MATHEMATICS OF COMPUTATION, VOLUME 28, NUMBER 125, JANUARY, 1974

Limiting Precision in Differential Equation Solvers*

By L. F. Shampine

Abstract. Machine dependent limits on the step size and local error tolerance are discussed. By taking them into account codes can be made more robust.

Robust software for the solution of differential equations should take reasonable action when a user requests more accuracy than it is possible to obtain in the machine's precision. Ideally, the maximum precision possible will be obtained and requests for more accuracy detected and handled without undue expense. One is inclined to think that good quality codes will terminate by reaching a minimum step size when impossible error requests are made, but this is not typical. It is common to find that as tolerances are decreased past limiting precision, routines start returning worse answers obtained at a rapidly increasing cost. Since this can be forestalled by a judicious use of a minimum step size, we discuss the role of this parameter. We shall present two simple requirements which greatly enhance the effectiveness of a wide variety of differential equation solvers and illustrate their use.

Every code should have a machine dependent minimum step size, though most do not. If a code is to advance from x to $x + h$, the floating-point number $x + h$ must be different from x. Unless $|h| \geqq u|x|$, where u is the unit roundoff of the machine, the numbers cannot be distinguished. In addition to this fundamental requirement, one has to consider the fact that the code will be calling a routine to evaluate the differential equation for various arguments. Adams codes evaluate only at x and $x + h$, so no additional requirement is placed on h but this is not true of all methods. For example, the Runge-Kutta-Fehlberg (4, 5) pair evaluates the differential equation at $x + 12h/13$ and $x + h$. Even if one evaluates these arguments exactly, their difference of $h/13$ must be greater than a unit roundoff in order to distinguish them so one must require at least $|h| \geqq 13u|x + h|$. The details of the method permit one to determine rather precise limits on the step size in this way. This is unduly optimistic though, since one needs a few digits' difference between arguments to distinguish them in practice, just how much depending on the differential equation and how it is programmed. In the variable order code STEP [1], which uses a divided difference form of the Adams methods, we have required

$$|h| \geqq 4u|x| \tag{1}$$

to provide a measure of additional protection.

Received March 19, 1973.

AMS (MOS) subject classifications (1970). Primary 65L05, 65G05.

Key words and phrases. Limiting precision, local error, minimum step size, robust software, roundoff.

* This work was supported by the United States Atomic Energy Commission.

To illustrate limitations, we use the problem of two bodies in circular motion

$$y_1' = y_2, \qquad y_1(0) = 0,$$
$$y_2' = -y_1, \qquad y_2(0) = 1, \tag{2}$$

which is to be integrated over $[0, 16\pi]$ using an absolute error test. An IBM 360/67 in single precision has $u \doteq$ 9.5E-7 and a CDC 6600 has $u \doteq$ 7.1E-15. Using an Adams code with the requirement (1), we find that at 16π the step size must be at least 1.9E-4 and 1.4E-12, respectively, on these machines.

Many codes allow the user to specify a minimum step size, the principal aims being to control the work and to detect trouble spots during an integration. As applied by a sophisticated user, especially during repeated integrations of a problem, control of the minimum step size can be informative. However, we feel few users care about the step sizes taken as long as their code solves the problem with adequate accuracy and reasonable cost. Few users have the knowledge about their problem to specify a good minimum step size, especially since it changes as an integration progresses. This objection is particularly forceful in the context of variable order Adams codes which may require extremely small steps in a perfectly ordinary computation. In this context, a minimum step size is singularly unsuccessful in controlling the work since it can allow enormous amounts of work with an entirely reasonable minimum step size. Counting function calls or steps is so easy and precise as a control on the work that this justification of a minimum step size need not be taken seriously. For these reasons, a number of codes, including those of the author [1], [3], do not allow the user to specify a minimum step size. Regardless of the code structure in this respect, if a user cannot be expected to choose reasonable error tolerances, he can hardly be expected to choose an appropriate minimum step size.

