ELSEVIER Journal of Computational and Applied Mathematics 58 (1995) 345–354

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

A simple step size selection algorithm for ODE codes

L.F. Shampine[a], A. Witt[b,*]

[a] *Mathematics Department, Southern Methodist University, Dallas, TX 75275, United States*
[b] *Department of Mathematics and Computer Science, Austin Peay State University, Clarksville, TN 37044, United States*

Received 8 June 1993; revised 14 December 1993

Abstract

A very simple way of selecting the step size when solving an initial problem for a system of ordinary differential equations (ODEs) is examined. It is shown to control the true error and the stability of the integration. In a gross measure it is shown to be as efficient as control of the local error. Numerical results illustrate the analysis and explore the robustness of the approach.

Keywords: Ordinary differential equations; Step size selection; Stability; Efficiency

1. Introduction

Discrete variable methods for the solution of an initial value problem for a system of ordinary differential equations (ODEs),

$$\boldsymbol{y}' = \boldsymbol{F}(x,\boldsymbol{y}), \quad a \leqslant x \leqslant b, \ \boldsymbol{y}(a) = \boldsymbol{y}_0,$$

step from a to b producing approximations $\boldsymbol{y}_j$ to $\boldsymbol{y}(x_j)$ on a mesh $a = x_0 < x_1 < \cdots$. Effective codes estimate the error made in the step from x_{j-1} and adjust the step size $x_j - x_{j-1}$ accordingly. There are several reasons for doing this, one being to gain confidence that the problem has been solved in some reasonable sense. Another is to stabilize the integration. A third is to make possible the solution of problems that are hard because an adequate representation of the solution requires a very much smaller step size in some portions of the interval than in others. A fourth is to solve problems more efficiently by using large step sizes on those portions of the interval where the solution is relatively easy to approximate.

It was recognized early that adaptation of the step size to the solution is advantageous. In the case of linear multistep methods, it was clear enough how to accomplish this, but Runge–Kutta

* Corresponding author. e-mail: witta@lynx.apsu.edu.

SSDI 0377-0427(94)00007-N

and other one-step methods were a different matter. A general approach that is still used in some contexts is first to introduce arc length as the independent variable and then to integrate with a constant step size. The approach endows any discrete variable method with the capability of solving hard problems with tolerable efficiency, but in this simple form it does not provide the other capabilities desired of a step size control. Eventually it was learned how to estimate and control the local error of one-step methods. Control of the local error has become generally accepted, but it is not all that one might want. For one thing, it is relatively expensive to estimate the local error of a given formula by a general scheme such as extrapolation. This led a number of authors, e.g., [9, 20, 24], to seek cheaper ways of adapting the step size to the solution when using the classical four stage, fourth order Runge–Kutta formula (RK4). For another, the popular Runge–Kutta codes do local extrapolation so as to amortize the cost of the local error estimate. This means that the step size is adapted to a formula different from the one used to approximate the solution, hence is not really optimal. Nowadays the derivations of formula and error estimator are interwined. It can be difficult to achieve simultaneously both an effective formula and a satisfactory error estimate, especially when the aim is to solve stiff problems. For instance, Kaps and Rentrop [5] found that within the family of formulas they investigated, they could not obtain the kind of damping at infinity that they wanted in both the formula and its error estimator. Another instance is provided by a series of papers [2, 8, 15, 23] that consider how to obtain a satisfactory estimate for an attractive semi-implicit formula found in [21].

A general approach seen early is to control the step size so that the size of the change in the solution is approximately constant from step to step. Though never widespread, physical scientists have employed it from time to time, recent examples of its use in books being [1, Appendix A] and [3, Appendix 1] to control the error of RK4. There do not seem to have been any previous attempts to analyze this control, perhaps because it has been assumed that it behaves much like using a constant step size in arc length. As we shall see, control of the change in the solution provides all the capabilities desired of a step size control. Our analysis is complemented by numerical examples computed with a number of formulas, including RK4 for nonstiff problems and Wolfbrandt's second order, semi-implicit formula for stiff problems.

