orders are increased, but showing those results will be deferred to Section 5 when they will be compared to results using adaptive error-reduction techniques.

The penalty of increased CPU time associated with using higher-order shape functions could be serious enough to thwart using this methodology to solve problems in real time. However, many of the element interior unknowns can be eliminated at the element level by solving a small set of nonlinear algebraic equations in which the nodal values are taken as given, splitting the problem (and hence the Jacobian) into outer and inner loops for the nodal and interior values to an element respectively. The scheme may then turn out to be especially powerful in a parallel computing environment since a different processor could be assigned to each element. The number of processors, strictly speaking, would not be required to be any larger than the number of sub-regions which are free of discontinuities.

Another way to bring down the computational effort while still reducing and smoothing errors is by more prudent selection of the element mesh and shape function orders. Starting with a coarse discretization, the optimal distribution of elements and higher-order shape functions could be found that minimizes the error for a given number of parameters. In the next section some indicators available to gauge error for use in such a process will be examined.

5 Implementation and Results with Adaptivity

To develop a means of adaptively adjusting mesh and shape function parameters, subroutines were developed to split a given element or raise the shape function order if a certain error criterion was met. Initial guesses are then determined for any new parameters, and then the new (larger) set of equations is solved again using the Newton-Raphson iteration.

This section describes two possible error indicators, one involving the calculation of a single function and one more elaborate method involving the solution to an adjoint problem. The simpler one has been implemented, and the corresponding results will be described first. The more elaborate error indicator is still in development, so the development of it will be more sketchy.

5.1 Hamiltonian as error indicator

The Hamiltonian is a logical choice for an error criterion since it is a first integral of the two point boundary value problem derived from the first-order necessary conditions for optimal control. In the problems under consideration, it contains all of the controls and costates, at least some of the states, and all of the state rates. Thus looking at how the Hamiltonian varies from its optimal value can be a valuable gauge for the magnitude of the error in the variables in the problem.

The development in appendix A shows how the optimal control problem can be cast such that the Hamiltonian at the optimal solution is always zero. With the Hamiltonian having been transformed in this way, the difference between $H(t)$ and the optimal (namely zero) can now easily be measured. What remains to be seen is how good of an indicator of error (either pointwise or integrated) in the solution that the Hamiltonian is.

The two criteria for parameter adjustment that have been looked at most closely are the deviation from zero of the Hamiltonian at any given node and the jump that the Hamiltonian makes between two adjacent nodes.

The two problems studied were the same as in Section 4. The first set of results is for the unconstrained problem in Section 4.1. The problem was first solved for 2 elements. At this point, the Hamiltonian was a non-zero constant

across both elements, so both elements were split. The resulting Hamiltonian and relative error distributions are shown in Figure 7. The "Total Error" is the two-norm of the relative errors in all the states, costates, and control; the "x-u error" includes only errors in the states and controls; and the "u error" includes only the error in the control. Relative error was unavailable at the endpoints for the states since they were constrained to be zero. Similarly, the optimal value of the control at the midpoint of the time interval is zero, so that data point will also be missing in each of these plots.

As can be seen, the Hamiltonian jumps the most in the two central elements. Using a criterion that elements should be split if the Hamiltonian jump across them was 10% of the maximum jump along the trajectory, the code split these two central elements, resulting in the smoother Hamiltonian distribution in Figure 8.

At this point, all the Hamiltonian jumps were about the same, so all the elements were split, resulting in the 12-element solution shown in Figure 9, where again some peaks developed, which were smoothed out in the 16-element solution of Figure 10. This process can continue until point the Hamiltonian and the overall error in all variables is below 10^{-10}.

In each case, the Hamiltonian peaks near the middle elements picked up the peak error in the control in that region. Meanwhile the peak in the overall state error (including the two states which did not appear in the Hamiltonian) seemed to correspond to a secondary peak in the Hamiltonian. The total error was dominated by the error in one of the costates which was a constant. Even so, the total error was in the same order of magnitude as the error in the Hamiltonian.

In the second example with the control constraints from Section 4.2, jumps in the Hamiltonian seem to better highlight regions where the mesh should be refined. Starting with 7 elements, so as not to have a node fall upon an optimal entrance or exit point for a control constraint, the Hamiltonian is constant within each phase of the problem, whether on or off the control constraint, as shown in Figure 11. The code senses the jumps in the Hamiltonian near each of the switching points and put more nodes there, resulting in the 16-element solution in Figure 12. To compare, Figure 13 shows how the error and nodes would be distributed with 17 uniformly spaced elements. The errors are an order of magnitude higher. Figure 14 shows the results of a couple of optimizing runs done on the results from Figure 13.

In all the plots for this problem, the control error is necessarily zero

along constrained arcs. Similarly, the Hamiltonian error was always zero in the last constrained arc since the Hamiltonian is actually enforced to be zero at the final time. With a constant control in that region, the equations can be integrated exactly. The plots show that to bring the error down in the early regions, the switching points have to be nailed down precisely, which is exactly what happens when optimization is based on Hamiltonian jumps. Unfortunately, the code can only make one set of element splits before resolving the problem. This bisection technique would take a long time to reach the switching points exactly. When state constraint capability is added to the code, the endpoints of a control constraint arc can then become variables.

Thus, element splitting based on Hamiltonian jumps seems to do a reasonable job of smoothing and reducing error, and the Hamiltonian itself seems to be a reasonable measure of errors within the elements, but the results are not consistant enough. The Hamiltonian, while easy to calculate from given information and a reasonable gauge for error, lacks the information about how well the differential equations are being approximated. Also lacking is a sense of how finely to discretize a given element, as demonstrated in the control constraint problem. The next *a posteriori* error estimator to be investigated is being designed to gauge the residual errors in meeting the equations within each element.

5.2 Residual error technique

As mentioned in section 2.2, the variational equations can be interpreted as residuals multiplied by test functions, the definition a Galerkin method. Since Galerkin methods minimize the two-norms of the residual errors, formulating the problem in this framework will help greatly in determining an error estimator. In this section, the preliminary work done in analyzing this system to find adequate residual error estimators will be presented.

As a starting point, the analysis will not treat complete optimal control problems, but rather two point boundary value problems (TPBVP) without controls, using the work of Estep *et al.* [3] in initial value problems as a basis. The extra constraints of the optimality condition and any control constraints will be added later. Likewise, only time-varying linear systems have been examined thus far, and each state must have exactly one given boundary condition. Dr. Estep's continued collaboration in developing this error estimator is acknowledged and appreciated.

The states are first split depending on at which endpoint the boundary condition is specified. For this case, the system being analyzed has the form

$$\begin{aligned} \dot{y} &= A(t)y \qquad 0 \le t < t_f \\ y_L(0) &= y_0 \\ y_R(t_f) &= y_f \end{aligned} \tag{63}$$

where $y \in R^n$ is partitioned as

$$y = \begin{pmatrix} y_L \\ y_R \end{pmatrix} \tag{64}$$

Using the continuous Galerkin finite element method, the object is to find the polynomial $Y \in C^q$ such that:

$$\begin{aligned} \int_0^{t_f} \left[\dot{Y} - A(t)Y, v\right]\ dt = 0 \qquad & \forall v \in D^{q-1} \\ Y_L(0) &= y_0 \\ Y_R(t_f) &= y_f \end{aligned} \tag{65}$$

where C^q is the set of all piecewise polynomials on $[0, t_f]$ of order q which are continous across element boundaries. D^q is the set of all piecewise polynomials on $[0, t_f]$ of order q which are not continuous across element boundaries.

