MATHEMATICS OF COMPUTATION, VOLUME 32, NUMBER 144
OCTOBER 1978, PAGES 1115–1122

Limiting Precision in Differential Equation Solvers. II: Sources of Trouble and Starting a Code*

By L. F. Shampine

Abstract. The reasons a class of codes for solving ordinary differential equations might want to use an extremely small step size are investigated. For this class the likelihood of precision difficulties is evaluated and remedies examined. The investigation suggests a way of selecting automatically an initial step size which should be reliably on scale.

1. Introduction. In [1] we considered a machine dependent limit on the step size and one on the local error tolerance when solving ordinary differential equations. The second limit can be stated simply as that it makes no sense to ask for an answer more accurate than the correctly rounded value in the precision being used. Two kinds of remedy are possible. One is just to ask for less accuracy, which is not an unreasonable action since the difficulty often arises by accident. The other is to augment the precision either by increasing the word length or by additional algorithmic devices. The latter are the more interesting and may be the only practical option. Some references to such devices were given in [1]. In this paper we shall examine the other machine dependent limit which is on the step size. We shall first investigate *why* too small a step size should appear necessary. It turns out that the situation must occur much less frequently than one might expect. This investigation is of fundamental importance to understanding practical computation. After isolating the main source of difficulty, we shall show that a change in one's point of view avoids the difficulty. This observation is of considerable value in those situations for which the niceties of error control yield precedence to computing (nearly) always a reasonable solution. Lastly, we apply the techniques developed to the question of providing a highly reliable automatic choice of an initial step size.

2. Assumptions. The problem to be solved is

$$\begin{aligned} y_1' &= f_1(x, y_1, \ldots, y_N), \qquad y_1(a) = A_1, \\ &\vdots \\ y_N' &= f_N(x, y_1, \ldots, y_N), \qquad y_N(a) = A_N. \end{aligned} \tag{1}$$

The numerical method employed is assumed capable of integrating correctly a prob-

Received December 1, 1977.
AMS (MOS) subject classifications (1970). Primary 65L05, 65G05.
Key words and phrases. Limiting precision, minimum step size, roundoff, initial step size.
*This work was supported by the U. S. Department of Energy (DOE), Contract No. AT(29-1)-789.

lem with each f_i constant. Only codes based on the control of the error per step are studied. Most of the best codes satisfy these assumptions [2]. In controlling the error, one forms an estimate le_i of the local truncation error in the ith solution component on the step from x_n to $x_{n+1} = x_n + h$ and insists that

$$\left(\sum_{i=1}^{N} \left(\frac{le_i}{w_i} \right)^2 \right)^{1/2} \leqslant \epsilon.$$

Here the w_i are weights and ϵ the error tolerance, all of which the user supplies. For convenience we shall write expressions like the preceding as $\|le\|_w \leqslant \epsilon$. We remark that similar results can be obtained using norms other than the Euclidean. We shall assume that the estimator of the local error has the (vector) form

$$le = h \sum_m \alpha_m f(x_m, z_{1,m}, \ldots, z_{N,m}),$$

where we shall abbreviate

$$f_m = f(x_m, z_{1,m}, \ldots, z_{N,m})$$

in the future. The arguments of the function f_m are specified by the estimator and may, or may not, represent previously computed solution values. We shall assume the estimator is asymptotically correct [3]. As an example, if we are using Euler's method, a standard error estimator is

$$(2) \qquad le = (h/2)[f(x_{n+1}, y_{1,n+1}, \ldots, y_{N,n+1}) - f(x_n, y_{1,n}, \ldots, y_{N,n})].$$

Another obvious class of examples with fixed constants α_m consists of typical estimators for Runge-Kutta methods [3]. In the case of methods with memory, e.g. the Adams methods, the α_m may depend on the mesh spacing, see for example [4, Chapters 4, 6]. In such a case we shall presume that matters have been arranged so that the α_m are uniformly bounded, as they have been in the DE/STEP, INTRP suite of codes. The form of error estimator postulated includes all the common procedures, and we shall need no further details of the estimator.

3. Sources of Difficulty with the Precision. The usual reasons cited for needing a very small step size are:

(i) a lack of smoothness of the equation, including possible discontinuities,

(ii) requesting an error control relative to a solution component passing through zero,

(iii) a solution component changing very rapidly, including approach to a singularity.

We shall consider each situation in turn.

When using an error per step control we think (i) rarely leads to a step size too small. The test to be passed is

$$(3) \qquad |h| \left\| \sum \alpha_m f_m \right\|_w \leqslant \epsilon.$$

If the functions are merely continuous, our hypotheses imply that $\| \Sigma \alpha_m f_m \|_w \longrightarrow 0$ as $|h| \longrightarrow 0$ so that the estimated local error is $o(h)$. Even with a jump discontinuity present the estimate is $O(h)$. Clearly, the step size need not be much smaller than the tolerance except in extraordinary cases. If one takes into account roundoff errors in the formation of the estimate, the situation is not quite so nice, but in all cases one finds a lack of smoothness is not a likely source of failure.

