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

Final Report
NASA Grant NAG-1-1435
Apr. 22, 1992 – Feb. 21 1993

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

Introduction

The purpose of this research effort was to begin the study of the application of *hp*-version finite elements to the numerical solution of optimal control problems. Under NAG-939, the hybrid MACSYMA/FORTRAN code GENCODE was developed which utilized *h*-version finite elements to successfully approximate solutions to a wide class of optimal control problems. In that code the means for improvement of the solution was the refinement of the time-discretization mesh. With the extension to *hp*-version finite elements, the degrees of freedom include both nodal values and extra interior values associated with the unknown states, co-states, and controls, the number of which depends on the order of the shape functions in each element. For details, see [1].

One possible drawback is the increased computational effort within each element required in implementing *hp*-version finite elements. We are trying to determine whether this computational effort is sufficiently offset by the reduction in the number of time elements used and improved Newton-Raphson convergence so as to be useful in solving optimal control problems in real time. Because certain 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, the scheme may turn out to be especially powerful in a parallel computing environment. A different processor could be assigned to each element. The number of processors, strictly speaking, is not required to be any larger than the number of sub-regions which are free of discontinuities of any kind.

Summary of Completed Work

In order to acquaint Mr. Warner better with the workings of GENCODE and with finite element methods in general, much of the first part of the year was spent in the study and modification of GENCODE. The code as described in [1] handled only two stages, defined here as a time intervals with distinct differential equations. Mesh refinement was also prescribed beforehand such that the code always began with two elements per phase, doubling upon successful convergence until the final desired number discretization was reached.

The modifications were conducted in parallel with a similar effort at NASA Langley Research Center to implement the *h*-version finite elements solution procedure into a MATLAB optimal control problem solver, VTOTS. Since the code was being expanded to include an arbitrary number of stages, the governing equations were rearranged to better take advantage of the sparse matrix solvers included in the Harwell subroutine library. Instead of having separate initial conditions, terminal conditions, and jump conditions at stage breaks, the code was modified such that all boundary conditions, include continuity between stages, were included in a single set of boundary conditions.

The other major advancement of GENCODE was the freedom to specify arbitrarily the desired time discretization in each phase. Once the solution to that discretization converges successfully, that solution can be interpolated in the code to provide initial guesses for any subsequently desired discretization. This flexibility proved most helpful in solving difficult problems.

Once completed, the new GENCODE was tested against VTOTS, with refinements and error correction occurring in both codes as errors were discovered. Since then, VTOTS has been updated to include state constraints with a more automated switching structure, an improvement that we plan to incorporate in the new GENCODE.

After the GENCODE testing was completed, we formulated the structure of using p-version finite elements to solve two-point boundary value problems. A summary of that development is included below. At this stage, a time-marching algorithm has been developed and tested which utilizes p-version elements. Results were obtained for a number of problems, looking at accuracy and computational effort over various length time elements, numbers of Gauss quadrature points, and orders of shape functions. The improvements in accuracy over h-version elements are dramatic, as expected, and the overall results are encouraging enough that we plan to move forward into the development of a two-point boundary problem solver.

Higher-Order Elements

In moving toward eventually applying p-version finite elements to the solution of optimal control problems, we first developed how to use them to the solve nonlinear two-point boundary value problems. In order to test their performance we started solving problems involving given initial states and only one time element (*i.e.*, initial-value or time-marching problems).

Formulation for Optimal Control Problems

The differential equations of interest are assumed to be of the form

$$\dot{x} = f(x) \qquad x \in R^n \tag{1}$$

where f is an autonomous function of the state vector x. The boundary conditions can be specified at the initial time, t_0, the final time t_f, or some combination of both. For this we denote

$$\begin{aligned} x(t_0) &= x_0 \\ x(t_f) &= x_f \end{aligned} \tag{2}$$

For solving optimal control problems, this problem formulation corresponds to the Euler-Lagrange equations when the control can be solved explicitly in terms of the states and costates from the optimality condition. Then the vector x would include both the states and the costates, and the boundary conditions would include the derived costate boundary conditions.