Subtracting Eqs. (63) and (65), and defining the error as $e = Y - y$, one yields

$$\begin{aligned} \int_0^{t_f} \left[\dot{e} - A(t)e, v\right]\ dt = 0 \qquad & \forall v \in D^{q-1} \\ e_L(0) &= 0 \\ e_R(t_f) &= 0 \end{aligned} \tag{66}$$

where e is partitioned similarly to y as

$$e = \begin{pmatrix} e_L \\ e_R \end{pmatrix} \tag{67}$$

Integrating Eq. 66 by parts yields:

$$\int_0^{t_f} \left[\dot{e} - A(t)e, v\right]\ dt = e(t_f)^T v(t_f) - e(0)^T v(0) + \int_0^{t_f} \left[e, -\dot{v} - A(t)^T v\right]\ dt \tag{68}$$

where

$$\begin{aligned} e(t_f)^T v(t_f) &= e_L(t_f)^T v_L(t_f) \\ e(0)^T v(0) &= e_R(0)^T v_R(0) \end{aligned} \tag{69}$$

Now let v be the solution of

$$\begin{aligned} -\dot{v} - A(t)^T v &= e(t) \qquad 0 \le t < t_f \\ v_L(t_f) &= 0 \\ v_R(0) &= 0, \end{aligned} \tag{70}$$

reducing the right hand side of Eq. (68) extensively:

$$\int_0^{t_f} \|e\|^2 \, dt = \int_0^{t_f} [\dot{e} - A(t)e, v] \, dt \tag{71}$$

Adding in Eq. (63) gives:

$$\int_0^{t_f} \|e\|^2 \, dt = \int_0^{t_f} \left[\dot{Y} - A(t)Y, v\right] \, dt \tag{72}$$

Defining π_q as a projection operator into the space of polynomials of order q, Eq. (65) implies that $\pi_{q-1}v$ is orthogonal to $\dot{Y} - A(t)Y$, and hence it can be added to the right hand side of the dot product:

$$\int_0^{t_f} \|e\|^2 \, dt = \int_0^{t_f} \left[\dot{Y} - A(t)Y, v - \pi_{q-1}v\right] \, dt \tag{73}$$

Now, $\dot{Y} \in D^{q-1}$ so it is orthogonal to $v - \pi_{q-1}v$, reducing the above to:

$$\int_0^{t_f} \|e\|^2 \, dt = \int_0^{t_f} [-A(t)Y, v - \pi_{q-1}v] \, dt \tag{74}$$

By a similar argument, any other function in D^{q-1} can be added on the left side of the dot product, so in order to make the term on the left small, add in the projection of $A(t)Y$:

$$\int_0^{t_f} \|e\|^2 \, dt = \int_0^{t_f} \{\pi_{q-1}[-A(t)Y] - A(t)Y, -\pi_{q-1}v + v\} \, dt \tag{75}$$

Using standard calculus integral inequalities plus the inequality that

$$\int_0^{t_f} (v - \pi_{q-1}v) \, dt \le \int_0^{t_f} \|\dot{v}\| \, dt, \tag{76}$$

Eq. (75) becomes:

$$\int_0^{t_f} \|e\|^2 \, dt \leq \int_0^{t_f} \|\dot{v}\| \, dt \cdot \max_{m \leq n} \left[k_m \cdot \max_{t \in I_m} \|A(t)Y - \pi_{q-1} A(t)Y\| \right] \equiv S \cdot \max_{m \leq n} (k_m \cdot R_m) \tag{77}$$

where k_m is the length of the mth element, I_m (out of n), S is the sensitivity function, and R_m is the residual on the mth element. Note that different choices for inequalities to use result in different error estimators, which may happen to suit optimal control problems better. That will be examined as work continues.

With a target integral error of TOL, the original finite element problem (65) is solved once, and the residuals are calculated. At the first iteration, the new time interval lengths $k_m^{(2)}$ can be computed for each m as

$$\frac{k_m^{(2)}}{k_m^{(1)}} = (TOL)^2 / \left(S \cdot k_m^{(1)} \cdot R_m \right) \qquad \forall m \in [1, n] \tag{78}$$

with S assumed to be one. For subsequent iterations, the dual problem (70) is solved with the forcing function $e(t)$ approximated as the difference between the approximate solutions from the previous two iterations. Then S is calculated and used in calculating the next mesh refinement. In this way the solution is refined, equidistributing the error.

This methodology has been implemented into FORTRAN and tested on two time-varying, linear systems, but the results are still preliminary. The next step will be to include nonlinear dynamics, but from previous work of Estep [3], the main difference will be that the Jacobian of the system dynamics will replace the $A(t)$ matrix in the dual problem, retaining its linear nature, and indeed making it a reasonably trivial calculation compared to solving the huge system of nonlinear equations. Once that is working, the next step is to add scalar functions and constraints to the two-point boundary value problem, which should also be a straightforward extension.

6 Summary and Future Work

An updated version to GENCODE, developed by Hodges and Bless [1], (but without state constraints) has been developed to solve a variety of opti-

mal control problems using *hp*-version finite elements with adaptive mesh-parameter adjustment. Textbook problems with non-linear system dynamics and control constraints have already been solved for a variety of combinations of finite element parameters, including adaptively modifying the distribution of mesh parameters with the Hamiltonian as an error indicator. The Hamiltonian has been shown to be an inadequate error indicator in and of itself, necessitating more sophisticated measures if optimal mesh-parameter design is to be done. The first steps toward a residual error measure have been taken and look promising.

To that end, research will need to continue to be done concerning the techniques for error estimation and adaptive adjustment of mesh parameters. The residual error techniques will have to be further refined and tested, while analysis needs to be done to verify the experimental evidence already obtained regarding the use of the Hamiltonian as an error indicator.

The methodology to handle state constraints, a realistic part of many aerospace applications, will have to be developed and implemented. Once that capability is added, control constraints will be able to be handled with variable endpoints as zeroth order state constraints, if desired, greatly improving the accuracy in solving those kinds of problems. That way any adaptive routine will not have to iterate as much to determine the optimal switching points.

References

[1] R. R. Bless, "Time-Domain Finite Elements in Optimal Control with Application to Launch Vehicle Guidance." Ph.D. Dissertation, School of Aerospace Engineering, Georgia Institute of Technology, Mar. 1991 (also NASA CR 4376, Mar. 1991).

[2] R. R. Bless and D. H. Hodges, "Finite Element Solution of Optimal Control problems with State-control inequality Constraints," *Journal of Guidance, Control, and Dynamics*, v. 15, n. 4, 1992, pp. 1029.

[3] D. Estep and D. French, "Global Error Control for the Continuous Galerkin Finite Element Method for Ordinary Differential Equations," California Institute of Technology Technical Report CRPC-93-7, Pasedena, CA, 1993.

[4] I. M. Galfand and S. V. Fomin *Calculus of Variations*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1963.

[5] A. E. Bryson and Y.-C. Ho, *Applied Optimal Control - Optimization, Estimation, and Control*, Hemisphere Publishing Corporation, New York, NY, 1975.

[6] T. Moan, "On the Local Distribution of Errors By Finite Element Approximations," *Theory and Practice in Finite Element Structural Analysis*, 1973 Tokyo Seminar on Finite Element Analysis.

[7] J. Barlow, "Optimal Stress Locations in Finite Element Models," *International Journal for Numerical Mathematics in Engineering*, v. 10, 1976, pp. 243 - 251.

[8] D. H. Hodges and L.-J. Hou, "Shape Functions for Mixed p-version Finite Elements in the Time Domain." *Journal of Sound and Vibration*, v. 145, n. 2, Mar. 8, 1991, pp. 169 - 178.

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

[10] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W.T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, Cambridge, 1986, Chapter 16.

[11] I. S. Duff, Harwell Subroutine Library, ComputerScience and Systems Division, Harwell Laboratory, Oxfordshire, England, 1988, Chapter M.

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

A Reformulating the Hamiltonian

In this section, we will develop the properties of the Hamiltonian and reformulate it to become a convenient measure of error in the system. The Hamiltonian will be studied for the general class of optimal control problems studied in section 2, but without control constraints as that simply complicates notation.

