UIUCDCS-R-80-1000

**AUTOMATIC MULTIRATE METHODS FOR ORDINARY
DIFFERENTIAL EQUATIONS**

by

C. W. Gear

DISCLAIMER

This book was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

January 1980

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

Supported in part by the U.S. Department of Energy, US ENERGY/EY-76-S-02-2383.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED *gd*

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

1. INTRODUCTION

A multirate method for integrating ordinary differential equations is one in which different equations are integrated using different time steps. The motivation for such an approach is obvious--the desire to reduce the integration time. Consequently, the idea has been mentioned many times and probably used on an ad hoc basis in a number of applications, particularly in real-time applications where speed is critical. However, there has been relatively little study of the problems involved in efficient, automatic, multirate integration. In this paper we will first look at the types of equations that can be handled efficiently by multirate methods. Next we will take a brief look at the theory of multirate methods. We will see that there are no complications in the conventional theory of convergence, and that the expected happens. Then we will look at the problems of automatic multirate methods. Here we encounter a number of difficulties in the estimation of errors and the control of stepsizes. The latter occurs because it may not be possible to estimate the error in a particular step until after a number of other steps have been taken, by which time it may be very costly to "back-up" the integration to redo the earlier step. Three different techniques which have been studied will be discussed.

2. APPLICATION OF MULTISTEP METHODS

One area in which multirate methods have been considered is simulation {1}{2}, although there is little documentation on the subject. Some work has been done on partitioning systems into subsystems for treatment by different methods, as in {3} where the nonlinear part is integrated by a conventional Runge-Kutta method and the linear part is handled by methods that use precomputed matrix exponentials. In our case, we are interested in partitioning based on the relative speeds of subsystems. In such applications, the engineer is often able to use a large amount of "engineering intuition" in the design of an integrator. For example, consider the simulation of an aircraft. It consists of a number of subsystems. For simplicity we will consider two shown in fig. 1.

Fig. 1. Simplified simulation model

The control subsystem reacts very rapidly (being electronic), whereas the flight dynamic subsystem reacts relatively slowly, being a mechanical change. Consequently, a simulation system tailored to this model might choose to integrate the control with small stepsizes, and the dynamics with large stepsizes. The rationale sometimes heard for this is that "because the dynamics are slow, we can break the feedback loop from the dynamics to the control and handle each subsystem separately." "Breaking the feedback loop" refers to assuming that there is no change in the output of the slow system over one of its steps while the fast system is being integrated over several smaller steps. What is actually happening? Suppose the outputs of the fast and slow systems are designated as y and z respectively, and they satisfy the differential equations

$$\begin{aligned} y' &= f(y, z, t) \\ z' &= g(y, z, t) \end{aligned} \tag{1}$$

(These could be two systems of equations of different dimensions.) Let us suppose that the stepsizes used for the y and z integrations are h and Mh respectively, where M is an integer, and that the values of y and z are known for time value t_{Mn} , where $t_k = t_0 + kh$. The process consists of integrating y over M small steps of size h while z is held constant at its value of $z(t_{Mn})$. Then z is integrated over one step of size Mh so that we have information at time $t_{M(n+1)}$. Fig. 2 shows a possible response of the numerical system where $f(y, z, t) = -100(y-z)$. The y values change rapidly after z has been updated, then settle to a new "stable" state based on the value of z at the beginning of the group of M small steps. In fact, we would expect that the true value of y at the end of the set of M small steps should be determined by the value of z at the end, and this suggests that it might be better to perform the integration of z over its big step before doing the M steps of y . This would correspond to "breaking the feedback loop" between the control and dynamic subsystems; intuitively less appealing but possibly more accurate. At first glance it might appear that

this simply transfers the problem to z which now will be integrated over one step Mh based on the value of y at the start of that step. This is not a problem if the integration method used for z is explicit because the value of $g(y,z,t)$ is needed only from the beginning of the step, but there is a more important reason why it is not a problem. If the behavior of y is fast but of z is slow, there can be very little coupling from y to z . That is to say, the dependence of $g(y,z,t)$ on y must be very small. Hence the link from y to z is a natural place to "break the feedback loop." This point will be important in later discussions.