The time interval is broken up into N not necessarily equal length time elements, Δt_i, such that the time at each element boundary $\hat{t}_i$ is calculated as

$$\begin{aligned} \hat{t}_1 &= t_0 \\ \hat{t}_i &= \hat{t}_{i-1} + \Delta t_i \qquad i = 2, \ldots, N+1 \end{aligned} \tag{3}$$

and we define the states at these nodes to be

$$\hat{x}_i = x(\hat{t}_i) \qquad i = 1, \ldots, N+1 \tag{4}$$

Ultimately, only the t_0 and t_f nodal values of the states are of interest as the values at the internal nodes will be known functions of the values of the states within each time interval. The time within the i^{th} element, t_i is expressed as $t_i = \hat{t}_{i-1} + \tau \Delta t_i$ where $0 \leq \tau \leq 1$ and

$$\tau = \frac{t_i - \hat{t}_{i-1}}{\Delta t_i} \tag{5}$$

so that $dt = \Delta t_i d\tau$.

The state equations are enforced through use of Lagrange multipliers $\delta\lambda$ inside the time interval and the boundary conditions are enforced as natural boundary conditions at the endpoints:

$$\int_{t_0}^{t_f} \delta\lambda^T \left[\dot{x} - f(x)\right] dt + \delta\lambda^T \left(\hat{x} - x\right)\Big|_{t_0}^{t_f} = 0 \tag{6}$$

Substituting for the normalized time τ yields

$$\sum_{i=1}^{N} \Delta t_i \int_0^1 \left[\delta\lambda_i^T \dot{x}_i - \delta\lambda_i^T f(x_i)\right] \, d\tau + \delta\lambda^T \left(\hat{x} - x\right)\Big|_{t_0}^{t_f} = 0 \tag{7}$$

where the subscript denotes the element number or node for each variable. Integrating by parts results in

$$\sum_{i=1}^{N} \Delta t_i \int_0^1 \left[\delta\dot{\lambda}_i^T x_i + \delta\lambda_i^T f(x_i)\right] \, d\tau + \delta\lambda_1^T \hat{x}_1 - .\delta\lambda_{N+1}^T \hat{x}_{N+1} = 0 \tag{8}$$

Now from Ref. 2, we define C^0 shape functions for the LaGrange multipliers in each element in terms of nodal values and internal values

$$\begin{aligned} \delta\lambda_i &= \delta\hat{\lambda}_i(1-\tau) + \delta\hat{\lambda}_{i+1}\tau + \delta\bar{\lambda}_{i,1}(1-\tau)\tau\sqrt{6} + \cdots + \delta\bar{\lambda}_{i,m}\alpha_m(\tau) \\ \frac{d\delta\lambda_i}{d\tau} \equiv \delta\lambda_i' &= \Delta t_i \delta\dot{\lambda}_i = -\delta\hat{\lambda}_i + \delta\hat{\lambda}_{i+1} + \delta\bar{\lambda}_{i,1}(1-2\tau)\sqrt{6} + \cdots + \delta\bar{\lambda}_{i,m}\alpha_m'(\tau) \end{aligned} \tag{9}$$

Here m is the order of the shape function, with $m = 0$ corresponding to h-version finite elements. The polynomial function $\alpha_m(\tau)$ is determined through a set of relationships developed in Ref. 2, and $\alpha_m'(\tau)$ is the derivative of $\alpha_m(\tau)$ with respect to τ. Similarly, the shape functions for the values of the states internal to each element are

$$x_i = \bar{x}_{i,1} + \bar{x}_{i,2}\sqrt{6}\left(\tau - \tfrac{1}{2}\right) + \cdots + \bar{x}_{i,m+1}\beta_{m+1}(\tau) \tag{10}$$

Here again the polynomial function $\beta_m(\tau)$ is a recursive relationship developed in [2]. One fewer $\delta\bar{\lambda}$ term is used than $\bar{x}$ terms to ultimately ensure an equal number of equations and unknowns.

Substituting Eqs. (9) and (10) into Eq. (8) and taking $m = 1$ for simplicity results in

$$\begin{aligned}
&\sum_{i}^{N} \int_0^1 \left\{ \left[-\delta\hat{\lambda}_i + \delta\hat{\lambda}_{i+1} + \delta\bar{\lambda}_{i,1}(1-2\tau)\sqrt{6} \right]^T \left[\bar{x}_{i,1} + \bar{x}_{i,2}\sqrt{6}\left(\tau - \tfrac{1}{2}\right) \right] \right. \\
&\left. + \Delta t_i \left[\delta\hat{\lambda}_i(1-\tau) + \delta\hat{\lambda}_{i+1}\tau + \delta\bar{\lambda}_{i,1}(1-\tau)\tau\sqrt{6} \right]^T f\left[\bar{x}_{i,1} + \bar{x}_{i,2}\sqrt{6}\left(\tau - \tfrac{1}{2}\right) \right] \right\} d\tau \\
&+ \delta\hat{\lambda}_1^T \hat{x}_1 - \delta\hat{\lambda}_{N+1}^T \hat{x}_{N+1} = 0
\end{aligned} \tag{11}$$

