Journal of Computational and Applied Mathematics 18 (1987) 3–16
North-Holland

Control of step size and order in extrapolation codes *

L.F. SHAMPINE
Numerical Mathematics Division, Sandia National Laboratories, Albuquerque, NM 87185, U.S.A.

Received 25 June 1985
Revised 1 March 1986

Abstract: Extrapolation of the semi-implicit midpoint rule is an effective way to solve stiff initial value problems for a system of ordinary differential equations. The theory of the control of step size and order is advanced by investigating questions not taken up before, providing additional justification for some algorithms, and proposing an alternative to the information theory approach of Deuflhard. An experimental code SIMP implementing the algorithms proposed is shown to be as good as, and in some respects better than, the research code METAN1 of Bader and Deuflhard.

Keywords: Stiffness, ODE codes, extrapolation.

1. Introduction

Extrapolation of the semi-implicit midpoint rule is an effective way to solve stiff initial value problems for a system of ordinary differential equations. Bader and Deuflhard [1,2] provide the basic theoretical results. In [5,6] Deuflhard builds upon previous work to develop algorithms for the control of step size and order in extrapolation codes. Based on this work, Bader and Deuflhard developed a research code METAN1 which is listed in [7].

In this paper we continue the study of the control of step size and order in extrapolation codes with particular attention given to software issues. A few basic results are first stated in Section 2. It turns out that in practice the order of a formula is somewhat ambiguous. Because we do not make the same assumption that Bader and Deuflhard do, this matter calls for a little discussion. Extrapolation generates a great many formulas, so theoretical results suggesting that only a few need be considered are of great value. The standard argument for considering only 'subdiagonal' formulas does not settle the matter in this context. An extremely simple and general argument is provided. We also provide justification for a further restriction to the formulas of Deuflhard's 'order window'.

In Section 3 we explain why we do not want to base order and step size selection on information theory, and in later sections we develop a more traditional approach as an alternative. Current extrapolation codes respond in a surprising way to a step failure. In Section

* This work performed at Sandia National Laboratories supported by the U.S. Dept. of Energy under contract no. DE-AC04-76DP0078.

4 we explain why this is appropriate. Extrapolation codes resort to much higher orders than their competitors. In Section 5 we provide some insight as to why this is so and use it to justify portions of our order selection algorithm. Sections 6 and 7 develop some software devices intended to improve the performance of extrapolation codes on difficult problems. The initial step size is a troublesome one with any integrator, and this is especially true for extrapolation. Section 8 presents one way to handle it.

Our algorithms have been implemented in an experimental code SIMP. It is compared to METAN1 in Section 9. The results show that our simpler, rather traditional approach to step size and order control works at least as well as the one based on information theory realized in METAN1. For easy problems the codes behave much the same. The devices we propose for coping with a number of software issues give SIMP an advantage for difficult problems.

2. Extrapolation

Extrapolation of the semi-implicit midpoint rule is way to solve a stiff initial value problem for a system of ordinary differential equations

$$y' = f(y). \tag{2.1}$$

A step of length H from $y_0 \doteq y(x_0)$ is carried out via a sequence of subintegrations with the semi-implicit midpoint rule. Details can be found in [1,2]. An approximate Jacobian, $J \doteq f_y(y_0)$, is formed at each step. For a sequence of integers $\{n_i\}$, the step sizes $h_i = H/n_i$ are defined, and quantities $T_{i,1}$ are constructed successively by a subintegration from x_0 to $x_0 + H$ using a semi-implicit midpoint rule and step size h_i. Cost is measured in units of the cost of evaluating the f of (2.1). If c_J is the cost of forming J, then the cost of carrying out the subintegrations with $n_1, n_2, \ldots, n_j$ is A_j where

$$A_1 = c_J + n_1 + 1, \qquad A_{k+1} = A_k + n_{k+1}, \quad k = 1, \ldots .$$

The sequence studied in this paper is $\{n_i\} = \{2, 6, 10, 14, 22, 34, 50\}$.

With suitable assumptions the error $T_{i,1} - y(x_0 + H)$ has an asymptotic expansion in powers of H^2. High order approximations $T_{i,k}$ to $y(x_0 + H)$ are constructed from the $T_{i,1}$ by polynomial extrapolation. The local error $e_{i,k} = T_{i,k} - y(x_0 + H)$ of each element $T_{i,k}$ of the extrapolation tableau is asymptotically

$$e_{i,k} \sim E_k(H) H^{2k} / (n_{i-k+1} \cdots n_i)^2 \tag{2.2}$$

where $E_k(H)$ is O(1). Thus elements in column k of the tableau all represent formulas of order $2k - 1$ in the step size H. The relation (2.2) implies that as $H \to 0$.

$$e_{i+1,k} \sim (n_{i-k+1}/n_{i+1})^2 e_{i,k}, \tag{2.3}$$

which we require later. For $k \leqslant i - 1$ the local error can be estimated by

$$e_{i,k} \doteq T_{i,k} - T_{i,k+1} = e_{i,k} - e_{i,k+1}. \tag{2.4}$$

This estimate is justified by the fact that $T_{i,k+1}$ is of higher order than $T_{i,k}$.