Fig. 2. Integration after "Breaking the Feedback Loop"

Wherever the feedback loop is broken, a constant value of one variable is being used in the integration of the other variable over an interval of length Mh . This "approximation" by a constant function will introduce an error of size $O(Mh)$, and its integration over an interval of size Mh contributes a local error of $O(Mh)^2$ which causes a global error of size $O(Mh)$. The natural thing to do is to interpolate or extrapolate the value of z (or y) while integrating y (or z) thus reducing the approximation error. If the extrapolation is done by a polynomial of degree $p-1$ it introduces an extrapolation error of $O(Mh)^p$. In that case the global error introduced will also be $O(Mh)^p$, which would be a natural ally of an order p

integration method. Section 4 will sketch the proof of the result being suggested here, namely that if stable order p methods are used for integration and order $p-1$ methods are used for extrapolation or interpolation, the method is order p convergent.

Another area in which multirate methods are being considered, although under another name, is in evolutionary partial differential equations (PDEs) where it is thought of as a form of local mesh refinement in time. Such methods will probably be necessary if large two- and three-dimensional time-dependent PDEs are to be integrated in the presence of shock-like fronts because the size of the time step must be sharply restricted in the front but not in regions of low activity. In this paper we consider the general problem of systems of ordinary differential equations which could arise from a discrete system being simulated or by the spatial discretization of PDEs. In the latter case, we should recognize that the system of equations may change from time to time as the spatial mesh is adjusted to adapt to the needs of the problem, but these particular questions will not be considered here.

3. SPEED CONSIDERATIONS

Multirate methods appear attractive because they should use less computer time. The reasoning is that the total time is roughly proportional to the number of integration steps taken for each equation. Since a conventional method must use a number of steps equal to the number of steps for the fastest component times the number of equations, a multirate method can reduce the number of steps for the slow components. However, multirate programs can be more complex than straightforward methods, so we must be certain that increased program overhead does not override any savings. The computational cost of a numerical integration program is due to several factors. They are:

- Evaluations of derivatives

- Application of integration formulas

- Interpolations/extrapolations if a multirate method is used

- Estimating errors

- Logic for automatic step control

Repeated steps when a failure occurs and completed steps must be discarded

Solution of implicit equations if they are used

In this section we will examine the relation between the first three of these costs for the pair of eq. (1) when they are integrated by fixed-stepsize methods.

Let us assume that the same method is used for each equation over stepsizes of h and Mh , and that it requires q function evaluations per step. We will consider the cost of one compound step of size Mh . Suppose that c_f , c_g , c_x , and c_i are the costs of an evaluation of f , an evaluation of g , an extrapolation of z , and an integration step, respectively. Then the cost of one compound step is

$$C_M = (qc_f + c_x)M + qc_g + (M + 1)c_i \quad (2)$$

This assumes that only one interpolation/extrapolation is needed for a z value in each integration step for y , as would be the case in a multistep method. If intermediate values are used, as in a Runge Kutta method, the cost must be increased by $M(q-1)c_x$. This must be compared with the cost of using stepsize h for both components over M steps, which is

$$C = (c_f + c_g)Mq + 2Mc_i \quad (3)$$

because no extrapolations will be necessary. The difference is

$$C - C_M = (qc_g + c_i)(M - 1) - M[q]c_x \quad (4)$$

where the $[q]$ term appears if a Runge-Kutta-like method is used. In Runge-Kutta methods, c_i is small and q is large (4 to 13, depending on the order of the method). In this case we can approximate the difference by

$$C - C_M \approx (c_g - c_x)(M - 1)q - qc_x \quad (5)$$

Therefore, there is nothing to be gained from multirate Runge-Kutta methods unless $c_g > c_x$ in which case there is a saving if