Integrating all the terms that do not depend on $f(\tau)$ and simplifying results in

$$\begin{aligned}
&\sum_{i}^{N} \Delta t_i \int_0^1 \left[\delta\hat{\lambda}_i(1-\tau) + \delta\hat{\lambda}_{i+1}\tau + \delta\bar{\lambda}_{i,1}(1-\tau)\tau\sqrt{6} \right]^T f\left[\bar{x}_{i,1} + \bar{x}_{i,2}\sqrt{6}\left(\tau - \tfrac{1}{2}\right) \right] d\tau \\
&+ \sum_{i}^{N} \left(-\delta\hat{\lambda}_i^T \bar{x}_{i,1} + \delta\hat{\lambda}_{i+1}^T \bar{x}_{i,1} - \delta\bar{\lambda}_{i,1}^T \bar{x}_{i,2} \right) + \delta\hat{\lambda}_1^T \hat{x}_1 - \delta\hat{\lambda}_{N+1}^T \hat{x}_{N+1} = 0
\end{aligned} \tag{12}$$

Finally, grouping terms multiplying each of the LaGrange multipliers yields

$$\begin{aligned}
&\delta\hat{\lambda}_1^T \left[-\bar{x}_{1,1} + \hat{x}_1 + \Delta t_1 \int_0^1 (1-\tau) f(x_1)\, d\tau \right] \\
&+ \delta\hat{\lambda}_{N+1}^T \left[\bar{x}_{N,1} - \hat{x}_{N+1} + \Delta t_N \int_0^1 \tau f(x_N)\, d\tau \right] \\
&+ \sum_{i=2}^{N} \delta\hat{\lambda}_i^T \left[-\bar{x}_{i,1} + \bar{x}_{i-1,1} + \Delta t_i \int_0^1 (1-\tau) f(x_i)\, d\tau + \Delta t_{i-1} \int_0^1 \tau f(x_{i-1})\, d\tau \right] \\
&+ \sum_{i=1}^{N} \delta\bar{\lambda}_{i,1}^T \left[-\bar{x}_{i,2} + \Delta t_i \int_0^1 \sqrt{6}(1-\tau)\tau f(x_i)\, d\tau \right] = 0
\end{aligned} \tag{13}$$

If n is the dimension of the state vector, then Eq. (13) results in $(2N+1)n$ equations for $(2N+2)n$ unknowns: n unknowns at each end point and $2n$ unknowns within each element. If the order of the shape function m is greater than one, then the last block of Eq. (13) becomes nm equations, and x_i would then depend on $\bar{x}_{i,j}$ for $j = 1, \ldots, m+1$. This brings the total to $(N+1+Nm)n$ equations and $(N(m+1)+2)n$ unknowns. Thus for a unique solution to the problem, a combination of n initial and final conditions on the states must be specified. Note that when solving these equations by a Newton-Raphson approach that the Jacobian matrix is block diagonal.

Note that if the order of the shape function m is zero, then $x_i = \bar{x}_{i,1}$ the h-version equations result

$$\begin{aligned} &\delta\hat{\lambda}_1^T \left[-\bar{x}_1 + \hat{x}_1 + \Delta t_1 f(x_i)/2\right] \\ &+\delta\hat{\lambda}_{N+1}^T \left[\bar{x}_N - \hat{x}_{N+1} + \Delta t_N f(x_N)/2\right] \\ &+\sum_{i=2}^{N} \delta\lambda_i^T \left[-\bar{x}_i + \bar{x}_{i-1} + \Delta t_i f(x_i)/2 + \Delta t_{i-1} f(x_{i-1})/2\right] = 0 \end{aligned} \tag{14}$$