Odd though it sounds, there is a question as to the order of the formulas used. Bader and Deuflhard [2, p. 390 ff.] show that the term $E_k(H)$ in (2.2) can be split into two parts, one of which is O(H) and the other, O(1). The latter part is always present, and for a problem of the

form $y' = Jy$, is the only part present. Thus as $H \to 0$, it is the case that $E_k(H)$ is O(1). Nonetheless, Bader and Deuflhard make the point that for practical step sizes H, the O(H) term may dominate the O(1) term in $E_k(H)$, leading to the higher order behavior $e_{i,k} = O(H^{2k+1})$. They argue that this is likely to happen when the Jacobian f_y changes rapidly. It is suggested that the higher order assumption might "... be significantly preferable in the more difficult non-linear problems". Because their numerical experience was in agreement with this, they implemented the higher order assumption in their code METAN1.

We have implemented the lower order assumption in SIMP. It correctly describes the behavior for problems of the form $y' = Jy$ and for any problem when H is sufficiently small. Of course, the step size is small enough in any non-stiff portion of the integration. Besides 'normal' situations like initial transients, this includes severe difficulties such as quasi-discontinuities. There is a restriction on H in the theory of [1,2] related to how fast the Jacobian changes; even without this, it is plausible that a rapidly changing Jacobian would require a comparatively small step size. Our numerical experience has been the opposite of that reported by Bader and Deuflhard. This may well be due to other differences in our implementations, but the experiments reported in Section 9 and others we have performed show that a code based on the lower order assumption can be just as effective as METAN1 in general and in particular for difficult, nonlinear problems.

For each i, k with $k \leqslant i-1$ we can estimate the error $e_{i,k}$ of $T_{i,k}$ resulting from a step of length H. If a step is a success so that

$$\| e_{i,k} \| \leqslant \tau \tag{2.5}$$

is satisfied, the largest step size, $H_{i,k}$, which would yield an accuracy of $\tau_a < \tau$ on the next step is found in the usual way to be

$$H_{i,k} \doteq H\left(\frac{\tau_a}{\| e_{i,k} \|} \right)^{1/2k}. \tag{2.6}$$

The cost of this formula is i subintegrations, hence A_i evaluations of f. Its efficiency is the distance the integration is advanced divided by the cost (in evaluations of f) of taking the step:

$$\text{eff}(i, k) = \frac{H_{i,k}}{A_i} \doteq \frac{H}{A_i}\left(\frac{\tau_a}{\| e_{i,k} \|} \right)^{1/2k}. \tag{2.7}$$

On taking a step of length H with a total of j subintegrations, we are in a position to ascertain the most efficient formula for the next step from all the $T_{i,k}$ with $i \leqslant j$, $k \leqslant i-1$. It is of obvious value to demonstrate that we can restrict our considerations to only a few of these formulas.

A method of analysis proposed by Stoer [18] and subsequently refined and applied by Deuflhard [5] and Shampine [13] allows one to study the relative efficiency of the formulas in a column of the tableau as $H \to 0$. For standard choices of the $\{n_i\}$ and the explicit midpoint rule, it turns out that in column $j-1$, the sub-diagonal element $T_{j,j-1}$ is the most efficient formula, and this is true for each $j-1$. This behavior is not true of the semi-implicit midpoint rule with the sequence we study. (Some details are given in Section 5). There is a simpler and more general way to see that the subdiagonal elements are to be preferred as $H \to 0$: In terms of function evaluations, all elements $T_{j,k}$ in row j of the tableau cost the same. Asymptotically $T_{j,k+1}$ is more accurate than $T_{j,k}$ (by a factor of $O(H^2)$), hence is more efficient. Thus as $H \to 0$, the subdiagonal element $T_{j,j-1}$ is the most efficient formula in row j, and this is true for each j.

(This does not contradict our claim that $T_{j,j-1}$ need not be the most efficient element in column $j-1$.)

The efficiency argument suggests that we restrict our attention to the subdiagonal and diagonal elements of the tableau. In this context the stability of the formulas is crucial. Kuhlemann [10] found the linear stability of these formulas to be excellent. Hairer, Bader, and Lubich [9] confirm the good stability of part of the tableau with respect to a nonlinear stability concept.

There is another reduction possible in the number of formulas to be considered. It is one aspect of what Deuflhard calls the 'order window'. Suppose that we predict j subintegrations to be optimal for the next step. As we take the step, we actually form $T_{i,i-1}$ for $i=1,\ldots,j$. Should we test them as in (2.5)? By assumption $T_{j,j-1}$ was the most efficient element in the previous step so that

$$H_{j,j-1}/A_j > H_{i,i-1}/A_i \quad \text{for } i=1,\ldots,j-1.$$

By definition $H_{i,i-1}$ is predicted to result in an error of τ_a. We actually take the step with step size $H_{j,j-1}$, hence we predict that at i subintegrations we shall see an error of

$$\tau_a\left(H_{j,j-1}/H_{i,i-1}\right)^{2i-2} > \tau_a\left(A_j/A_i\right)^{2i-2}.$$

The cost coefficients A_k increase pretty rapidly with k. For $i<j-1$ the factor A_j/A_i here is large, and we are predicting an error rather bigger than the tolerance τ. If we observe an error less than τ, we cannot trust the result because it differs so much from what was predicted. The case $i=j-1$ is marginal because τ_a is substantially smaller than τ and A_j/A_{j-1} is not so large. It seems reasonable, then, to restrict our attention to $i \geqslant j-1$. The first step is a special case we take up in Section 8.