$$M > 1 + \frac{c_x}{c_g - c_x} \quad (6)$$

If c_g is only a little larger than c_x , the slight savings for large M will be lost to the undoubtedly higher overhead of a more complex program. On the other hand, for multistep methods, q is small, typically 2. Also, the number of prior points used for extrapolation is naturally the number of points used in the multistep method for two reasons. The first is that we are saving values at those points anyway. The second is that the predictor step of a multistep method is an extrapolation; we have chosen the number of values and the stepsize in the multistep method such that the error of the prediction is not too large, and this implies that the error in the extrapolation will also be appropriately small if the same number of past values are used. Consequently, the cost of a multistep integration step is only slightly higher than the cost of an extrapolation because it involves a prediction step followed by a correction. Therefore,

$$c_x < c_I$$

Using this in eq. (4), we get

$$C - C_M > q(M - 1)c_g - c_x \quad (7)$$

Therefore, we may see a saving as long as

$$M > 1 + \frac{c_x}{qc_g} \quad (8)$$

Eqs. (6) and (8) indicate that multistep methods are more likely to yield a savings in multirate schemes than Runge-Kutta methods, but that unless c_g is large compared to c_x , a large M will be needed, which means that there must be a very large disparity in behavior of the two components.

Similar comments apply to larger systems, but sparsity can have a positive effect. Suppose that y and z above each represent systems of r equations, but that the evaluation of f requires the values of only s of the z variables. Then the extrapolation cost is sc_x whereas the cost of evaluation of all of the components of g is rc_g if c_g is the cost for each

component. Thus the sparsity ratio s/r multiplies c_x in eqs. (6) and (8), reducing the value M for which savings can be achieved. Clearly this analysis can be extended to cases in which more than two different stepsizes are employed and the same general result will be obtained, namely that the ratio of the cost of extrapolation to the cost of function evaluation is critical. Also note that it is very important where possible different components are evaluated at the same point. If, for example, the two components in eq. (1) were integrated with the same stepsize on different meshes, say $t_0 + nh$ and $t_0 + (n+1/2)h$, two unnecessary extrapolations would be done in each step, and the method would take longer than if the equations were "kept in step."

4. FIXED-STEP CONVERGENCE THEORY

It is natural to wonder if the use of multirate techniques can cause problems with stability and convergence. This section will show that they do not. Indeed, the expected happens: if the methods used for each equation are individually stable and p -th order accurate, if the functions-- f and g in eq (1)--are Lipschitz continuous in all variables, and if the extrapolation scheme is $(p-1)$ -th order accurate--in the sense that the error in extrapolation is $O(h^p)$, then the solution is p -th order convergent. This result is true for any number of equations, each with a separate integrator using a separate stepsize. We will sketch the proof for two equations. It is a straightforward extension of standard proofs, and this sketch will make it clear how the result can be obtained in any particular application without obscuring the principles with generality.

Consider the numerical integration of a system of equations $y' = f(y,t)$ all using same stepsize h_n for the n -th step. It has been shown in (4) that the global error e_n^y at the n -th step of a method satisfies the equation

$$e_{n+1}^y = (S_n^y + h_n S_n^{yy}) e_n^y + d_n \quad (9)$$

where S_n^y is the stability matrix for one step of the method with stepsize h_n^y applied to the differential equation $y' = 0$, S_n^{yy} is the contribution of terms dependent on the differential equation (and depends on the partials

f_y), and d_n is the "local truncation error." In this case the local truncation error is defined quite simply; it is the amount by which the true solution fails to satisfy the method. In the case of a multirate method, eq. (9) is applicable to each set of differential equations being integrated with the same stepsize, but d_n^y is composed of three components: the truncation error due to the method applied to this set of differential equations, the interpolation/extrapolation errors when other components are approximated for the evaluation of f , and the global errors in the other components used to evaluate f . For the case of the two equations (1), we will name these components d_n^y , i_n^z , and e_n^z , respectively. The latter two terms enter into eq. (9) through the function $f(y,z,t)$, and are always multiplied by $h_n f_z$, hence the revised form of eq. (9) is

$$e_{n+1}^y = (S_n^y + h_n S_n^{yy})e_n^y + h_n f_z e_n^z + h_n f_z i_n^z + d_n^y \quad (10)$$