Borrowing notation from section 2, define the Hamiltonian in the standard way as

$$H = L(x, u, t) + \lambda^T f(x, u, t) \tag{79}$$

From the first order necessary conditions for optimality (see section 2), the condition on the Hamiltonian at the final time is determined from

$$\left[H(t_f) + \frac{\partial \phi}{\partial t_f} + \nu^T \frac{\partial \Psi}{\partial t_f} \right] dt_f = 0 \tag{80}$$

If t_f is fixed, then $dt_f = 0$, and the Hamiltonian can remain free at the final time. Otherwise dt_f is a free variation and thus

$$H(t_f) = -\frac{\partial \phi}{\partial t_f} - \nu^T \frac{\partial \Psi}{\partial t_f} \tag{81}$$

at the optimal solution. In either case, no conditions are imposed on the Hamiltonian at any time other than the final time. Often t_f is not constrained or penalized. In this case, Eq. (81) reduces to

$$H(t_f) = 0 \tag{82}$$

In the case when t_f is a fixed number τ, the code is set up to assume that t_f is still a free variable. This way the same set of equations is solved by the code for any general problem. The user has to add the extra constraint

$$\Psi_{p+1} = t_f - \tau \tag{83}$$

and Eq. (81) becomes the dummy equation

$$H(t_f) = \nu_{p+1}, \tag{84}$$

where ν_{p+1} is the associated extra dummy unknown. Whatever the code decides the optimal value of H should be at the final time from the other equations, ν_{p+1} will be set to that.

To get an idea of the behavior of the Hamiltonian along the optimal trajectory, take the time derivative:

$$\dot{H} = \frac{\partial H}{\partial x}\dot{x} + \frac{\partial H}{\partial \lambda}\dot{\lambda} + \frac{\partial H}{\partial u}\dot{u} + \frac{\partial H}{\partial t} \tag{85}$$

For the optimal trajectory, the first two terms cancel, and the third is zero, reducing (85) to

$$\dot{H} = \frac{\partial H}{\partial t} \equiv H_t \tag{86}$$

By definition, if the system is autonomous, $H_t = 0$, which implies that $\dot{H} = 0$, and thus

$$H(t) = H(t_f), \tag{87}$$

though that's not an enforced constraint.

If $H(t_f)$ were a known quantity, this could be used as an independent check on the error in an obtained solution. Thus we would like to make $H(t_f)$ equal to a known constant and make $H(t)$ equal to $H(t_f)$ for a general problem formulation.

One solution to both problems is to make time a state in the problem, i.e.,

$$x_{n+1} = t \tag{88}$$

which changes the cost function to:

$$J = \phi[x(t_f)] + \int_0^{t_f} L(x, u)\, dt, \tag{89}$$

where x here includes both the old states plus time, with constraints

$$\dot{x}_i = f_i(x, u) \quad i \in 1...n \tag{90}$$

$$\dot{x}_{n+1} = 1 \tag{91}$$

$$\Psi_i[x(t_0), x(t_f)] = 0 \quad i \in 1...p \tag{92}$$

$$\Psi_{p+1} = x_{n+1}(t_0) = 0 \tag{93}$$

Now if t_f is a fixed constant τ, we add:

$$\Psi_{p+2} = x_{n+1}(t_f) - \tau = 0 \tag{94}$$

In any case, Ψ is not dependent on t_f explicitly, so we always enforce:

$$H(t_f) = 0. \tag{95}$$

Similarly, since explicit dependence on time has been removed from the problem,

$$\frac{\partial H}{\partial t} = 0, \tag{96}$$

even for a non-autonomous problem, so $H(t) = H(t_f) = 0$ over the whole trajectory.

The new state equation for the time state is enforced with a new costate. This costate is not as meaningless as one might assume at first glace. To find an interpretation for it, we first define a psuedo-Hamiltonian, $\tilde{H}$.

$$\tilde{H} = L(x,u) + \sum_{1}^{n} \lambda_i f_i(x,u). \tag{97}$$

Since along the optimal trajectory the states, costates, and control will be the same in either formulation and the new state simply substitutes for the running time, $\tilde{H}(t)$ is the same as the Hamiltonian before the time state was added. The time derivative of the new Hamiltonian is then

$$\dot{H} = \dot{\tilde{H}} + \dot{\lambda}_{n+1} = 0 \tag{98}$$

and the enforced boundary condition is:

$$H(t_f) = \tilde{H}(t_f) + \lambda_{n+1}(t_f) = 0 \tag{99}$$

These two equations imply that

$$\dot{\lambda}_{n+1} = -\dot{\tilde{H}} \tag{100}$$

and

$$\lambda_{n+1}(t_f) = -\tilde{H}(t_f) \tag{101}$$

which indicates that the extra costate obtained by adding a time state is in fact the old Hamiltonian. This relationship boils down to something very simple if the problem is autonomous.

From the necessary conditions for optimality, the costate equation and boundary conditions for the new costate are

$$\begin{aligned} \lambda_{n+1}(t_0) &= \nu_{p+1} \\ \lambda_{n+1}(t_f) &= \nu_{p+2} \qquad t_f \text{ fixed} \\ &= 0 \qquad\quad t_f \text{ free} \end{aligned}$$

For an autonomous problem,

$$\frac{\partial H}{\partial x_{n+1}} = 0. \tag{102}$$

This implies both that

$$\dot{\lambda}_{n+1} = 0 \rightarrow \lambda_{n+1}(t) = \lambda_{n+1}(t_f) \tag{103}$$

and that $\lambda_{n+1}(t_f)$ will not appear in the other costate equations or the optimality equation, $H_u = 0$ (and of course not the system dynamics.) Thus the only place $\lambda_{n+1}(t_f)$ enters the problem is by balancing $\tilde{H}(t_f)$ in Eq. (101). Comparing this to Eqs. (84) and (82) it is clear that for autonomous problems, free- or fixed-time, that the additional costate associated with the time state simply acts as a dummy Lagrange multiplier. So in adding the time state, effectively the Hamiltonian absorbs the right hand side of its terminal boundary condition so that the new Hamiltonian can always be zero. This is essentially what happens in non-autonomous problems except that it is not a constant being absorbed.