The tests of [2] report the behavior of several kinds of codes which satisfy our assumptions when confronted with mild, pp. 387–388, and severe, pp. 403–404, lack of smoothness. They confirm our prediction that even the most severe lack of smoothness will not ordinarily cause precision difficulties. (They also show that an error per unit step control behaves quite differently in this situation.)

To investigate (ii) we can confine our attention to the case of a single equation and a zero in the solution at some point x_n. Suppose that at x_n the solution $y(x)$ has a zero of order m so that

$$y(x_n + h) = \frac{h^{m+1}}{(m+1)!} y^{(m+1)}(x_n) + O(h^{m+2}).$$

Assume the method of order p has a local error expansion of the form

$$y(x_n + h) = y_{n+1} + h^{p+1}\phi(x_n, y(x_n)) + O(h^{p+2}).$$

One wants to pass the test

$$|(y(x_n + h) - y_{n+1})/y(x_n + h)| \leqslant \epsilon.$$

It is clear that if $m < p$, this is possible; if $m = p$, in general this is not possible for all ϵ; and if $m > p$, in general this is not possible at all. Accordingly, if $m \geqslant p$, we may fairly diagnose the situation as an unreasonable one for the method being used; and we should hope that a minimum step size test would detect it. In point of fact, we think such a difficulty very uncommon in practice. The author does not recall ever seeing this difficulty arise due to a zero of high order nor has any of his colleagues or associates ever mentioned such an example to him. What *does* happen with some regularity is that one is using a variable order Adams code or backward differentiation code and gets into trouble because of a zero initial value. This is mainly due to the coding, and we are going to presume here that the program has some protection against an "accidentally" small numerical value being used in the error test. A genuine difficulty *can* occur because such codes begin with $p = 1$, and the author has seen problems with a double zero at the initial point. In [5] we considered the handling of this matter in the starting phase of such codes. After the start the codes go to high orders, and zeros are then no particular problem. Another real possibility is that the above analysis is not relevant because the solution is not in principle zero; but its computed representation *is*, due to underflow. We do not wish to state the matter too strongly because of the situations just mentioned, but we do feel that pure relative error control is not nearly as dangerous as many conceive it to be. If a difficulty should arise, one might fairly argue that this whole source of trouble is due to an

improper match of code and error criterion which can be cured by changing one or the other.

Lastly, we come to the case of a small step size caused by a rapid change in some solution component. This situation can and does occur with some regularity. One way to view it is that the wrong coordinate system is being used. In [4, pp. 274–278] we suggest some alternatives. If we were to solve (1) with arc length, s, as the independent variable, a number of advantages would accrue. In this variable the system (1) becomes

$$\begin{aligned} dy_1/ds &= f_1/S = F_1, \\ &\vdots \\ dy_N/ds &= f_N/S = F_N, \\ dx/ds &= 1/S = F_{N+1}, \qquad x(0) = a. \end{aligned}$$

Here $S = (1 + \Sigma_{i=1}^{N} f_i^2)^{1/2}$. Obviously $|F_i| \leqslant 1$, $i = 1, 2, \ldots, N+1$, for the entire interval in s. Working in arc length practically does away with the difficulty of a minimum step size when using an error per step control. For example, supposing that we wish to use a pure absolute error control, $w_i = 1$ each i, we see from (3) that the error test will surely be passed if

$$h \leqslant \frac{\epsilon}{\Sigma|\alpha_m|} \leqslant \frac{\epsilon}{\Sigma|\alpha_m|\,\|F_m\|}.$$

Thus, we can say, *a priori*, that when using Euler's method with the specified error estimator in the circumstances postulated, it is never necessary to use a step size smaller than ϵ. When using the Fehlberg (4, 5) formulas, a little calculation shows that it is never necessary to go below 8ϵ.

Another possibility which is less smooth in its behavior but is attractive for one step methods is at x_n to scan the derivatives to find

$$\left|\frac{dy_j}{dx}\right| = \max_i \left|\frac{dy_i}{dx}\right| = \max_i |f_i(x, y_1, \ldots, y_N)|.$$

If we were to introduce y_j as the new independent variable at this point and so solve

$$\begin{aligned} \frac{dy_1}{dy_j} &= \frac{f_1(x, y_1, \ldots, y_N)}{f_j(x, y_1, \ldots, y_N)} = F_1(x, y_1, \ldots, y_N), \\ &\vdots \\ \frac{dx}{dy_j} &= \frac{1}{f_j(x, y_1, \ldots, y_N)} = F_j(x, y_1, \ldots, y_N), \\ &\vdots \\ \frac{dy_N}{dy_j} &= \frac{f_N(x, y_1, \ldots, y_N)}{f_j(x, y_1, \ldots, y_N)} = F_N(x, y_1, \ldots, y_N), \end{aligned}$$

we would have the magnitude of each F_i roughly bounded by 1 for a short distance. It is easy to see that if some slope is much larger than the rest, this change of variables does approximately what arc length does.