A similar type of equation will apply to the global errors in z , although it will apply to a different set of mesh points. However, we can define values of the e^y and e^z on the union of all mesh points used in any equation to be the values of the global errors at the last point to which the component was actually integrated. Then a form of eq. (10) applies to all components at all mesh points. For components not evaluated at that mesh point, everything on the right-hand side of eq. (10) except for $S_n^y e_n^y$ is zero and in that term S_n^y is the identity. If we denote the vector $[e_n^y, e_n^z]^T$ by e_n , we get a difference equation for e_n of the form

$$e_{n+1} = (S_n + h_n S_n^W) e_n + d_n^W \quad (11)$$

where

$$S_n = \begin{bmatrix} S_n^y & 0 \\ 0 & S_n^z \end{bmatrix},$$

$$S_n^W = \begin{bmatrix} S_n^{yy} & f_z \\ f_y & S_n^{zz} \end{bmatrix}.$$

and d_n^w is a sum of truncation and interpolation errors. The latter are bounded by $O(h_n^{p+1})$ when p -th order integration and $(p-1)$ -st order extrapolation methods are used. All that is needed is to prove the stability of the matrices appearing in eq. (11), and this follows immediately from an application of the theory in (2) since the matrices S_n^y and S_n^z , which are either the matrices of an integration method or identity matrices, satisfy the stability property individually and the matrix S_n^w is bounded because its four components are bounded. Stability and the fact that d_n^w is $O(h_n^{p+1})$ is sufficient to prove that e_n is $O(h^p)$ where h is the maximum stepsize.

5. VARIABLE-STEP METHODS

Very few problems are such that the user can predetermine the stepsize to be used throughout an integration, so most modern codes vary the order and stepsize during the integration based on local error estimates. In this section we look at the problem of varying the stepsize in a multirate method.

There are problems for which it is known that some components are always faster than others, and for these problems it may be possible to permanently set the ratio of the stepsizes. Then it is a question of choosing one step size and letting the other adjust correspondingly. A typical automatic code performs a single step integration, estimates the error, and then decides whether to accept the step because the estimated error is within tolerance, or to reject it and repeat it with a smaller stepsize. A multirate method with a fixed ratio between stepsizes in different components could be viewed as a type of cyclic method with a cycle equal to the largest stepsize used in any component. Error estimations could be made over that cycle and either the cycle accepted or rejected and repeated. Unfortunately this introduces two inefficiencies: the loss of a relatively large amount of work when a cycle is rejected, and the need to back-up a number of steps in some components. This means that a considerable number of additional past values must be saved.

If large-scale back-up and loss of work is to be avoided, errors must be estimated as each step is made. In that case, there does not seem to be

any great reason for fixing the stepsize ratio; indeed in general we cannot decide a priori on a ratio of stepsizes, some components may be the fastest at some times and the slowest at others. Three approaches to the organization of automatic multirate methods have been tried. In all approaches the local error is estimated for each component separately, and then the step in that component is either accepted or rejected. If it is rejected, it will be tried with a smaller stepsize later. The approaches are:

- An "event-driven" model in which each mesh point is viewed as an event, and the next event in time is chosen for execution.
- A "recursive" view in which all equations are attempted with the largest possible stepsize, and those that are rejected are integrated with a pair of recursive calls at half the stepsize.
- A "largest-step-first" approach which retains some of the advantages of the recursive approach but reduces function evaluations.

The first approach was tried in an experimental code described in (5). Each component of the system was integrated with a possibly different stepsize and order. They were selected by the integrator for each component independently of the behavior of the other components. Consequently, at any given time each component i had been integrated to a different t value t_i , and the integrator had suggested a different stepsize h_i for the next step for each. The basic strategy was to select the component with the smallest value of $t_i + h_i$ as the component to be integrated next. It was integrated one step, its error estimated, and a value for its next stepsize recommended. The idea behind this is that the extrapolation performed in the other components is to points within the range of their next recommended steps and within this range the extrapolation should be reasonably accurate. The snag to this arrangement is that the recommended step may be too large. If the recommended step is far too large, the extrapolation can be badly in error causing large errors to be propagated to other components. The difficulties do not stop there. If a recommended step turns out to be too large and must be greatly reduced, we will find that we need to approximate a fast component at a t value that is too far back to safely extrapolate backwards. Consequently,