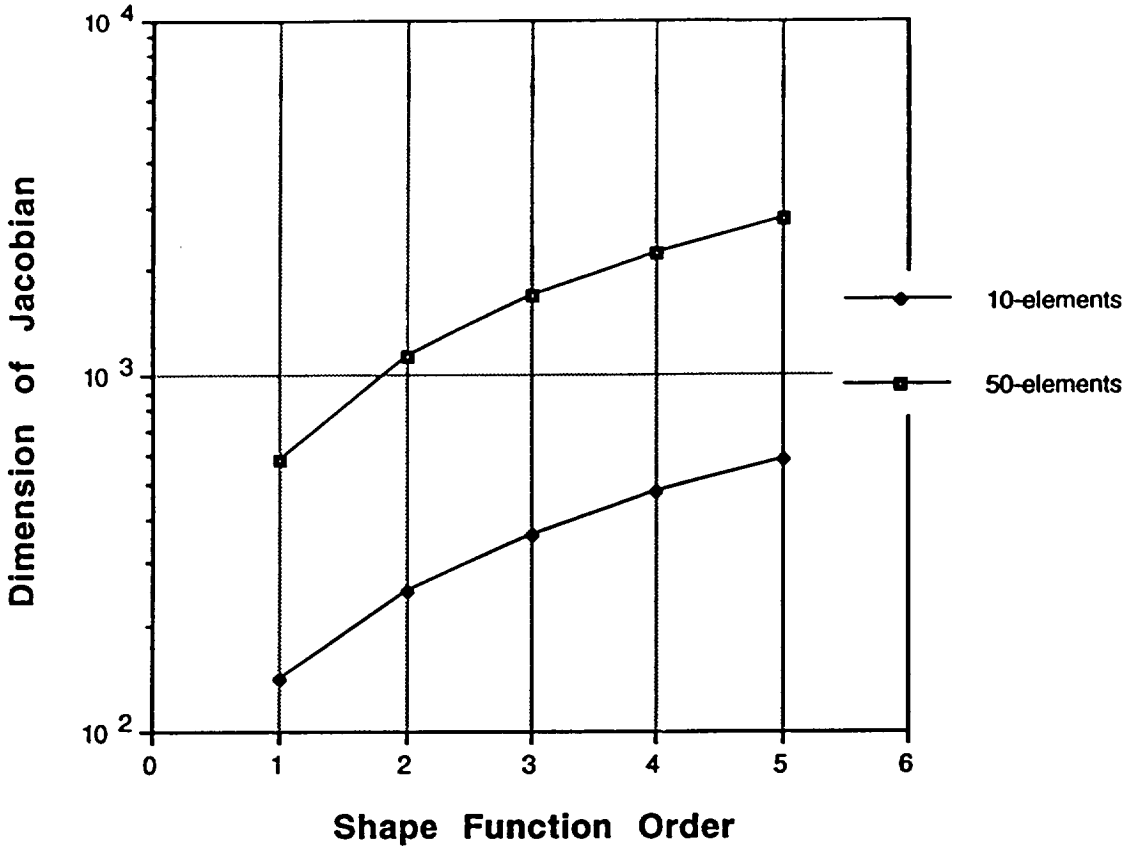

Figure 1. Dimension of Jacobian for Bryson and Ho problem, Sect. 2.4

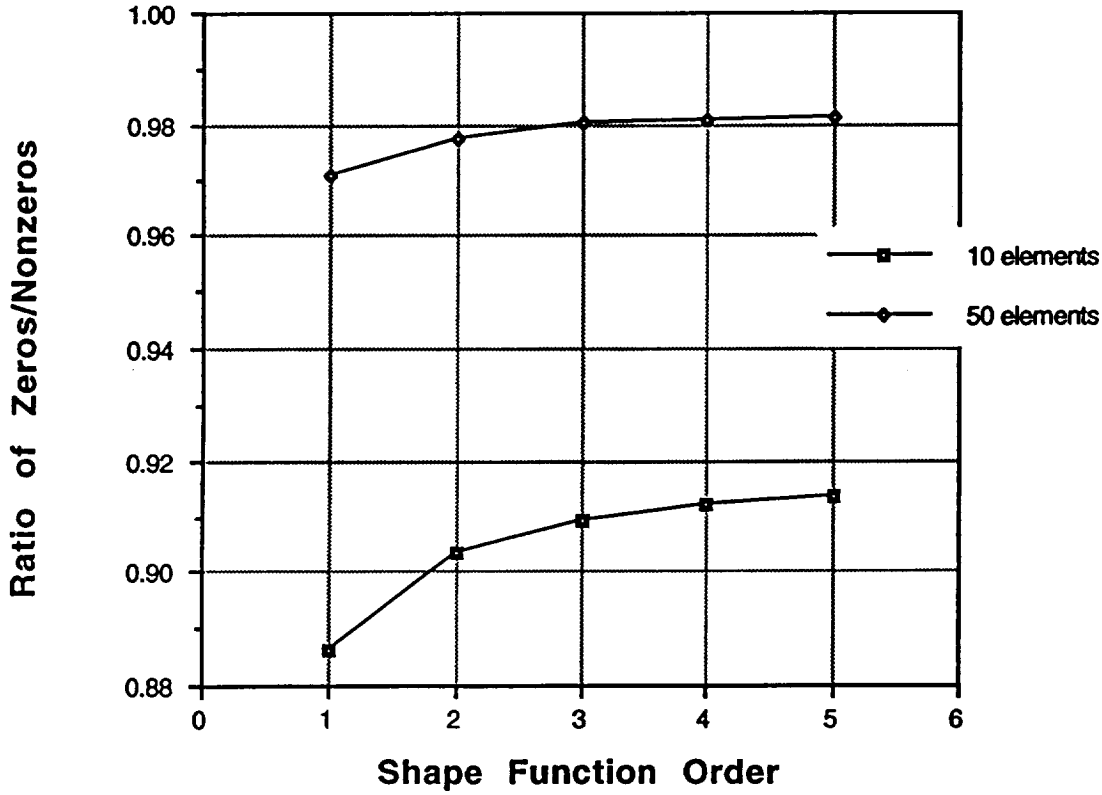

Figure 2. Sparsity of Jacobian for Bryson and Ho problem, Sect. 2.4

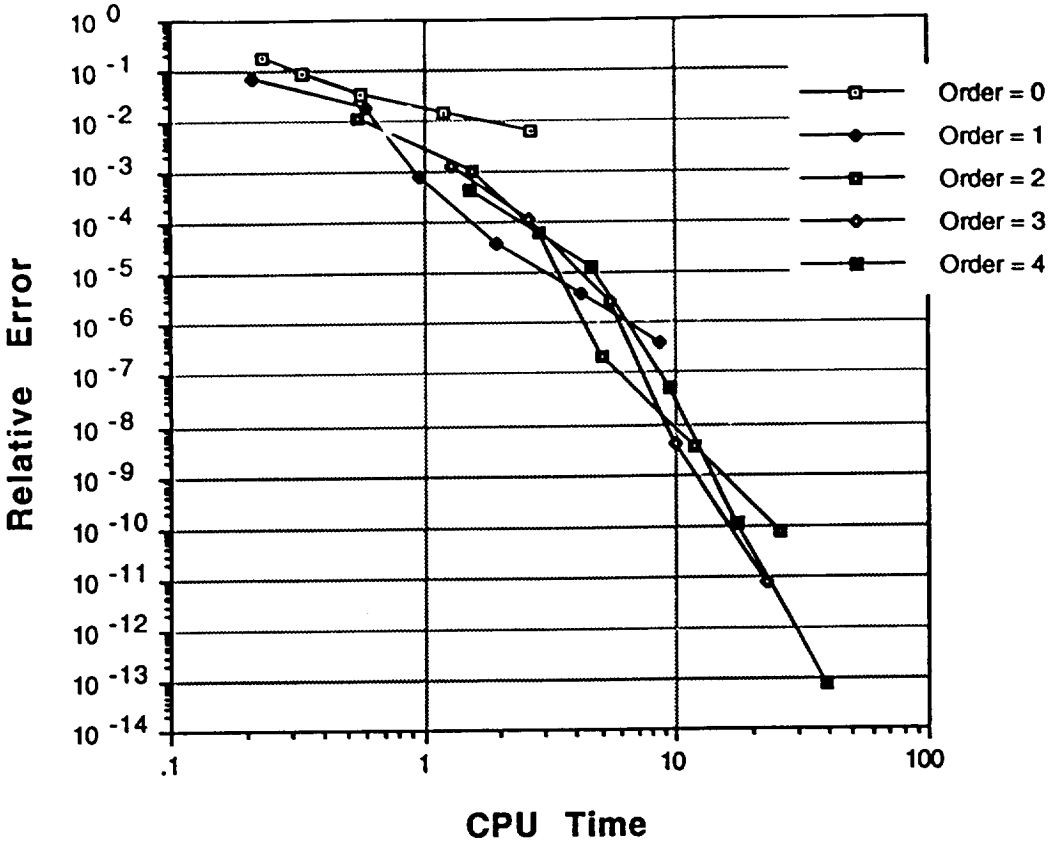

Figure 3. 2-norm of relative error in states for Bryson and Ho problem, Sect. 2.4