3. Information theory approach

The standard approach to predicting an optimal step size and order just presented works well, but only those orders actually tried in the step can be considered. Key practical questions are when to try a higher order and what step size to try then. In [6] Deuflhard attempts to answer these questions using information theory. The approach considers the behavior of an ideal integrator when applied, for a given tolerance τ, to a large ensemble of sufficiently smooth problems. Assuming that $q+1$ subintegrations is optimal, it is argued that the 'average' error $e_{i,k}$ satisfies

$$\operatorname{avg}(e_{i,k}) = \tau^{(A_i - A_{i-k}+1)/(A_{q+1}-A_1+1)}. \tag{3.1}$$

Obviously there is some difficulty applying this theory because of the assumption that the optimal number of subintegrations is known. Leaving this aside, it is still necessary to ask if extrapolation behaves enough like an ideal integrator that it can be described by (3.1). Now (3.1) implies that

$$\operatorname{avg}(e_{i+1,k}) = \tau^{(A_{i+1}-A_i+A_{i-k}-A_{i+1-k})/(A_{q+1}-A_1+1)} \operatorname{avg}(e_{i,k}).$$

On the other hand, we already know that (2.3) holds asymptotically for all smooth problems, hence for their average:

$$\text{avg}(e_{i+1,k}) \doteq (n_{i-k+1}/n_{i+1})^2 \text{ avg}(e_{i,k}).$$

In this known relation there is no dependence on the tolerance τ as there is in the case of an ideal integrator. If a wide range of τ is considered, there can be a considerable difference in behavior. Thus it is far from clear that extrapolation is close enough to an ideal integrator that the theory can provide useful predictions.

Even if the information theory model were applicable, it would not provide what is needed. In the integration we deal with a specific problem, and we must answer questions like, should we raise the order now? The answers depend on the problem at hand; obviously they cannot be provided by a theory which averages out the effect of the problem like (3.1) does. The standard prediction (2.6) is based on the observed behavior of the formula applied to the given problem. The information theory approach says that for a given tolerance τ, the efficiency increases *on the average* as the order is increased, up to a certain value. (This is the content of (4.8′) of [6], see also the discussion of p. 9 of [7].) The distinction is most clearly seen when, as often happens, the standard approach says to reduce the order *for this particular problem* when the information theory approach says that a higher order is better for a '*typical*' problem. Of course, one should give preference to the scheme based on the observed behavior in this situation, and METAN1 does.

The rules for adjustment of order presented in [6] and implemented in METAN1 are plausible enough. It is not clear that extrapolation is modelled well enough by the information theory approach to justify these rules and we have noted that the approach does not really answer the right questions. For these reasons we tried a more traditional approach to the adjustment of order. The numerical results of Section 9 show that it works at least as well as the approach based on information theory.

4. A failed step

In this section it is supposed that $T_{j,j-1}$ was predicted to have the required accuracy τ, but on trying the step, it did not. With a fixed order method, the standard procedure is to reduce the step size H and try the step again. With variable order methods like the backward differentiation formulas (BDF), the same thing is done except that a lower order might be used. It is remarkable that the extrapolation codes try a *higher* order formula, $T_{j+1,j}$. If this attempt fails, the step size is reduced and the step tried again. This remarkable action on a failed step has not received comment that the author has noticed. We shall explain in this section why we believe it to be the correct way to proceed. Also, we contribute to understanding a matter that has always puzzled the author, namely that the extrapolation codes resort to much higher orders than competitors like the BDF codes.

Recall that $T_{j,j-1}$ results from subintegrations with step sizes $h_2, \ldots, h_j$. The formula $T_{j+1,j-1}$ of the same order results from subintegrations with the smaller step sizes $h_3, \ldots, h_{j+1}$. The errors of these two formulas are related asymptotically according to (2.3) as

$$e_{j+1,j-1} \doteq (n_2/n_{j+1})^2 e_{j,j-1}.$$

One possibility for dealing with an error $e_{j,j-1}$ too large is to repeat the step with the formula $T_{j,j-1}$ and a smaller step size H'. Another is to form $T_{j+1,j-1}$ with the same step size H. The observations just made about the formulas and their errors show that the two possibilities are very closely related. There are two main differences. The first possibility shortens the interval to $[x_0, x_0 + H']$, and the second continues to work with $[x_0, x_0 + H]$. If the problem changes character in $(x_0 + H', x_0 + H]$, the possibilities behave quite differently. The other main difference is that with the standard procedure, one can reduce H as much as is needed, but in going to $T_{j+1,j-1}$, we have already chosen n_{j+1}, hence the asymptotic reduction of the error.