an arbitrary amount of back-up may be necessary. Clearly, it is not feasible to save information for backing up to arbitrary earlier values, so we tried various techniques to prevent step failures. Consider the integration of z for a moment. The recommended step may be too large because of a change of behavior in g that is not predicted by earlier values of z , or by a change in the behavior of y that couples into z through g . The first can be predicted by doing a trial integration of z with a simple extrapolation for y . We used this to see if the recommended step seemed reasonable. It is clearly expensive because it almost doubles the number of integration steps. The second source of problems was examined by trying to estimate the effect of changes in the behavior of y on z . We assumed that the stepsize chosen for z had been based on the current knowledge of y . This is equivalent to saying that it is based on the extrapolated value of y . When y was integrated, the predictor-corrector difference is the difference between the extrapolated value of y and the new value. The effect of this difference on the z integration was estimated. This required knowledge of the Jacobian g_y . The method consisted of integrating one component one step using the selection algorithm given above, and then determining if the recommended steps of any other components had to be reduced because of the error in the step just performed. The amount of work was made manageable for a system of several equations by keeping the sparsity structure of the Jacobian. Only components which had nonzero entries in the row of the Jacobian corresponding to the equation just integrated had to be considered. The results of numerical tests of this technique indicated that it is feasible if the evaluation costs of g are sufficiently high and the ratio of stepsizes that could be used in different components was high enough. For a number of test problems of 6 to 20 equations, the number of individual evaluations of derivatives for the multirate method was about half that of the standard method. However, the overhead was very much higher.

The principal difficulty in the event-driven method is due to the integration of fast components before one is certain that the selected stepsize for the slow components is not too large. Recall from Section 2 that we said the coupling between fast and slow components is necessarily small and it might make sense to integrate the slower component first. In

the recursive method we do just that. All components are integrated using the largest recommended stepsize. Then the error in each is estimated and the results for those with small errors are accepted. The remainder are reintegrated using two steps of the half size by the same technique. The advantages of the recursive approach are that it is conceptually simple and not too difficult to program even if recursion is not provided in the language. Also, more accurate interpolation rather than extrapolation is used to get the intermediate values of the slow variables when integrating the faster ones. Its disadvantage is that there are many unnecessary function evaluations, integrations, and interpolations for those components whose stepsizes have to be reduced several times. In fact, there are almost twice as many evaluations, integrations, and interpolations because if a step of $H/2^N$ is used when the largest step is H , the number of unsuccessful steps is $2^N - 1$ to the number of successful ones 2^N .

The third technique uses the same principle as the recursive technique, that is, it integrates the slowest components with the largest stepsizes first. A maximum stepsize H is chosen, and all stepsizes are $H/2^N$ for some N . Initially, all components are known at a given time value t_0 . The components whose stepsizes are H are integrated first. A step is halved if its estimated error is too large. Next the steps of size $H/2$ are integrated, and so on. Finally, the step with the smallest size, $H/2^N$, are integrated. This process is repeated for all components whose current time value is least. The immediate effect of this is to integrate the components with the smallest step to $t_0 + H/2^{(N-1)}$. Repeating the process again causes the components with stepsizes $H/2^{(N-1)}$ and $H/2^N$ to be integrated, in that order. If the local error estimate in a component is very small, the stepsize can be doubled. However, a step of size $H/2^M$ for $M > 0$ can only be doubled when the time status of its differential equation is on a mesh point corresponding to steps of size $H/2^{(M-1)}$. The effect of this is that components using stepsize $H/2^M$ are evaluated only on the mesh points $T_0 + mH/2^M$ for $m = 0, 1, \dots, 2^M$. When time $T_0 + H$ is reached, a new maximum stepsize H is chosen on the basis of the error estimates for the slowest components.

The last method appears to be the most efficient on the basis of

preliminary tests. Note that it has one hidden additional cost. It is necessary to extrapolate the values of the fast components in order to evaluate the derivatives of the slower components which are integrated first. Because of the assumed small coupling between the fast and slow components, we have been using very low order extrapolation such as linear methods, even when higher-order integration schemes are used.