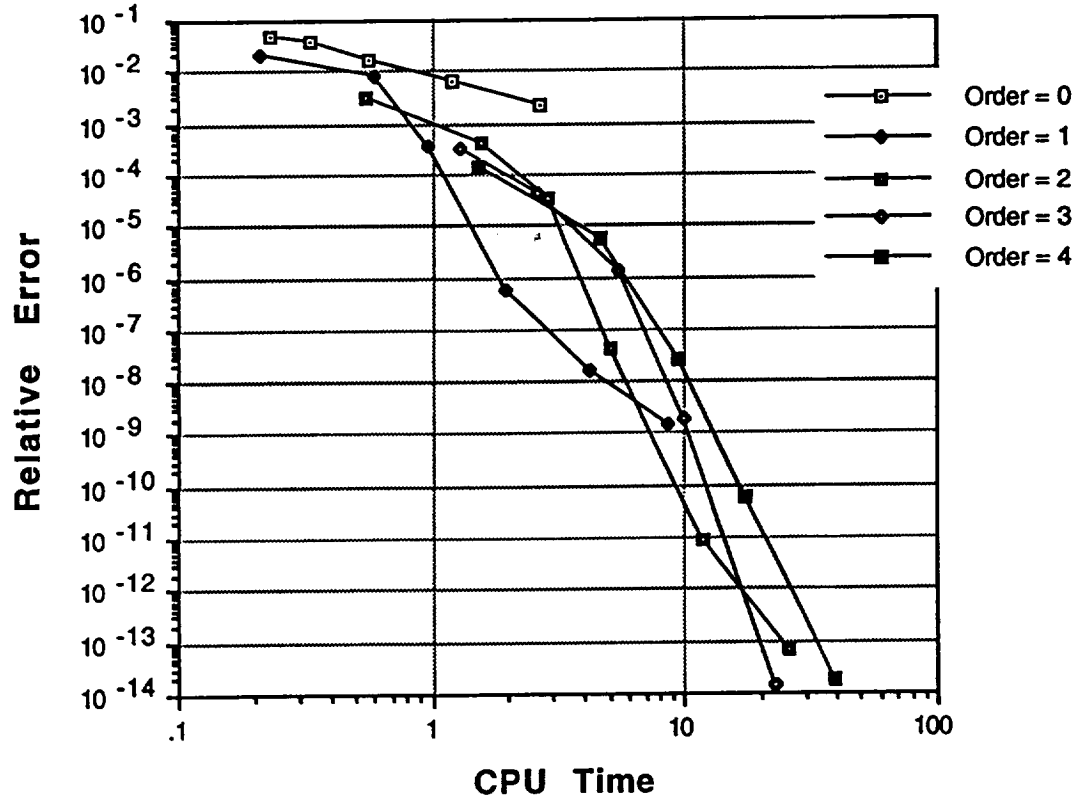

Figure 4. 2-norm of relative error in control for Bryson and Ho problem, Sect. 2.4

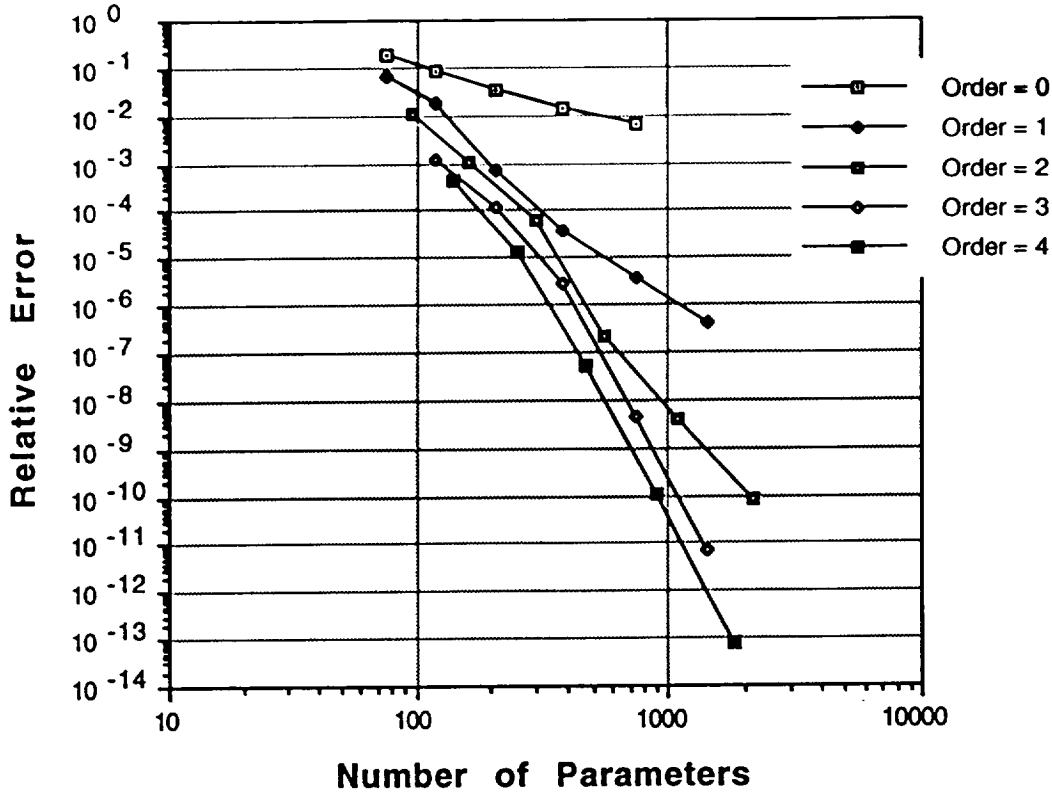

Figure 5. 2-norm of relative error of states for Bryson and Ho problem, Sect. 2.4

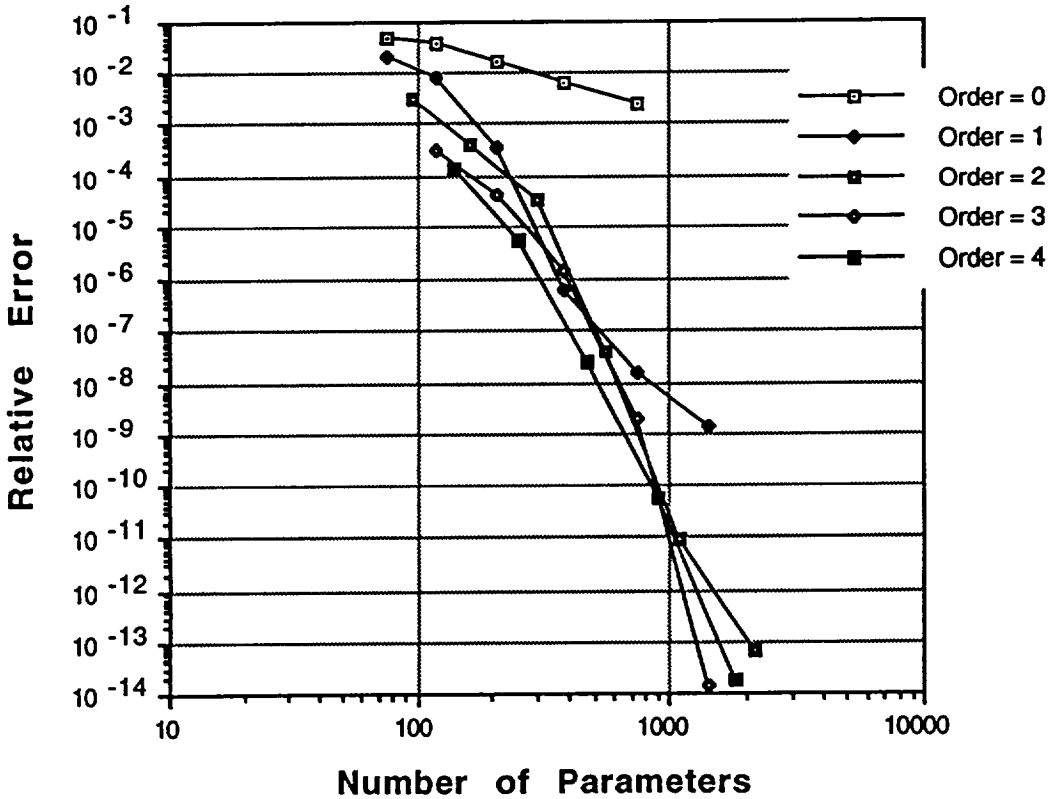

Figure 6. 2-norm of relative error of control for Bryson and Ho problem, Sect. 2.4