We predicted that $\| e_{j,j-1} \|$ would be roughly $\tau_a < \tau$. In view of the fact that the observed error is greater than τ, either the problem has become somewhat harder, or the assumptions justifying the prediction have broken down. If the assumptions have some validity, $T_{j+1,j-1}$ should be enough more accurate to yield convergence. Moreover, we expect the higher order result $T_{j+1,j}$ to be even more accurate, and it is available at no extra cost. Thus it is reasonable to expect that $T_{j+1,j}$ will provide a sufficiently accurate solution. In our view, the raise in order is largely formal. What is done can be regarded as the functional equivalent of trying again with a smaller step size, followed by local extrapolation.

There are a couple of other reasons for raising the order. As Stoer [18] observed, if we reject the step, we must go to all the work of building up to $T_{j,j-1}$ again. It may well be cheaper to form $T_{j+1,j}$. This matter can be quantified, but we do not do so because it is useful to raise the order even if it is not a cheaper way to get a successful step. A variable order code must explore the possibility that higher orders are more efficient. In METAN1 and SIMP raising the order in this situation is an important way to find the most efficient order.

If $T_{j+1,j}$ does not result in a successful step, it is clear that the assumptions justifying the prediction are not valid. Because we do not then trust the estimates of the local errors, we halve the step size in SIMP. In METAN1 an 'optimal' reduction is made, but the step is at least halved. Usually the codes behave the same, but it is easy to understand why an 'optimal' reduction might be very much too small. If the problem changes character in $[x_0, x_0 + H]$, the code might think a very large reduction necessary when all that is needed is a step size H' small enough that the problem is well behaved on $[x_0, x_0 + H']$. To cope with this kind of difficulty efficiently, we must do more than just halve the step size, we must limit subsequent step size increases. The matter is discussed in Section 7. This is one situation handled much more efficiently by SIMP than by METAN1.

5. Raising the order

After a step is taken successfully, we have the information available to select the most efficient among orders up to the highest tried. Crucial questions are when to try a still higher order and what step size to try then. We have already argued that if we are not successful at the order expected, we should try the next higher order. This is an important way to explore higher orders, but it does not suffice. Unfortunately there is a way by which we actually lose information. In connection with our discussion of the 'order window' in Section 2, we observed that we might obtain a sufficiently accurate approximation with one fewer subintegrations than predicted. Accepting this result reduces the orders we can consider for the next step. We have experimented with a version of SIMP which does not allow this loss of information, but using a cheaper, lower order when possible helps the overall efficiency too much to exclude accepting early convergence.

We must be careful about our assumptions when discussing the most efficient order in a variable order code. This is because as $H \to 0$, a higher order formula eventually is more efficient regardless of how much more it costs. Experience with extrapolation codes suggests that they must resort to rather high orders to be competitive. This is not just a consequence of the information theory approach for not all codes use it, and experiments with many different algorithms in the investigation reported here support the belief. Exercising due care about our assumptions, we use asymptotic arguments in this section to conclude that high orders are often efficient when extrapolating the semi-implicit midpoint rule. This is the basis of the algorithm we then detail.

Following the method of [13] we compare the efficiency of $T_{j+1,j-1}$ to $T_{j,j-1}$. We suppose that we have taken a step of length H with $T_{j,j-1}$. The expression (2.6) with $(i, k) = (j, j-1)$ provides an estimate of the largest step size, $H_{j,j-1}$, which will achieve an accuracy of τ_a. The same argument with the formula $T_{j+1,j-1}$, the definition of efficiency, and the asymptotic relation (2.3) lead to

$$\frac{\text{eff}(j+1,\ j-1)}{\text{eff}(j,\ j-1)} \doteq \frac{A_j}{A_{j+1}} \left(\frac{n_{j+1}}{n_2} \right)^{1/(j-1)} . \tag{5.1}$$

Here we assume only that H is small enough that the leading terms dominate in the asymptotic expansions. This is a fundamental practical assumption for the adjustment of the step size in the code.

The costs A_j depend on the cost c_J of forming a Jacobian. For the procedure studied in this paper, it is now easy to evaluate (5.1) numerically to learn that $T_{j+1,j-1}$ is more efficient than $T_{j,j-1}$ for $j = 2, \ldots, 6$ for all $c_J \geq 8$.

The error in $T_{j,j-1}$ is estimated by comparison to $T_{j,j}$, so another fundamental practical assumption is that $\| e_{j,j} \|$ is less than $\| e_{j,j-1} \|$, say,

$$\| e_{j,j} \| = \rho \| e_{j,j-1} \|, \quad \rho < 1.$$

Asymptotically ρ is $O(H^2)$, but with practical step sizes H it need not be particularly small. The asymptotic relation (2.3) implies that approximately

$$\| e_{j+1,j} \| \doteq \rho (n_1/n_2)^2 \| e_{j+1,j-1} \| = \tfrac{1}{9} \rho \| e_{j+1,j-1} \|.$$

This argument supports our claim that $T_{j+1,j}$ is normally rather more accurate than $T_{j+1,j-1}$. We conclude that if it is more accurate at all, then for all $c_J \geq 8$, $T_{j+1,j}$ is more efficient than $T_{j,j-1}$. If it were just 10% more efficient, it would be better to raise the order for all $c_J \geq 3$.