Control of the change in the solution is interesting because it is so simple and because it applies to virtually any discrete variable method. The analysis of this paper shows that it is a reasonable way to control the step size, though not as efficient as control of the local error. We do not suggest that this way of selecting the step size replace those seen in popular codes. Rather we observe that it is useful when no satisfactory local error estimator is known, when it is desired that the control apply to the formula used to approximate the solution, and when a premium is placed on simplicity.

2. Analysis

In this section we begin by seeing that controlling the change in the solution is easy to implement and relating it to other ways of controlling the step size. We then prove that control of the change in the solution "works" by establishing convergence in reasonable circumstances. Indeed, we establish an asymptotic expansion for the error that we exploit to investigate the efficiency of the step sizes chosen. Finally we argue that the control stabilizes the numerical integration.

2.1. Step size adjustment

In this step from x_j to x_{j+1}, the goal is to determine the largest $h_j = x_{j+1} - x_j$ for which the difference in the numerical solution is no greater than a given tolerance τ. For any discrete variable method of the form $\boldsymbol{y}_{j+1} = \boldsymbol{y}_j + h_j\boldsymbol{\Phi}$, consistency implies that for "small" h_j, the increment function $\boldsymbol{\Phi}$ is related to the local solution $\boldsymbol{u}(x)$ by $\boldsymbol{\Phi} \approx \boldsymbol{u}'(x)$. Except at points where $\boldsymbol{u}'(x)$ vanishes, this implies that the change in the numerical solution is proportional to the step size. This simple behavior makes it easy to adjust the step size so as to make the change approximately equal to τ. It also makes it easy to select automatically an initial step size. Indeed, the first algorithm [16] for automatic selection of the initial step size begins by selecting a step size in just this way.

If all is going well, the step size h_j is such that $\tau \approx \|\boldsymbol{y}(x_{j+1}) - \boldsymbol{y}(x_j)\| \approx h_j\|\boldsymbol{y}'(x_j)\|$, hence $h_j \approx \tau\|\boldsymbol{y}'(x_j)\|^{-1}$. It is illuminating to relate this to arc length. When the variable x is replaced in the system $\boldsymbol{y}'(x) = \boldsymbol{f}(x,\boldsymbol{y}(x))$ by arc length s, the system is augmented with the equation $\mathrm{d}x/\mathrm{d}s = (1 + \|\boldsymbol{f}\|_2^2)^{-1/2}$. A constant step size of δ in s corresponds to a change in x of $h_j = x_{j+1} - x_j \approx \delta(1 + \|\boldsymbol{y}'(x_j)\|_2^2)^{-1/2}$. If an integration is done with the change in the solution controlled to be about τ in the Euclidean norm, it is clear that where the solution changes rapidly in the sense that $\|\boldsymbol{y}'(x)\|_2 \gg 1$, the step chosen is essentially the same as that provided by integration in arc length with a constant step size of τ. On the other hand, where the solution changes slowly in the sense that $\|\boldsymbol{y}'(x)\|_2 \ll 1$, controlling the change in the solution results in a step size much larger than the τ of arc length.

The research code used to compute the numerical examples reported here is based on a weighted maximum norm. The weights correspond to a mixed relative-absolute error test with both tolerances being scalars. The average magnitude of a solution component at the two ends of the step is used in the relative error test. Step sizes are not permitted to increase at one time by more than a factor of 5 nor decrease by more than a factor of 0.5. The step size is chosen so that the change in the solution is predicted to be 0.8τ. The efficiency of the algorithm would benefit from refinements like those in [22], but a simple scheme suffices for illustrative purposes.

2.2. Convergence and asymptotic behavior

Shampine [12] develops an approach to the analysis of step size selection schemes and their efficiency that we apply here. The convergence theory for (fixed order) variable step methods assumes that the step size is specified in terms of a step selection function, a piecewise continuous function $\theta(x)$ such that $0 < \xi \leqslant \theta(x) \leqslant 1$ for $a \leqslant x \leqslant b$. At a mesh point x_j, the step size is given in terms of a step size selection function and a maximum step size H as $h_j = \theta(x_j)H$. With the usual assumptions about the problem and the numerical method, it then follows that the method converges and

$$\boldsymbol{y}_j = \boldsymbol{y}(x_j) + H^p\boldsymbol{e}(x_j) + \mathcal{O}(H^{p+1}).$$

The theory supposes that the step sizes are specified in advance. To model practical computation we must ask whether the usual ways of selecting a step size automatically lead to step sizes that can be described by a step size selection function. For this purpose it is important to appreciate that the asymptotic expression for the error holds for step sizes that are not quite those given by a step size selection function, namely when $h_j = \theta(x_j)H + \mathcal{O}(H^2)$.