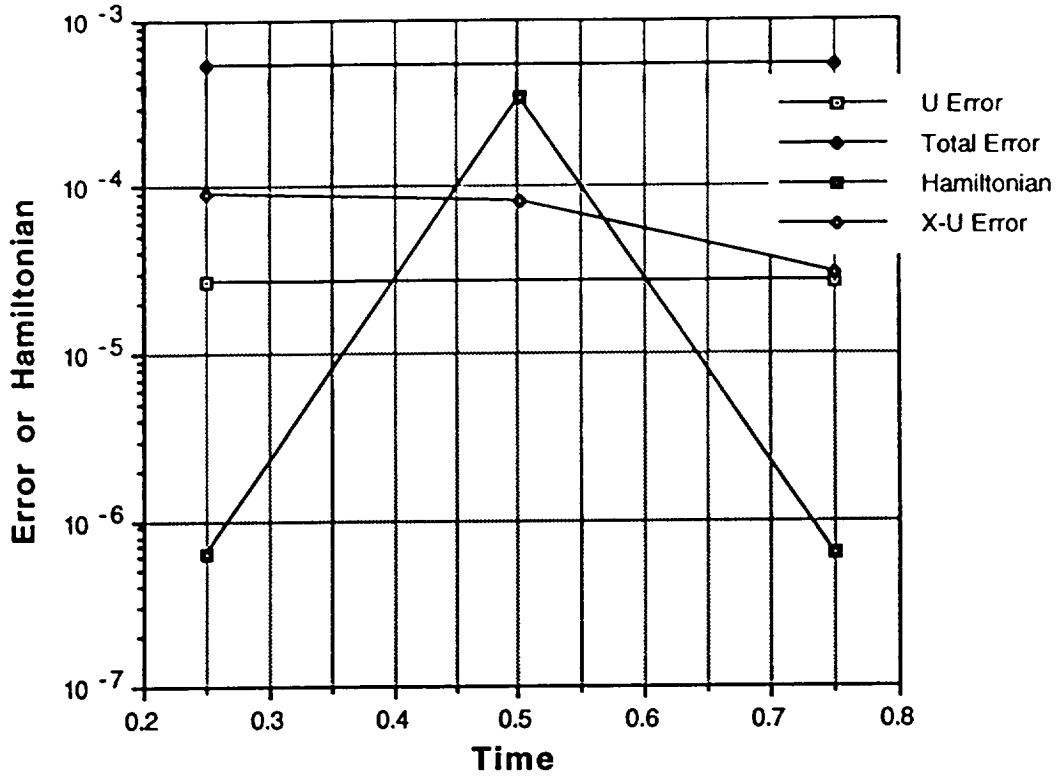

Figure 7. Error Analysis for Bryson and Ho problem Sect. 2.4, 4 elements

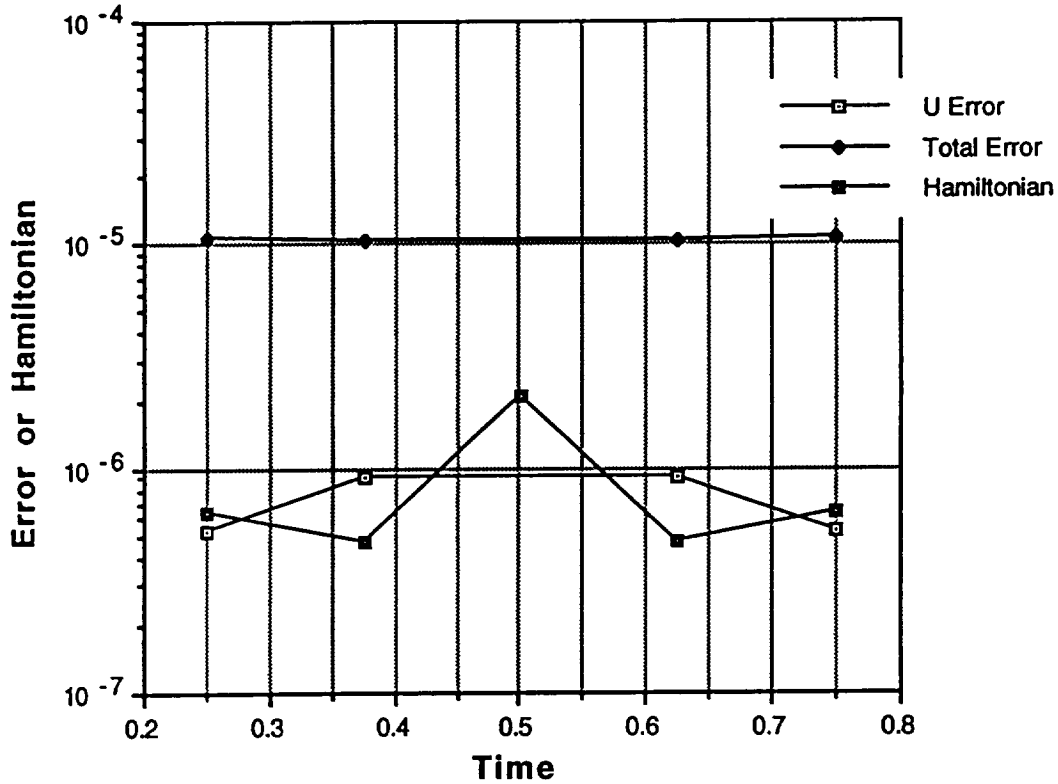

Figure 8. Error Analysis for Bryson and Ho problem, Sect 2.4, 6 elements

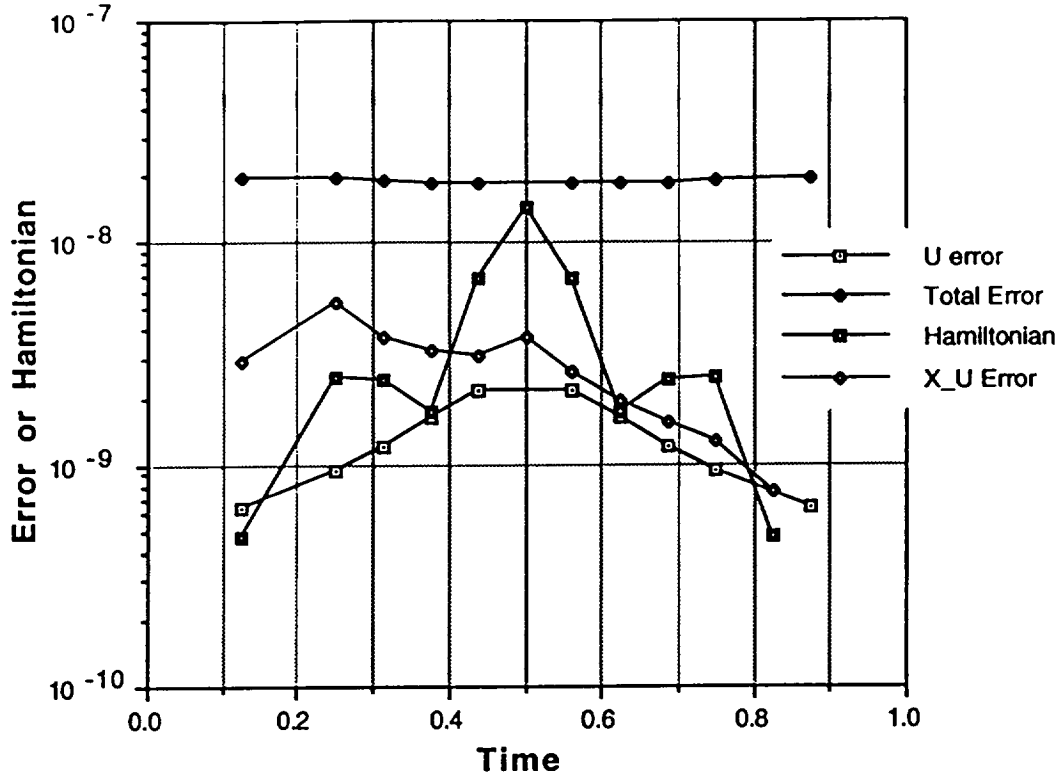

Figure 9. Error Analysis for Bryson and Ho problem, Sect 2.4, 12 elements