Time-Marching Algorithm

To help determine the feasibility of using p-version finite elements to solve optimal control problems, Eqs. (13) were first incorporated into a time-marching algorithm. In this case, the required number of specified boundary conditions are given as initial conditions, $\hat{x}_1 = x(t_0)$. Eqs. (13) then reduce to the case of only one element $N = i = 1$, and we can drop the element subscripts. Thus, the remaining subscripts refer only to the element order. Rewriting Eqs. (13) in this way one obtains

$$\begin{aligned} \hat{x}_1 &= \bar{x}_1 - \Delta t \int_0^1 (1-\tau) f\left[\bar{x}_1 + \bar{x}_2\sqrt{6}\left(\tau - \tfrac{1}{2}\right)\right] d\tau \\ 0 &= -\bar{x}_2 + \Delta t\sqrt{6} \int_0^1 (1-\tau)\tau f\left[\bar{x}_1 + \bar{x}_2\sqrt{6}\left(\tau - \tfrac{1}{2}\right)\right] d\tau \\ \hat{x}_2 &= \bar{x}_1 + \Delta t \int_0^1 \tau f\left[\bar{x}_1 + \bar{x}_2\sqrt{6}\left(\tau - \tfrac{1}{2}\right)\right] d\tau \end{aligned} \tag{15}$$

again assuming first-order shape functions. For each additional order, there is one more equation and one more unknown.

For shape functions of arbitrary order, the first two of Eqs. (15) become $m+1$ equations, and they are solved for the internal values, $\bar{x}_i$ for $i = 1, 2, \ldots, m+1$. These values are then used to calculate the nodal values, $\hat{x}_2$ from the third of Eqs. (15). These final values become the initial values for the next element, and the process repeats over all the elements. Notice that in the solution of this problem with a Newton-Raphson algorithm, the Jacobian matrix is, in general, fully populated.

The process of solving these equations has been implemented in a FORTRAN subroutine using a standard full-step Newton-Rapheson algorithm. The routine utilizes the Harwell family of subroutines for solving fully-populated linear systems. A user interface was developed using the symbolic manipuation software MACSYMA which generates the appropriate expressions for error evaluation and the Jacobian matrix. Unlike for h-version finite elements, the quadratures in Eq. (15) in general cannot be done be done by inspection and similarly are too complicated for software such as MACSYMA. To perform the integrations, Gaussian quadrature with a variable number of Gauss points was implemented here.

Example Problem Formulations

The code was run on a variety of problems for varying orders of shape functions, numbers of elements, and numbers of Gauss points. The goal was to determine whether such a code could run in a reasonable length of time for complicated problems, thereby determining the feasiblility of applying p-version elements to two-point boundary value problems. That being established, the next step was to determine the relationship between the number of Gauss points and the accuracy of the solution, in hopes of finding a formula such as in the case of h-version elements. The results of this experimenting will be described for a two-state linear system, two one-state systems with different types of nonlinear dynamics, and an 8-state missile system.

Linear System

The first problem that was examined was an idealized spring-mass system, with a spring constant, k and mass, m.

$$m\ddot{x} = -kx$$

with

$$\begin{aligned} x(0) &= 1.0 \\ \dot{x}(0) &= 1.0 \end{aligned} \tag{16}$$

The analytical solution to this with $k = m = 1$ is

$$\begin{aligned} x(t) &= \cos(t) \\ \dot{x}(t) &= -\sin(t) \end{aligned} \quad \rightarrow \quad \begin{aligned} x(1) &= 0.543087528 \\ \dot{x}(1) &= -0.839676091 \end{aligned} \tag{17}$$

In Fig. 1, the log of the percentage error in the time-marching algorithm at the final time is plotted versus the number of Gauss points for various orders of shape functions. The number of elements in all cases was 5. The percentage error, E, was calculated in terms of the exact answer from Eq. (17) as

$$E(1) = \frac{100}{n} \sum_{i=1}^{n} \frac{y_i(1) - \hat{x}_N}{y_i(1)} \tag{18}$$

where N is the number of elements, and n is the dimension of the state vector $y = \{x, \dot{x}\}$. In subsequent examples, the error is calculated similarly with y again being the appropriate state vector. This error criterion gives a good idea of how errors are being propagating through the algorithm.

Notice that in this case the error is minimized when the number of gauss points is one more than the order of the shape function in each element. This will hold true for all linear systems because the polynomial in τ to be integrated is of no higher order than $2m + 1$, where m is the order of the shape function. As shown in [3], using n Gauss quadrature points and weights results in exact integration of polynomials up to order $2n - 1$. Thus

only $m+1$ Gauss points are needed to exactly evaluate the integrals when the order of the shape functions is m. Unfortunately this result does not hold for nonlinear systems.