The asymptotic analysis presented is based on assumptions reasonable for describing computing practice. We shall not push it far, merely use it to justify an aggressive approach to trying higher orders. Now we shall describe and explain this approach as implemented in SIMP, except in the case of the first step which is taken up in Section 8.

In some circumstances we cannot, or should not, consider a raise of order. If we are already doing the maximum number of subintegrations allowed (7), we obviously cannot raise the order. In Section 7 several reasons for restricting the step size are given. If they cause the step size to be limited, raising the order is not efficient because a higher order will be a more expensive way to take the same step. On taking a step, we can consider a number of orders for their efficiency. If the highest order tried is not the most efficient, we would normally lower the order for the next step. It hardly seems reasonable to consider an increase of order then, and we do not.

We have argued that if certain assumptions hold, it is likely to be efficient to raise the order. There is, of course, the possibility that the assumptions do not hold. Part of the algorithm in METAN1 can be thought of as a way of recognizing when the hypotheses are not valid. We do something similar, though much simpler. The idea is that unexpected behavior indicates a possible breakdown in the assumptions so that one should be cautious about raising the order. In SIMP a step failure for any reason prevents a raise in order for two successful steps. (Recall that the order might be raised anyway in an attempt to get a successful step.) If there is a loss of order because of convergence one subintegration earlier than predicted, a raise in order is considered immediately (subject to the basic qualifications about the last step and the step size restrictions). The much more elaborate algorithm in METAN1 keeps track of the order at which failures occur and how often. Though plausible, the complication did not seem justified in our experiments.

Having decided to try a higher order, we must select a corresponding step size. As an alternative to the information theory model, we take the simple approach of hypothesizing that $T_{j+1,j}$ will be a little more efficient than $T_{j,j-1}$. Experiments showed that 10% is a suitable value for SIMP. Thus we use

$$H = 1.1(A_{j+1}/A_j)H_{j,j-1},$$

reduced if necessary to satisfy the various restrictions imposed on the step size in general as described in Section 7. In the code we do not reset the variable which contains the predicted 'optimal' order. In effect we say that j is the best number of subintegrations, but that we are very aggressive in the selection of H, hence expect to have to resort to $j+1$ subintegrations to get convergence. Indeed, in this situation only, we allow $j+2$ subintegrations before declaring a step unsuccessful.

6. Consistency checks

For reasons of efficiency an extrapolation code must allow large increases in the step size. Unfortunately this may result in a step size so out of scale that basic assumptions fail. The situation *may* be recognized by the local error estimates and corrected by a reduced step size. Here is a situation in which an 'optimal' step size reduction deduced from the asymptotic behavior of the error estimates is certainly not justified and may be far from a suitable value. Besides the difficulty of the local error estimates being of, at best, marginal value, they are formed only after quite a bit of computation. In [15] a consistency check is proposed which can be performed early in the computation. It is based on the fact that two results of the same order are formed at the end of each subintegration — the basic result of the semi-implicit midpoint rule, η_n, and the smoothed result, S_n. It is reasonable to insist that these results be consistent. In SIMP we require that

$$\| \eta_n - S_n \| \leq 0.75$$

where the 0.75 is an experimentally determined quantity. This relies on the fact that the error control in SIMP is a relative one which we describe fully below. If the results are not consistent, the step is rejected and tried again with half the step size. This check is performed at the end of the first subintegration, hence early in the computation. Experiments showed that it was worth the trouble to apply the same consistency check to the second subintegration, too.

One of the examples we take up in Section 9 is a relaxation oscillation described by Van der Pol's equation. The oscillation has long stretches of slow change followed by very large changes taking place in very short times. The role of the consistency checks is easy to understand here. While the solution is changing slowly the code will take large steps and possibly resort to high orders. Eventually the code steps past the sharp change. The step is then completely out of scale, which can be recognized cheaply by the consistency checks. Attempting to use an error estimate to reduce the step size 'optimally' is likely to result in a tiny step size which falls far short of the trouble spot and so is equally inappropriate. Very sharp changes in the solution is one situation leading to a failure of consistency. A less dramatic, but much more common, situation is the solution of problems to nominal accuracy. In this accuracy range it is always difficult to obtain reliable results. The consistency checks are a big help then.

In an earlier version [1] of METAN1, Bader and Deuflhard monitored the growth of the approximate solutions in the subintegrations. This played the role of the consistency checks we have described. Later they abandoned this device in favor of the same kind of consistency checks we propose, but executed in a much more elaborate manner. The scheme we propose seems to work at least as well as previous attempts to cope with step out of scale, and has the virtue of simplicity.