The interchange of independent and dependent variables shows a little more clearly than with arc length what is happening. If, say, dy_j/dx is very large, we expect (though it is not *necessarily* true) that a small step size in x will be needed. With the interchange of variables, dx/dy_j is very small and we expect that a large step in y_j is possible. The efficiency of these transformations seems unpredictable and our experiments over many years have not detected any significant advantage. Thus, we emphasize that the important point here is that the behavior with respect to machine precision is altered, not that the problem is solved more cheaply. There is one possibility for increased efficiency. When using arc length one can reasonably expect his mesh to suffer much less distortion in the presence of near discontinuities of the solution. Some implementations of the Adams methods, e.g. [4], compute the coefficients of the integration formulas so as to account for variable step size. The overhead of doing this is significant and because it can be reduced by taking advantage of steps of the same size, the selection of the step size is biased towards a constant value. Reducing the mesh distortion by transformation could reduce the overhead in such codes by making constant step size more likely to be acceptable. A more important implication is that backward differentiation formulas with interpolatory step size changing can become unstable in the presence of extreme mesh distortion. Also, being essentially a fixed step procedure, other aspects of the basic algorithms and the overhead are adversely affected. In [6] an example of a photocatalyzed reaction is given which causes the GEAR package to fail, this justifying the use of a more sophisticated step size changing procedure in EPISODE. With arc length as an independent variable, GEAR will integrate this problem efficiently.

An interesting example of the situation at hand appears in a paper [7] describing the numerical solution of a model of a cavitating bubble. The authors describe in some detail the cusp-like behavior of the solution at certain points where the bubble bounces back. They employ a variety of techniques for coping with the difficulty of machine precision. At this author's suggestion the integration was tried with arc length as an independent variable and the integration proceeded without novelty. We have also solved their problem using STEP, an Adams code, with arc length as the independent variable and RKF45, a Runge-Kutta code, using interchange of dependent and independent variables. Both codes have a machine dependent minimum step size as specified in [1] and because of it will not pass the first cusp when given the original problem. They integrate the problem easily with the changes described.

The changes of variable specified get around the difficulty of a rapidly changing solution but we have not been entirely fair with the reader. The mathematical problems being solved are identical, but the computational problems are not. The essential difference is that the usual scheme allows *no* error in the independent variable x whereas the changes of variable do permit errors in x. Thus, when solving a single equation with a pure absolute error tolerance ϵ_1, we are saying that at a given point

x_n we shall be satisfied with any approximate solution y_n such that $|y_n - y(x_n)| \leqslant \epsilon_1$. When going to arc length we must specify a reasonable tolerance ϵ_2 on the variable x. Thus, we shall be satisfied with any pair x_n, y_n such that $|x_n - x(s_n)| \leqslant \epsilon_2$, $|y_n - y(x(s_n))| \leqslant \epsilon_1$. After such a change of variables, the code has an extra bit of freedom which one might regard as the reason it can perform better. We think of this freedom as being much the same as allowing some absolute error near a zero of the solution when the integration is basically intended to have a relative error control. It is not our intent to describe these changes of variable as a panacea for precision difficulties, but they do furnish a valuable alternative provided the user understands how he is altering the problem. In contexts like simulation it may be appropriate to regard the integration as a "black box" which produces reasonable approximations very reliably. Such a black box can be written pretty easily by insisting some relative accuracy be used to avoid the first machine dependent limit of [1], using moderate error tolerances, using error per step control to cope with the difficulties described earlier in this paper, and interchanging dependent and independent variables as just described. (We comment that the last action also makes it easy to terminate the integration at a given value of any dependent variable which is itself a valuable feature.) Our investigation says that such a code could be very robust indeed by present standards.