The problem to be solved is

$$\begin{aligned} y_1' &= f_1(x, y_1, y_2, \cdots, y_N), & y_1(a) &= A_1, \\ y_2' &= f_2(x, y_1, y_2, \cdots, y_N), & y_2(a) &= A_2, \\ &\vdots \\ y_N' &= f_N(x, y_1, y_2, \cdots, y_N), & y_N(a) &= A_N. \end{aligned}$$

We are concerned with numerical methods which advance from approximate solution components $y_n^i \doteq y_i(x_n)$ to $y_{n+1}^i \doteq y_i(x_n + h)$ by a formula of the form $y_{n+1}^i = y_n^i + h\Phi^i$. Error is measured with respect to a weight vector with components w^i in a suitable norm, usually L_2 or L_∞. To be specific, let us suppose L_2 is used since it is the more common. The user specifies a local error tolerance ϵ. The local error in the ith solution component in stepping to x_{n+1} is estimated by le_{n+1}^i and controlled so that

$$\left(\sum_{i=1}^{N} (\mathrm{le}_{n+1}^i / w^i)^2\right)^{1/2} \leqq \epsilon$$

at each step. For convenience, we shall write expressions like this as $||\mathrm{le}||_w \leqq \epsilon$. (This is error per step control; the analysis is much the same for error per unit step, $||\mathrm{le}||_w \leqq |h|\epsilon$, which is the other common criterion.)

A local error of no more than ϵ is permitted in advancing from x_n to x_{n+1} in

exact arithmetic. There is, however, an additional error in the step due to the use of floating-point arithmetic. We cannot ask for more than the correctly rounded values $y_{n+1}{}^i$, which is to say, we cannot anticipate that the addition of $y_n{}^i$ to $h\Phi^i$ can be done with less than a unit roundoff in the answer $y_{n+1}{}^i$. Adjusting the step size affects only the discretization error of the method, $\mathrm{le}_{n+1}{}^i$. Consequently, it is pointless for a user to ask a code to adjust h in an attempt to make the total error arising in a step less than the roundoff error from the addition. That is, if $\epsilon < u||y_{n+1}||_w$, the user has clearly asked for too much accuracy. Indeed, unless the local error permitted is rather larger than this roundoff error, one will see no effect due to changing the step size. So, one ought, for example, to keep $\epsilon \geqq 2u||y_{n+1}||_w$.

There is room for considerable variation in applying this test. It is inconvenient to work with $y_{n+1}{}^i$ since it is not immediately available. Typically, at limiting precision, $|h\Phi^i| \ll |y_n{}^i|$ so that $|y_{n+1}{}^i| \doteq |y_n{}^i|$ gives a convenient substitute. It is also possible to actually estimate the error of addition in forming $y_{n+1}{}^i$ but it is better to be conservative and to assume a unit roundoff. After trying various possibilities, we found a simple application to work about as well as any. An important consideration is that production codes choose their step sizes so that the predicted error is less than a fraction of ϵ to reduce failures. For example, STEP and the variable order code DVDQ [2], which uses the difference form of the Adams methods, aim at 0.1 ϵ and accept errors of ϵ. The Runge-Kutta-Fehlberg code RKF [3] uses 0.8 times the step estimated to give an error of ϵ. Since it is a fourth-order code using error per unit step, this corresponds to aiming at $(0.8)^4\epsilon$. The code DIFSUB [4] using the Nordsieck form of the Adams methods varies the fraction depending on the order and on whether the order is being changed. This is particularly inconvenient for our test. When at order k it aims at $(1/1.2)^{k+1}\epsilon$; an adequate test can be devised by ignoring potential changes of order during the step.