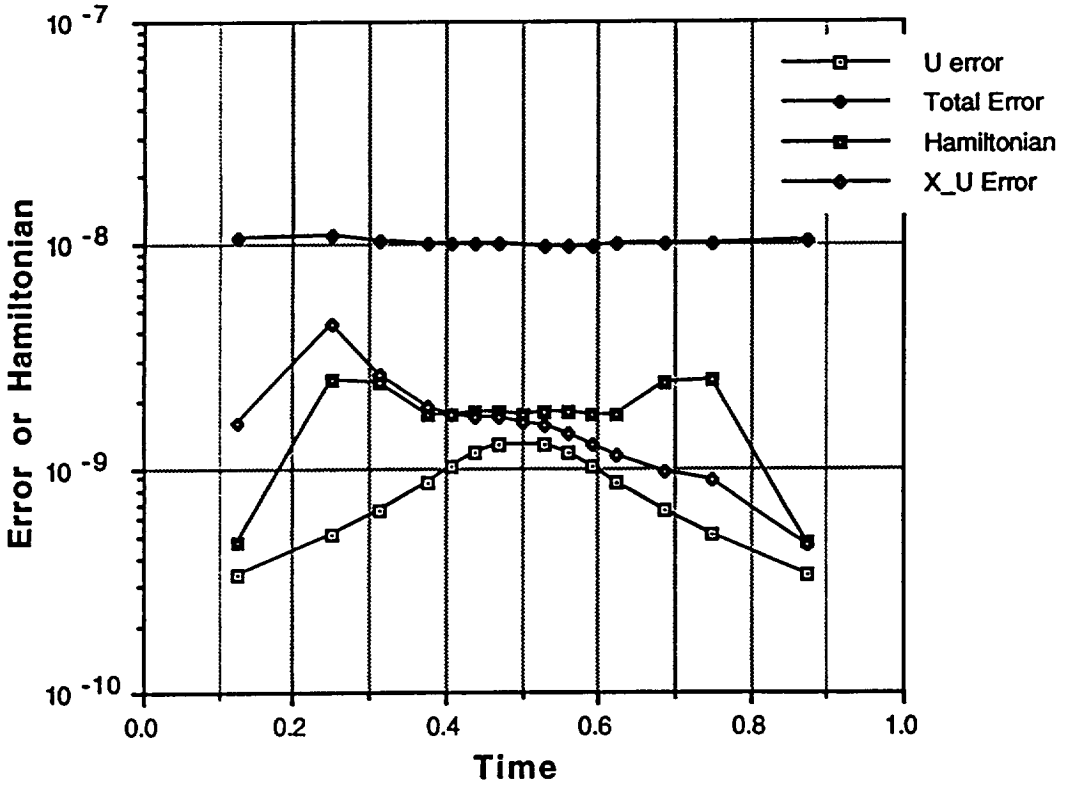

Figure 10. Error Analysis for Bryson and Ho problem, Sect 2.4, 16 elements

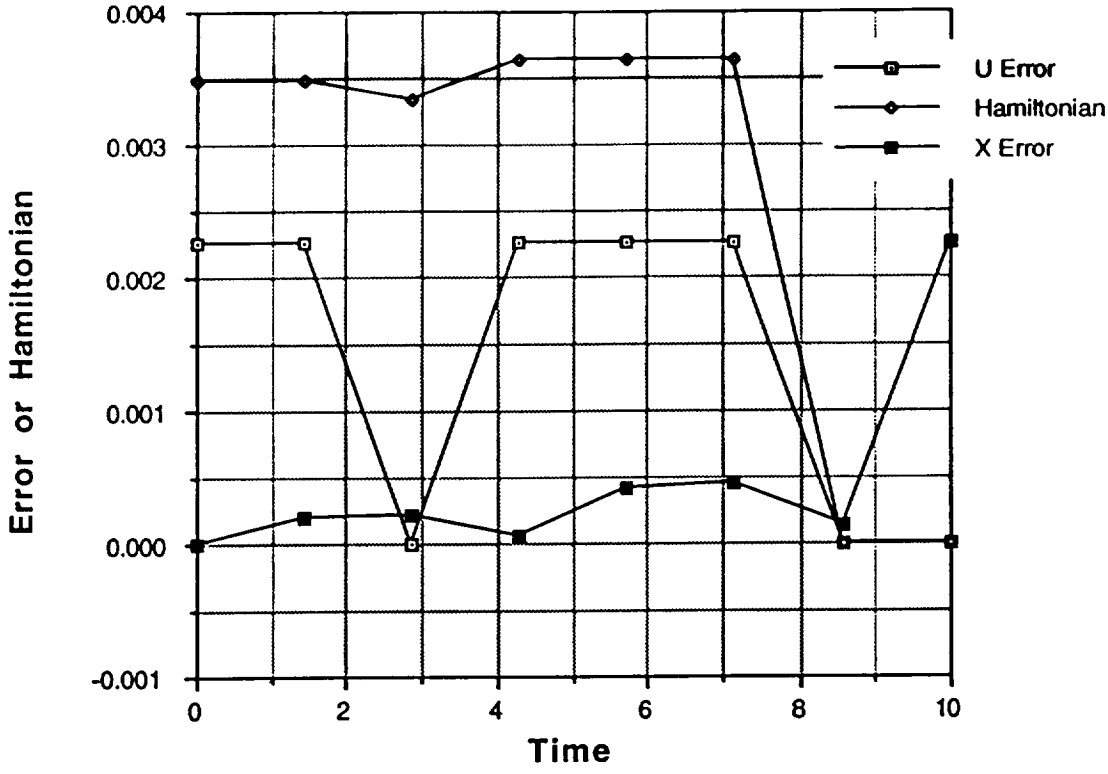

Figure 11. Error Analysis for Bryson and Ho problem, Pg 109, 7 elements

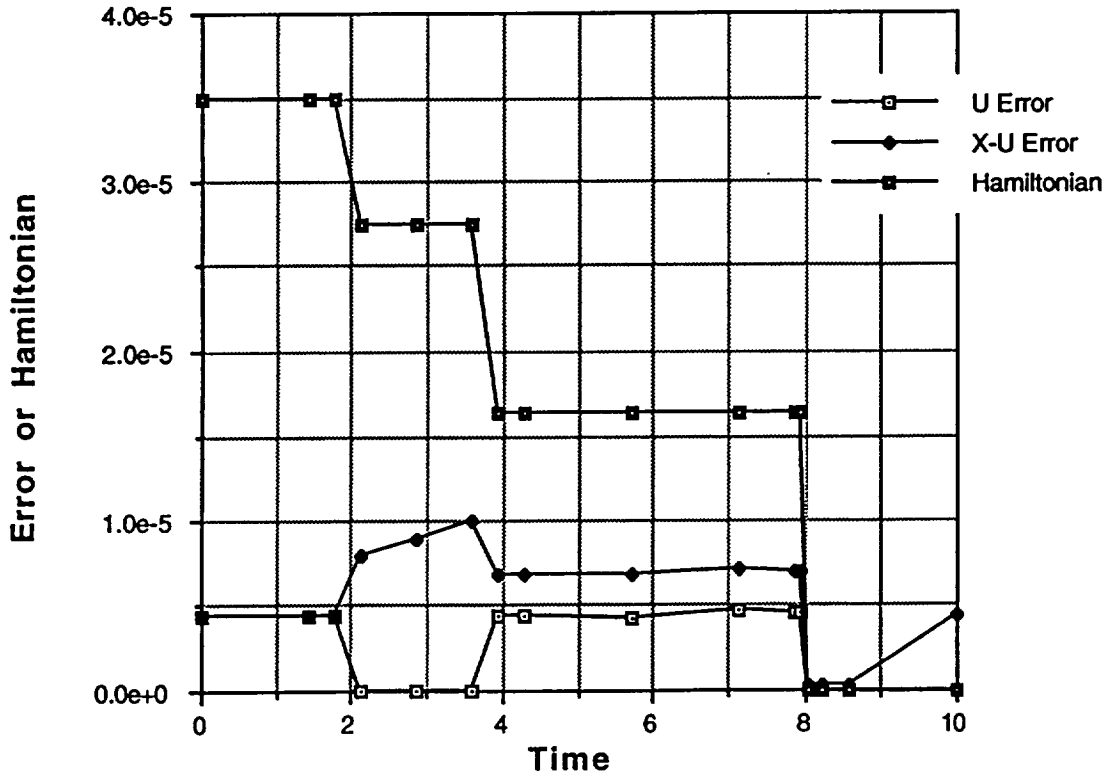