7. Limits on the step size

There are a number of reasons for altering an 'optimal' step size selected by the code. One is to accommodate output. It is not our purpose to compare extrapolation codes to a competitor like the backward differentiation formula (BDF) codes, but they differ so dramatically in this matter that some comment is necessary. A step with an extrapolation code is very expensive compared to a step with a BDF code, so to be competitive, an extrapolation code must take much longer steps. This means that fewer answers are produced by an extrapolation code. By limiting the step size to a maximum value specified by the user, more answers can be obtained. Although this increases the cost, variation of order ameliorates the effect. A BDF code not only produces many more answers directly, it can even interpolate between steps. A carefully written code is, then, weakly affected by the number and placement of points where an answer is desired. Some work [16] has been done on interpolation for extrapolation codes, but there is still much to be done. Like METAN1, SIMP integrates to a specified output point with the option of returning results at each step. A maximum step size provides a way to get more frequent output. Extrapolation codes in general, and METAN1 and SIMP in particular, are very inefficient if a lot of output is requested at specific points. To reduce this inefficiency, the technique of 'stretching' [17] is used in both codes. Specifically, in SIMP a step size can be increased by up to 10% so as to reach an output point.

The fact that an extrapolation code takes much longer steps than a BDF code does not matter as long as appropriate step sizes vary slowly. Extrapolation is at a disadvantage when a large increase of step size is needed. A BDF code can accumulate such a large change over a number of steps. To be competitive, an extrapolation code must accomplish the increase in a very few steps. METAN1 allows an increase by a factor of 100 in a single step. BDF codes do not permit such a large increase because it cannot be justified by asymptotic arguments. Nevertheless, we are convinced that this must be done for reasons of efficiency in an extrapolation code and that

it is possible. The distinction between the methods is that computation of a step by extrapolation is independent of the results of earlier steps. In taking a step, increasingly high orders are tried, and each order is very much more accurate than the preceding one. For these reasons even extremely poor step size predictions can be handled. Experiments we have made confirm that permitting large increases of step size are necessary to an efficient extrapolation code. On the other hand, this can also cause trouble in a way that we now take up.

In Section 4 we emphasized that proceeding as we do, a failed step is likely to be due to a breakdown of basic assumptions. Suppose the problem abruptly changes character somewhere in $[x_0, x_0 + H]$. In the absence of other information it is appropriate to halve H when the step fails. After possibly repeated halving, we are able to take a successful step from x_0 to $x_0 + H'$. It may well be that the problem is perfectly regular on $[x_0, x_0 + H']$ with the consequence that a very large increase of step size is indicated. Allowing this increase is likely to get us back into trouble immediately. Simply preventing an increase on the first step after a failure helps the efficiency a great deal. In SIMP we even restrict the rate of increase for several steps. Specifically, we limit the increase at each step to a factor of SAFE. The factor is initialized to 100. On a step failure it is set to 1. On subsequent steps SAFE is multiplied by 3, but its value is limited to 100. METAN1 permits an increase by a factor of 100 after any step. This small algorithmic difference in SIMP makes it more efficient when dealing with difficult problems.

8. Initial step size

The first step is special because we have no assurance that the step size provided is at all appropriate. In variable order codes based on the backward differentiation formulas, the first step is taken at the lowest order, and the step size and order are built up gradually. We have tried this plan with extrapolation, and found it to be ineffective because very few steps are taken in the whole integration. Although the procedure we present here for handling the first step performs adequately, we think this matter needs more investigation.

The consistency checks described in Section 6 are important to recognizing and coping efficiently with an initial step size much too large. Let us suppose in the rest of this section that the step size H being attempted passes these checks.

We need to explore the higher orders, but we do not want to waste a lot of effort by going to a high order only to reject the step. On the first step it is natural to test convergence at each order, and this is done in both METAN1 and SIMP. The information theory model is employed in METAN1 by assuming that the optimal number of subintegrations is the maximum (for the tolerance specified) and raising the order until convergence is obtained, or the maximum order is reached, or the error observed is not consistent with the prediction of the model. The model is certainly not justified in these circumstances because we do not know the optimal order. The scheme is properly viewed as a heuristic one that expects a certain accuracy at each order and if it is not found, the step is rejected. We do this too, but in a simpler way. After j subintegrations, we have an estimate of $\| e_{j,j-1} \|$. If we do not have convergence, we ask if an order of magnitude improvement at each subintegration would yield convergence by the time we reach the maximum number of subintegrations; specifically, is

$$10^{-(7-j)} \| e_{j,j-1} \| \leqslant \tau ?$$

If so, we try another subintegration. Otherwise we reject the step.

When the step is rejected, the optimal order (and step size) is determined. In SIMP we at least halve the step size, but do not allow a reduction of more than two orders of magnitude. METAN1 proceeds in a similar way. Normally we do not trust the error estimates on a failed step, so we do not make an 'optimal' reduction. We do not trust them on the first step either; this is just a reasonable way to reduce the step size faster than halving in circumstances when this might be desirable.

9. Experimental results

In this section we present some numerical results comparing METAN1 to SIMP. The rather traditional approach to step size and order control realized in SIMP performs at least as well as the information theory approach realized in METAN1. For easy problems the codes behave much the same. In this paper we have raised a number of software issues and proposed ways of coping with them which give SIMP an advantage for difficult problems.