Changing the number of elements adjusted the error appropriately in each case, but did not change the relative characteristics between the orders of the shape functions. Also, the number of Newton steps required was only one for all elements sizes, orders of shape function, and numbers of Gauss points. Thus for a linear system, the tradeoffs between accuracy and computational effort are very clear.

Nonlinear System with Quadratic Nonlinearities

The next problem complicates matters some by replacing the linear term with a quadratic in a first order differential equation

$$\begin{aligned} \dot{x} &= ax^2 \\ x(0) &= 1.0 \end{aligned} \tag{19}$$

where a is a constant. The analytic solution to this problem with $a = 0.5$ is

$$x(t) = \frac{-1}{(0.5t-1)} \rightarrow x(1) = 2.0 \tag{20}$$

In Fig. 2, the log of the percentage error at the final time from Eq. (18) is again plotted versus the number of gauss points for various orders of shape functions. The results plotted were obtained with 5 elements, but the trends are the same for any number. In this case the error is again minimized for the number of Gauss points equals the order of the shape function plus one. What is different here is that the accuracy actually goes down as more Gauss points are added, enforcing the point described in [3] that more Gauss points does not always mean higher accuracy.

The number of Newton steps per element decreased on average as the number of elements increased, but only from 6 to 5. And there was no significant decrease in the number of Newton steps required as the order of the element increased. Thus the high accuracy solutions still require considerably more computational effort.

Nonlinear System with Infinite Order Nonlinearities

The next problem examined replaced the quadratic nonlinearity with an exponential function, an infinite-order nonlinearity.

$$\begin{aligned} \dot{x} &= e^x \\ x(0) &= -1 \end{aligned} \tag{21}$$

The analytic solution to this problem is

$$x(t) = -\ln(-t+e) \rightarrow x(1) = -\ln(e-1) \tag{22}$$

In Fig. 3, the log of the percentage error at the final time from Eq. (18) is again plotted versus the number of Gauss points for various orders of shape functions. The figure correspond to 5 elements used, but the trends are the same for any number. Here the line begins to blur about the optimal number of Gauss points. In all cases, the error is very close to the minimum with one more Gauss point than the order of the shape function, but depending on the order and the number of elements, one additional Gauss point can still improve accuracy significantly.

It is encouraging, however, that the number of Newton steps here did decrease with increased order, though only from 6 to 5 on average for each element. In this problem, extra iterations are rather insignificant, as the problem has only one state. Also note that in none of the above examples were there any cases in which the algorithm failed to converge.

Nonlinear Missile Model

To give the time-marching algorithm a harder test, we next tested it on an 8-state missile system. While the equations themselves are unimportant, it should be noted that all the nonlinear system dynamics were retained, including table-lookups on highly nonlinear lift and drag coefficients. The missile model was integrated from launch for 5 s, when the engines would be throttled back. The two controls are angle of attack and roll angle, both of which were set nominally to 1° throughout the time interval.

To provide a comparison, the model was integrated using the Runge-Kutta method and 1000 time steps. Fig. 4 plots the error in the finite-element algorithm as calculated in Eq. (18) using the shooting results as the exact answer. With any fewer than 5 time elements, the algorithm had difficulty converging. A restricted-step Newton method [4] was then implemented, which alleviated some of the convergence difficulties while not appreciably affecting accuracy.

Unfortunately, none of the trends we encountered before with regard to optimal number of Gauss points held true in this case. It is clear that for "real life" problems, the optimal number of Gauss points may have to be determined on a case-by-case basis. Therefore, any future algorithms will include capability for adjusting this number. Also, increasing the order once again brought down the number of Newton steps required for convergence.

Future work

The improvement in accuracy achievable by increasing the order of the shape functions in all four problems presented here is exceptional. The question still remains, however, as to how the extra computational effort will trade off with this increased accuracy when the methodology is extended to two-point boundary value problems. Another open question is how to find a simple means for obtaining the optimal number of Gauss points for more complicated problems.

Given the encouraging results we have seen using *h*-version finite elements so far, we plan to extend GENCODE to include the *hp*-version finite elements and begin testing it on

optimal control problems, including those with state constraints. There are also a number of theoretical aspects of these finite elements yet to be explored including the effect on element stability and convergence properties.