4. Starting a Code. We can make an interesting application of these ideas to the important problem of the automatic selection of the initial step size. In [5] we tried to approximate a suitable value. Although the approximation is crude quantitatively, it does exhibit many desirable qualitative properties. As far as reliability is concerned, the important thing is to choose a step size small enough. Sedgwick [8], in fact, starts every integration with the smallest step size permissible in the precision used and works up from there. This is obviously inefficient and there is some danger that in one's eagerness to increase the step size rapidly he will permit a growth rate so large that he skips over the phenomena which were the justification for starting with a step size so small in the first place. Furthermore, there is the worry that the initial step size is too small for the error estimator (and other algorithms) to be reliable. The idea is a good one, we just think it a bit extreme. It was not described as such but we have already derived in this paper a suitable procedure when the problem has been formulated in terms of arc length. We can choose an initial step size that we are certain will succeed and which is comparable in size to the tolerance specified. This is most satisfactory. Our proposal is to use either of the two changes of variables to take a step which we can be sure will succeed. Then we see how far we advanced in the original independent variable and use this quantity to initiate the integration of the original problem. The reason for, in effect, repeating the first step is that in the new variables we must specify a tolerance on the computation of the old independent variable. Thus, on our return to the original variables we are not certain that the step size we deduced will succeed. However, it is very likely indeed that we shall be on scale. This is especially true if our trial step uses a low order method. For this reason and that of extreme convenience we explore a procedure based on

Euler's method. We point out to the reader that the important Adams codes and backward differentiation codes actually do start with Euler's method so the resulting initial step size should not be absurdly inefficient.

If we were to go to arc length and take a step of length Δs, the error test would be $\|le\|_w \leqslant \epsilon$. A reasonable weight for x is to make its error relative to the distance to the first output point b so as to account for the scale of the independent variable. From (2) and the fact that each $|F_i| \leqslant 1$ we find that the test will surely be passed if

$$|\Delta s|\,\|e\|_w \leqslant \epsilon,$$

where e is the vector of $N+1$ components each of which is 1. Using $|\Delta s| = \epsilon/\|e\|_w$, we shall not actually need to form the error estimate. Furthermore, we wish only to advance the variable x so that we just need to compute

$$x_1 = a + \Delta s F_{N+1}(a, y_1(a), \ldots, y_N(a));$$

hence

$$h = x_1 - a = \frac{\Delta s}{S} = \frac{\epsilon}{\|e\|_w S}, \tag{4}$$

where

$$S = \left(1 + \sum_{i=1}^{N} f_i^2(a, y_1(a), \ldots, y_N(a))\right)^{1/2}$$

Notice the extremely simple and cheap computation involved here; we scarcely need more than the initial slopes.

For small ϵ and steep slopes one might well find the h of (4) to be smaller than the machine dependent minimum step size and in taking the larger value he does what Sedgwick has suggested. Ordinarily though, this scheme will result in step sizes which are not small compared to the machine precision. It seems to us to be about as reliable as Sedgwick's idea without being nearly so expensive. Indeed, the optimal step size when using Euler's method is $O(\sqrt{\epsilon})$. Clearly, (4) is conservative, as we wish it to be, but it is not ridiculously so. It is especially attractive for "black boxes" used at crude tolerances—a situation by no means uncommon. With codes starting at high orders and particularly at stringent tolerances, one might well prefer an approximate initial step size selected as in [5]. The decision mainly rests on how much one is willing to pay for enhanced reliability and what kind of cost might be expected in the working environment.

5. Conclusions. In our examination of the reasons for precision difficulties in a popular kind of code for solving ordinary differential equations, we found that this kind of code is surprisingly robust. In each case we have suggested some possibilities to the user of the code for avoiding such difficulties. The study has suggested a way of automatically selecting an initial step size which appears to be highly

reliable and not too inefficient. In conjunction with [1] the results of our investigation will be helpful to those needing to write unusually robust software for unsophisticated users or as low level parts of more complicated software.

Numerical Mathematics Division
Sandia Laboratories
Albuquerque, New Mexico 87115

1. L. F. SHAMPINE, "Limiting precision in differential equation solvers," *Math. Comp.*, v. 28, 1974, pp. 141–144.

2. L. F. SHAMPINE, H. A. WATTS & S. M. DAVENPORT, "Solving nonstiff ordinary differential equations–the state of the art," *SIAM Rev.*, v. 18, 1976, pp. 376–411.

3. L. F. SHAMPINE & H. A. WATTS, "Comparing error estimators for Runge-Kutta methods," *Math. Comp.*, v. 25, 1971, pp. 445–455.

4. L. F. SHAMPINE & M. K. GORDON, *Computer Solution of Ordinary Differential Equations*, Freeman, San Francisco, Calif., 1975.

5. L. F. SHAMPINE, *Starting an ODE Solver*, Report SAND 77–1023, Sandia Laboratories, Albuquerque, N. M., 1977.

6. G. D. BYRNE & A. C. HINDMARSH, "A polyalgorithm for the numerical solution of ordinary differential equations," *ACM Trans. Math. Software*, v. 1, 1975, pp. 71–98.

7. G. J. LASTMAN, R. A. WENTZELL & A. C. HINDMARSH, "Numerical solution of a bubble cavitation problem," *J. Computational Phys.* (To appear.)

8. A. E. SEDGWICK, *An Effective Variable Order Variable Step Adams Method*, Report 53, Dept. of Comp. Sci., Univ. of Toronto, Toronto, Canada, 1973.