It is relatively easy to compare METAN1 [7] and SIMP because they are so closely related. The error control implemented in METAN1, like all controls for stiff problems, presents difficulties for certain problems. This is the only thing we altered in METAN1, except for those changes necessary to run in single precision on CDC and Cray computers with a unit roundoff u of about 7.1×10^{-15}. Roughly speaking, METAN1 controls the error in a component relative to the largest value of the solution seen so far in the integration. This may not be sufficiently stringent when a component has a peak, a difficulty that occurs in one of the examples. An RMS norm is used which this author feels is not sufficiently stringent. We accepted these difficulties, but we could not accept the initialization of the error control when a solution component has initial value zero. Thus we have altered METAN1 so that the error in a component is taken relative to the larger of $10u$ and the largest magnitude of the component seen so far in the integration, including the tentative result of the step. This is a minor modification which leads to a *much* more stringent control on components with initial value zero, which is common with the test problems.

For a test set we have basically chosen to work with the ten problems Chm 1,..., Chm 10 of Enright and Hull [8]. Like Curtis [4] we describe this as a set of small but realistic problems. Also like Curtis [4, p.79], we corrected an error in the sign of a coefficient of Chm 4. We changed Chm 6 back to its original form [11, p.197] posed on the interval [0, 1000] with initial step size 7.55×10^{-18}. We added the kidney model problem with parameter 0.99 considered by Bader and Deuflhard [2, p.394], which has interesting features they discuss. Van der Pol's equation in relaxation oscillation as stated in [12, p.411] with initial step size 3.33×10^{-4} is an additional, demanding problem.

One of the difficult aspects of comparing codes is to treat fairly the accuracy they achieve. SIMP and METAN1 usually get about the same accuracy when given the same tolerance, so for the limited objective of this section, we ignore this point and just compare the cost of solution. Conventional tolerances with stiff problems are 10^{-2}, 10^{-4}, 10^{-6}. Some of the problems are so difficult that the results with tolerance 10^{-2} are not consistent. Indeed, in the tests of Bader and Deuflhard [2] for the kidney model problem, they report consistent results only for tolerances 10^{-6} and smaller. In the present tests the harder problems are solved with tolerances 10^{-4}, 10^{-6}, 10^{-8} and consistent results are obtained with one exception. As we alluded to earlier and Curtis

[3] discusses at some length, the error control implemented is quite inappropriate for Chm 9 with the consequence that the end point errors are not consistent. Because we believe that the codes are producing comparable accuracy, we did not give this problem special treatment.

The Van der Pol equation problem is so much harder than the rest that results for it are presented separately. Chm 6 in its original form, Chm 9, and the kidney model problem are aggregated as the 'hard' problems. The remaining problems, Chm 1–5, 7, 8, 10, are aggregated as the 'moderately hard' problems. All problems are written in autonomous form and all are nonlinear.

The test set [8] specifies an initial step size h_0, and we computed one in a similar way for the problems we added or changed. To reduce the effect of the given initial step size, we solved each problem with h_0, $10h_0$, and $10^{-1}h_0$ and averaged the results.

There are a number of the usual annoyances arising in numerical testing. One is that METAN1 would not solve Chm 6 in its original form for initial step size $10^{-1}h_0$ at all the tolerances. We simply dropped this step size in the comparison for the 'hard' problems.

Neither code can solve the Van der Pol equation problem posed in a straightforward way because of overflow (even with the very large exponent range we had available). We modified the definition of the equation so that whenever a solution component becomes larger in magnitude than 10^5, the derivative of the component is defined to be zero. The true solution of the problem posed has components rather smaller in magnitude than 10^5, so this device amounts to redefining the problem well outside a region containing the desired solution. The codes do try to leave this region at which time the lack of smoothness of the derivatives then causes the codes to realize that the step size is out of scale.

Tables 1–3 present a comparison of SIMP and METAN1 with respect to the major measures of work, viz. NJAC — the number of approximate Jacobians formed, NDEC — the number of times a matrix was decomposed into triangular factors, NFCN — the number of evaluations of the f of (2.1), NSOL — the number of times a linear system with factored matrix was solved. Both codes form approximate Jacobians by differences. Because the test set involves only small problems, we preferred to count NJAC rather than to include the cost of forming the Jacobian in NFCN. It is hoped that a better guide to costs for larger problems is thereby provided. It must be recognized that the basic algorithms take c_J into account. Here it is set equal to the number of equations.

The codes differ in certain respects that we do not think affects the results. Specifically, they do not use the same subroutines for the solutions of linear systems nor for the approximation of Jacobians.