References

[1] Bless, Robert R.: "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] Hodges, Dewey H.; and Hou, Lin-Jun: "Shape Functions for Mixed p-version Finite Elements in the Time Domain." *J. Sound and Vibration*, vol. 145, no. 2, Mar. 8, 1991, pp. 169 – 178.

[3] Press, W. H., Flannery B. P., Teukolsky, S. A., and Vetterling, W. T., *Numerical Recipes*, Cambridge University Press, Cambridge, U.K., 1986.

[4] Achar, N. S., "Trim Analysis by Shooting and Finite Elements and Floquet Eigenanalysis by QR and Subspace Iterations in Helicopter Dynamics," Ph.D. Dissertation, Florida Atlantic University, Boca Raton, Florida, 1992.

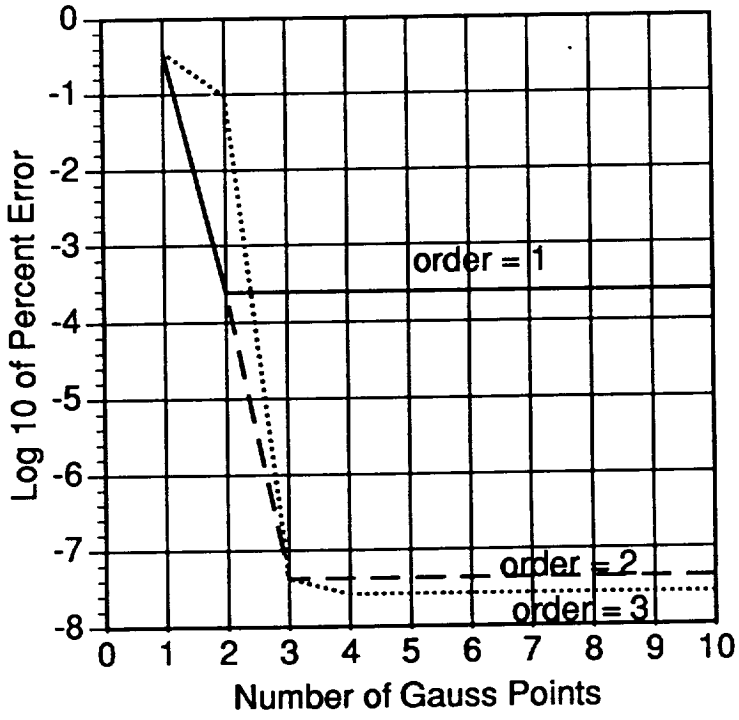

Fig. 1: Time-Marching Error at Final Time for Spring Mass System

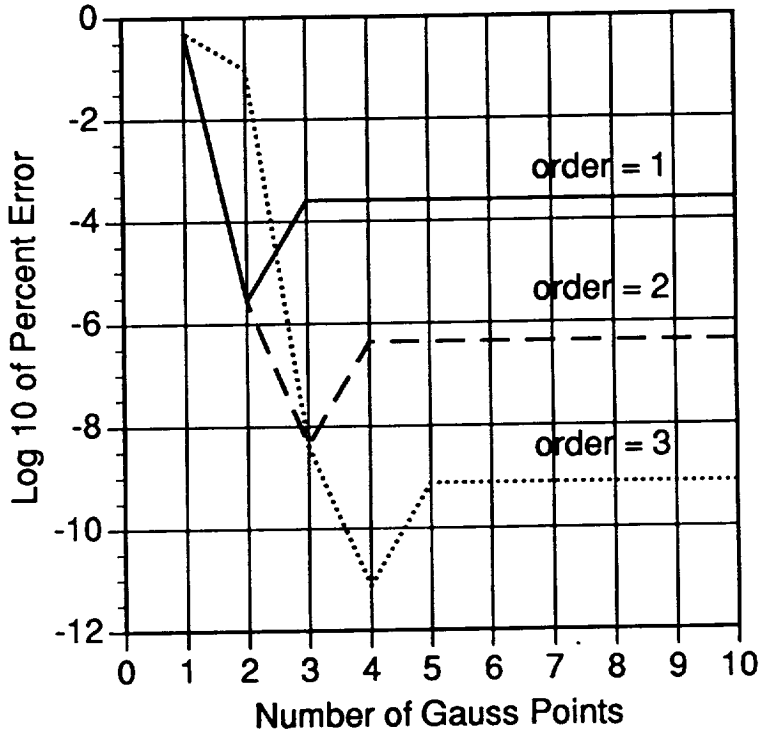

Fig. 2: Time-Marching Error at Final Time for $\dot{x} = x^2$

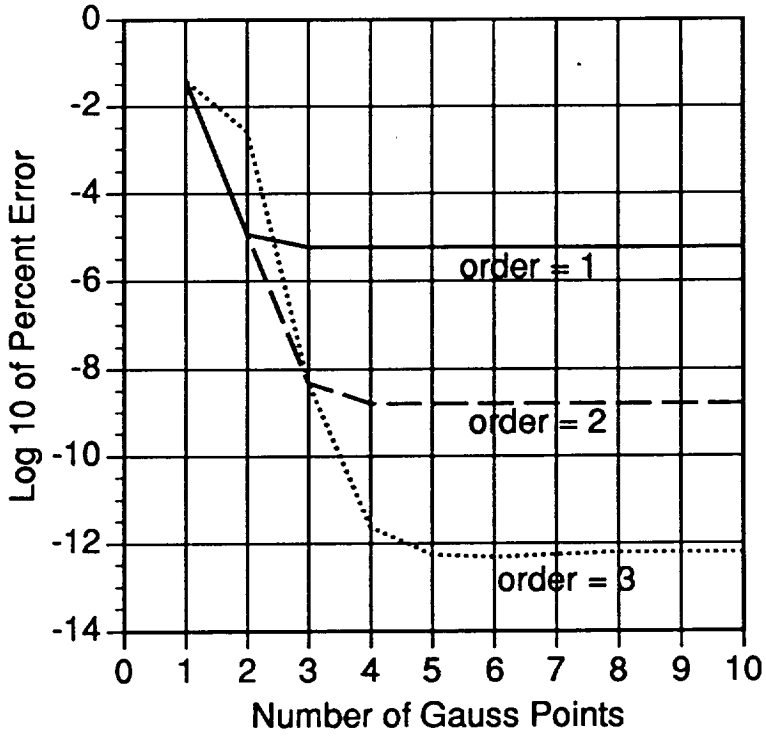

Fig. 3: Time-Marching Error at Final Time for $\dot{x} = e^x$

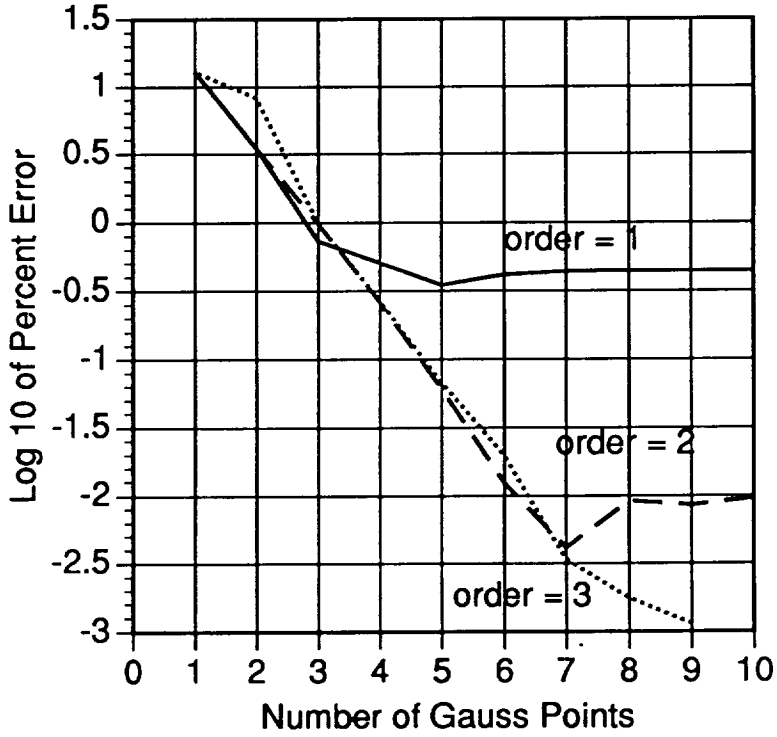

Fig. 4: Time-Marching Error at Final Time for Missile system