6. STABILITY, STIFFNESS, AND IMPLICIT METHODS

It must be stressed that the existence of slow and fast components has nothing to do with stiffness per se. A system may be neutrally stable and still profit from multirate methods, but the existence of stiffness can be a complicating factor because of the need to use implicit integration methods and to solve nonlinear equations at each step. When a stiff problem is to be solved, the Jacobian of all equations being integrated to the same mesh point must be formed, and this Jacobian must be used in a quasi-Newton iteration for the solution of the implicit integration formulas. In the event approach above, the values of the other variables will be extrapolated and fixed. In the recursive method, all components with larger stepsizes can be interpolated, and the system is solved for all components with the current and smaller stepsizes. In the largest-step-first method, values of variables with smaller stepsizes are not yet known. Consequently they must either be extrapolated--a risky business in stiff equations--or can be held constant. This corresponds to a zero-th order extrapolation and seems to be the best choice. In systems that arise from PDEs, there is also the possibility that the variable with the smallest time step can be ignored by reverting to a spatial discretization based on a coarser mesh or finite element discretization. This is related to the multigrid techniques discussed in (6).

7. CONCLUSION

The use of multirate methods is attractive when function evaluation costs are very high or if there is a high degree of sparsity. However, the organization of an automatic code is not simple, and it is very easy to allow the overhead to become a major part of the cost. A particular

problem is that of avoiding back-up over several steps at a step failure, and the counter-intuitive approach of integrating the slowest components first seems to be most effective in this.

REFERENCES

- {1} O. A. Paluszinski, Simulation of dynamic systems using multirate techniques, CSRL Memo 333, Engineering Experiment Station, University of Arizona, Tucson, November 1979.
- {2} J. F. Andrus, Numerical solution of systems of ordinary differential equations separated into subsystems, SIAM J. Numerical Analysis, vol. 16, no. 4, August 1979, 605-611.
- {3} O. A. Paluszinski and J. V. Wait, Simulation methods for combined linear and nonlinear systems, Simulation, vol. 30, no. 3, March 1978, 85-94.
- {4} C. W. Gear and K.-W. Tu, The effect of variable mesh size on the stability of multistep methods, SIAM J. Numerical Analysis, vol. 11, no. 4, October, 1974, 1025-1043.
- {5} A. Orailoglu, A multirate ordinary differential equation integrator, Report 959, Dept. of Computer Science, University of Illinois at Urbana-Champaign, March 1979.
- {6} A. Brandt, Multi-level adaptive solutions to boundary-value problems, Mathematics of Computation, vol. 31, no. 138, April 1977, 333-390.

BIBLIOGRAPHIC DATA SHEET		1. Report No. UIUCDCS-R-80-1000	2.	3. Recipient's Accession No.
4. Title and Subtitle AUTOMATIC MULTIRATE METHODS FOR ORDINARY DIFFERENTIAL EQUATIONS			5. Report Date January 1980	
			6.	
7. Author(s) C. W. Gear			8. Performing Organization Rept. No. R-80-1000	
9. Performing Organization Name and Address Department of Computer Science University of Illinois U-C Urbana, IL 61801			10. Project/Task/Work Unit No.	
			11. Contract/Grant No. US ENERGY/ EY-76-S-02-2383	
12. Sponsoring Organization Name and Address US Department of Energy Washington, DC			13. Type of Report & Period Covered technical	
			14.	
15. Supplementary Notes				
16. Abstracts This paper studies the application of integration methods in which different step sizes are used for different members of a system of equations. Such methods can result in savings if the cost of derivative evaluation is high or if a system is sparse. However, the estimation and control of errors is very difficult and can lead to high overheads. Three approaches are discussed, and it is shown that the least intuitive is the most promising.				
17. Key Words and Document Analysis. 17a. Descriptors numerical integration error control adaptive step selection				
17b. Identifiers/Open-Ended Terms				
17c. COSATI Field/Group				
18. Availability Statement unlimited		19. Security Class (This Report) UNCLASSIFIED		21. No. of Pages 18
		20. Security Class (This Page) UNCLASSIFIED		22. Price