The codes behave much the same for the eight moderately hard problems of Table 1, although SIMP does seem more efficient at the crudest tolerance. This supports our claim that a rather

Table 1
Cost of solution of 8 moderately hard, nonlinear problems. Results are averaged over 3 initial step sizes

TOL	SIMP				METAN1			
	NJAC	NDEC	NFCN	NSOL	NJAC	NDEC	NFCN	NSOL
10^{-2}	66	143	698	775	76	174	872	970
10^{-4}	87	248	1554	1715	87	263	1669	1845
10^{-6}	104	392	3192	3480	101	391	3237	3527

Table 2
Cost of solution of 3 hard, nonlinear problems. Results are averaged over 2 initial step sizes

TOL	SIMP				METAN1			
	NJAC	NDEC	NFCN	NSOL	NJAC	NDEC	NFCN	NSOL
10^{-4}	99	387	2654	2942	133	459	2778	3104
10^{-6}	108	488	4354	4734	144	591	4645	5092
10^{-8}	128	642	6862	7376	162	786	7836	8460

Table 3
Cost of solution of a relaxation oscillation described by the nonlinear Van der Pol equation. Results are averaged over 3 initial step sizes

TOL	SIMP				METAN1			
	NJAC	NDEC	NFCN	NSOL	NJAC	NDEC	NFCN	NSOL
10^{-4}	77	386	2636	2945	133	466	2743	3076
10^{-6}	87	497	4438	4848	177	731	5120	5674
10^{-8}	132	831	8833	9532	224	1000	8707	9483

traditional approach to step size and order selection can work as well as one based on information theory. The additional software devices of SIMP come into play on the three harder problems of Table 2 and especially on the difficult problem of Table 3. Considering that the underlying method is the same for both codes, the differences seen in these two tables are notable. The performance of SIMP on the problems of Table 2 supports our statement in Section 2 that a code based on the lower order assumption can perform as well as one based on the higher order assumption when solving difficult problems. We have already explained how integration of the relaxation oscillation of Table 3 benefits from software devices in SIMP. The improvement with respect to NJAC and NDEC is dramatic.

Even with the difficult problems, there is not a large number of successful steps so that the initial step size has a *much* greater effect than with, say, a BDF code. Nevertheless the relative behavior of the two codes for each initial step size is the same as the average behavior displayed.

Acknowledgement

An investigation of this kind involves a great deal of numerical experimentation. L.S. Baca provided the author with this vital support, and she provided something even more important — constructive criticism which helped the author sort out his ideas. He acknowledges with gratitude her contribution to this paper. The constructive criticism of an anonymous referee made this a better paper.

References

[1] G. Bader and P. Deuflhard, A semi-implicit mid-point rule for stiff systems of ordinary differential equations, Univ. Heidelberg, SFB 123 Rept. 114, 1981.

[2] G. Bader and P. Deuflhard, *Numer. Math.* **41** (1983) 373–398.
[3] A.R. Curtis, Solution of large, stiff initial value problems — the state of the art, in: D. Jacobs, Ed., *Numerical Software — Needs and Availability* (Academic Press, New York, 1978) 257–278.
[4] A.R. Curtis, The FACSIMILE numerical integrator for stiff initial value problems, in: I. Gladwell and D.K. Sayers, Eds., *Computational Techniques for Ordinary Differential Equations* (Academic Press, New York, 1980) 47–82.
[5] P. Deuflhard, Order and stepsize control in extrapolation methods, Univ. Heidelberg, SFB 123 Rept. 93, 1980.
[6] P. Deuflhard, *Numer. Math.* **41** (1983) 399–422.
[7] P. Deuflhard, Recent progress in extrapolation methods for ordinary differential equations, *SIAM Rev.* **27** (1985) 505–535.
[8] W.H. Enright and T.E. Hull, Comparing numerical methods for the solution of stiff systems of ODEs, in: L. Lapidus and W.E. Schiesser, Eds., *Numerical Methods for Differential Systems* (Academic Press, New York, 1976) 45–66.
[9] E. Hairer, G. Bader, and Ch. Lubich, On the stability of semi-implicit methods for ordinary differential equations, *BIT* **22** (1983) 211–232.
[10] C. Kuhlemann, Stabilitätsuntersuchungen an der semi-impliziten Mittelpunktsregel, Univ. Heidelberg, Inst. Angew. Math. Diplomarbeit, 1973.
[11] L. Lapidus, R.C. Aiken and Y.A. Liu, The occurrence and numerical solution of physical and chemical systems having widely varying time constants, in: R.A. Willoughby, Ed., *Stiff Differential Systems* (Plenum Press, New York, 1974) 187–200.
[12] L.F. Shampine, Evaluation of a test set for stiff ODE solvers, *ACM TOMS* **7** (1981) 409–420.
[13] L.F. Shampine, Efficient extrapolation methods for ODEs, *IMA J. Numer. Anal.* **3** (1983) 383–395.
[14] L.F. Shampine, Efficient extrapolation methods for ODEs II, *IMA J. Numer. Anal.* **5** (1985) 23–28.
[15] L.F. Shampine and L.S. Baca, Smoothing the extrapolated midpoint rule, *Numer. Math.* **41** (1983) 165–175.
[16] L.F. Shampine, L.S. Baca and H.-J. Bauer, Output in extrapolation codes, *Comput. Math. Appl.* **9** (1983) 245–255.
[17] L.F. Shampine and H.A. Watts, The art of writing a Runge–Kutta code II., *Appl. Math. Comp.* **5** (1979) 93–121.
[18] J. Stoer, Extrapolation methods for the solution of initial value problems and their practical realization, in: D.G. Bettis, Ed., *Proc. Conference on Numerical Solution of ODEs*, Lecture Notes in Math. **362** (Springer, New York, 1974) 1–21.