Figure 12. Error Analysis for Bryson and Ho problem Pg. 109, with 16 elements, optimized from 7

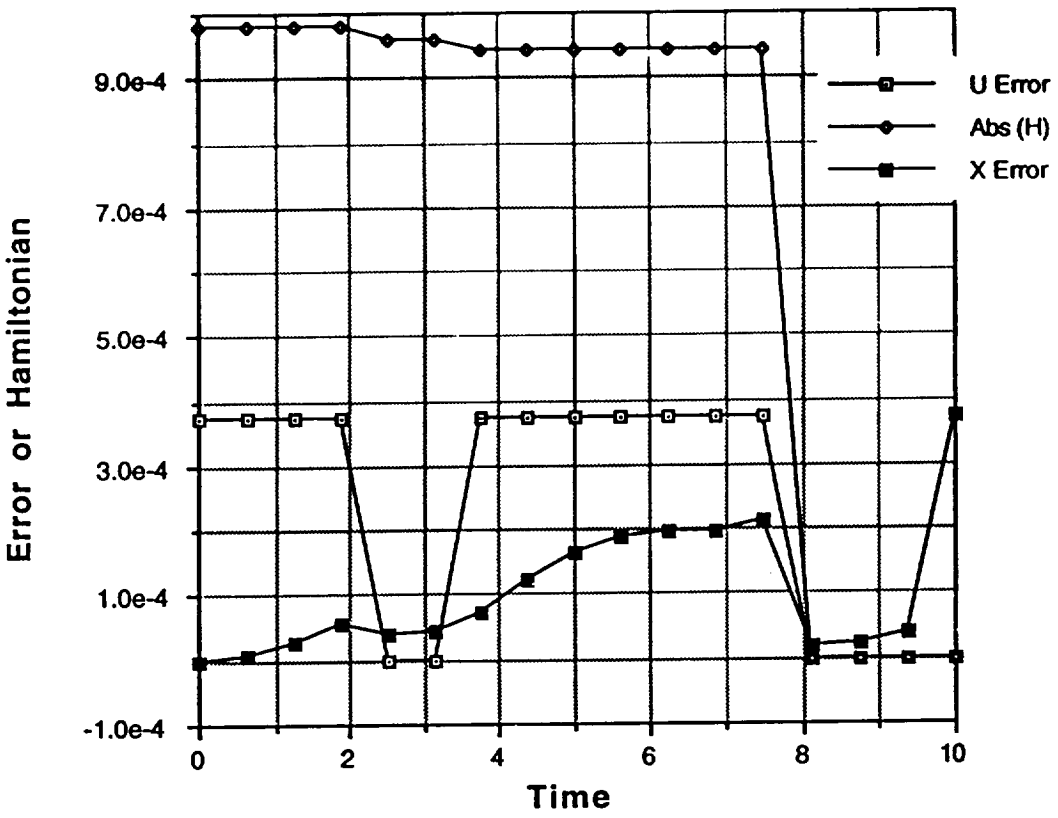

Figure 13. Error Analysis for Bryson and Ho problem, pg 109, with 17 elements, not optimized

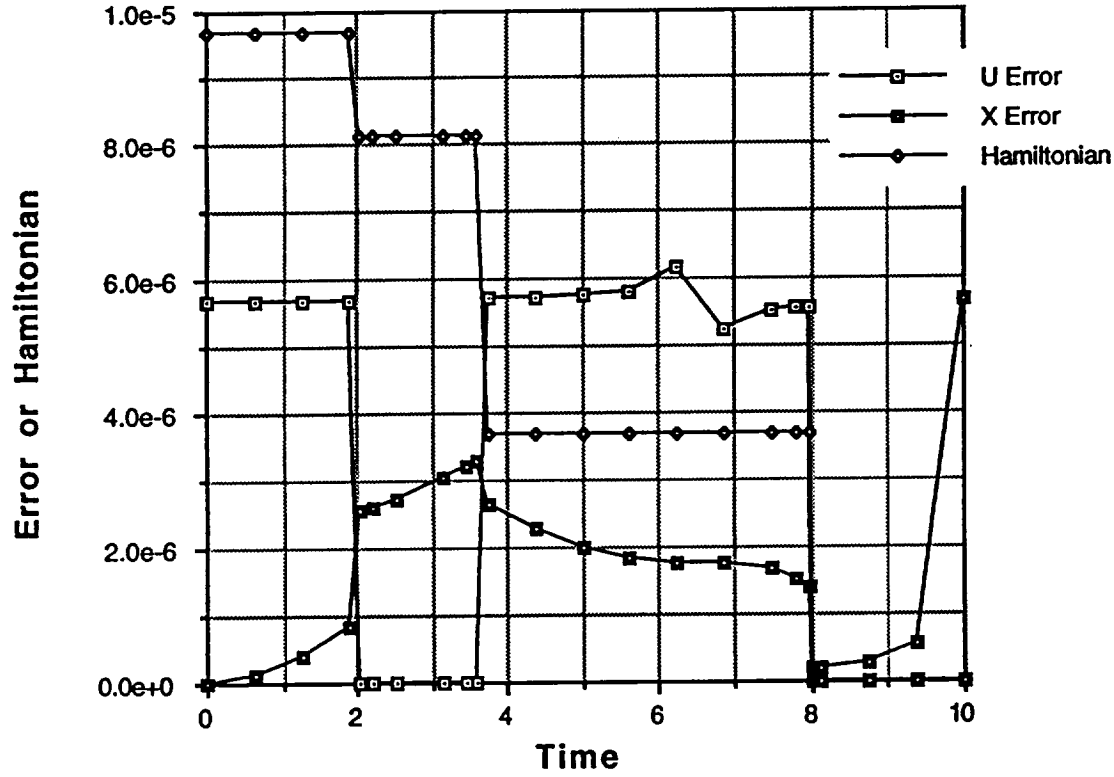

Figure 14: Error Analysis for Bryson and Ho problem, Pg 109, with 24 elements, optimized from 17