The test we suggest is that if a user calls a code with a tolerance such that the code will aim at an error smaller than $2u||y_n||_w$ in stepping from x_n to $x_n + h$, the code should reject the tolerance and report a value which is permissible. There are two main reasons for this second test. On machines with short word lengths it is easy to ask for too much accuracy accidentally. If one seeks maximum accuracy, the test allows him to ask the integrator for the best it can do. As an example, if STEP or DVDQ is called to solve (2), the smallest permissible tolerance ϵ satisfies $0.1\epsilon = 2u\,||y_0||_w = 2u$ hence $\epsilon =$ 1.9E-6 on the IBM 360/67 and 1.4E-14 on the CDC 6600. For this particular problem, the true solution has $y_1{}^2(x) + y_2{}^2(x) \equiv 1$, hence $||y_n||_w \doteq 1$ for all n and this limit on ϵ holds at every step. In general, one must ascertain an appropriate limit at each step.

The problem (2) is one of Krogh's set of test problems [5]. Illustrative computations using DIFSUB and a CDC 6600 on the whole set can be obtained from the author. DIFSUB was used because it is widely known and easily available. To make testing easy, we put a short code "in front of" DIFSUB which imposed the minimum step size (1) and made optional the use of the second requirement, namely that the code not aim at a tolerance smaller than $2u||y_n||_w$. In all the test problems, the maximum accuracy possible is obtained and the cost is stabilized. We report here the results with and without the optional control on the tolerance ϵ for problem (2). The errors are the maximum encountered at any step during the integration over the specified interval. Cost is measured by the number of derivative evaluations ND.

	$(0, 2\pi)$		$(0, 6\pi)$		$(0, 16\pi)$	
ϵ	*error*	ND	*error*	ND	*error*	ND
			without control			
1E-12	3E-10	493	3E-9	1348	3E-8	3487
1E-13	2E-11	631	2E-10	1774	2E-9	4624
1E-14	7E-11	1617	6E-10	4924	4E-9	13023
1E-15	5E-10	9216	4E-9	28206	3E-8	75409
			with control			
1E-12	3E-10	493	3E-9	1348	3E-8	3487
1E-13	2E-11	632	2E-10	1772	2E-9	4619
1E-14	5E-12	674	6E-11	1871	6E-10	4868
1E-15	3E-12	656	7E-11	1849	6E-10	4826

We stress that there are other limitations imposed by the precision and that the two pointed out here are simple necessary conditions and no panacea. Other devices which depend on the method, e.g. [6], [7], [8], for detecting and controlling the effects of limited precision should also be considered when very accurate results are desired. The codes RKF and STEP have the first, respectively both, requirements written into them. The usefulness and effectiveness of the two tests have been demonstrated in steady use of these codes in both academic and industrial environments. Considering how easy they are to incorporate and how cheap they are, we commend the tests for general use.

Applied Mathematics Division 5121
Sandia Laboratories
Albuquerque, New Mexico 87115

1. L. F. SHAMPINE & M. K. GORDON, *Computer Solution of Ordinary Differential Equations: Initial Value Problems*, Freeman, San Francisco, 1974.
2. F. T. KROGH, VODG/SVDQ/DVDQ—*Variable Order Integrators for the Numerical Solution of Ordinary Differential Equations*, TU Doc. No. CP-2308, NPO-11643, May 1969, Jet Propulsion Laboratory, Pasadena, California.
3. L. F. SHAMPINE & R. C. ALLEN, *Numerical Computing: An Introduction*, Saunders, Philadelphia, Pa., 1973.
4. C. W. GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, N.J., 1971.
5. F. T. KROGH, "On testing a subroutine for the numerical integration of ordinary differential equations," *JACM*. (To appear.)
6. F. T. KROGH, *Changing Stepsize in the Integration of Differential Equations Using Modified Divided Differences*, Proc. Conference on the Numerical Solution of Ordinary Differential Equations (Austin, Texas, Oct. 1972), Lecture Notes in Math., Springer-Verlag. (To appear.)
7. E. VITASEK, *The Numerical Stability in Solution of Differential Equations*, Proc. Conf. Numerical Solution of Differential Equations (Dundee, Scotland, 1969), Springer, Berlin, 1969, pp. 87–111. MR **42** #2681.
8. E. K. BLUM, "A modification of the Runge-Kutta fourth-order method," *Math. Comp.*, v. 16, 1962, pp. 176–187. MR **26** #3190.