We have already observed that if all is going well, $h_j \approx \tau/\|y'(x_j)\|$. Notice that to this degree of approximation, the step size does not depend on the formula. Of course, if $\|y'(x_j)\| = 0$, some other rule for selecting the step size must come into play. All step size selection algorithms must provide for situations like this. Typically a bound on the rate of change of the step size is used to get past an isolated point of this kind. For analytical purposes we suppose that such ad hoc rules do not come into play by assuming that

$$y'_{\min} = \min_{a \leq x \leq b} \|y'(x)\|$$

is positive. Let $H = \tau/y'_{\min}$. With the assumptions made about the problem, it is easily verified that $\theta(x) = y'_{\min}/\|y'(x)\|$ is a step size selection function and H is the maximum step size. Accordingly, numerical integration with step sizes $h_j = \theta(x_j)H = \tau/\|y'(x_j)\|$ converges as $H \to 0$ (equivalently, as $\tau \to 0$). The error expansion stated earlier is valid, implying that the true, or global, error is $\mathcal{O}(\tau^p)$.

Having defined a step size selection function $\theta(x)$, we now must argue that it describes reasonably well the step sizes chosen automatically in codes that control the change in the solution. Using first the error expansion and then the definition of the step size, it is seen that the change in the numerical solution satisfies

$$\|y_{j+1} - y_j\| = \|y(x_{j+1}) - y(x_j) + \mathcal{O}(H^p)\| = h_j\|y'(x_j)\| + \mathcal{O}(h_j^2) = \tau + \mathcal{O}(\tau^2).$$

This says that for small tolerance τ, the step sizes given by this step size selection function do lead to almost constant changes of τ in the numerical solution.

There are a number of variants on control of the local error that are seen in popular codes. The error of a formula of order p is estimated by comparison to a formula of order $p+1$. The criterion of error per step, EPS, requires that the estimated error be no larger than a given tolerance at each step. That of error per unit step, EPUS, requires that the error be no larger than the product of the step size and the tolerance. The idea of local extrapolation is to control the error of the formula of order p, but to advance the integration with the higher order formula used to estimate the error. Accordingly, XEPS is error per step control with local extrapolation and XEPUS is error per unit step control with local extrapolation.

It is important to understand how the error depends on the tolerance. For control of the change in the solution, we have just seen that the global error is proportional to τ^p. It is shown in [12] that for control of the local error, the global error is proportional to τ for both the EPUS and XEPS, proportional to $\tau^{p/(p+1)}$ for EPS, and proportional to $\tau^{(p+1)/p}$ for XEPUS. It is perhaps ideal that the error be proportional to the tolerance as for EPUS and XEPS. The behavior on a change of τ is less obvious for EPS and XEPUS when the order p is low, especially when the code varies the order in the course of the integration. Of course, as suggested in [14], it is possible for codes to use an internal tolerance chosen to take the global error proportional to the tolerance given the code. This could be done for control of the change of the solution as well as for control of the local error by EPS and XEPUS. If something like this is not done, a decrease of one order of magnitude in τ will result in a decrease of p orders of magnitude in the global error. This contrasts with a reduction of about one order of magnitude when the local error is controlled.

One of the things that is different about control of the change in the solution is that the step size selection function in our model is independent of the formula. To illustrate our conclusion that for stringent tolerances the step size does not depend on the formula, we solved the standard two body

problem D1 [4] using minimal stage Runge–Kutta formulas of orders 2, 3 and 4, specifically the improved Euler formula, Ralston's formula [7] of order 3, and RK4. The relative and absolute error tolerances were both taken to be 0.01. For each formula, the maximum step size, HMAX, was calculated and an experimental step size selection function was computed at mesh points as $\theta(x_j) = (x_{j+1} - x_j)/\mathrm{HMAX}$ and at intermediate points by linear interpolation. The graphs of these $\theta(x)$ are indistinguishable. The average step size was about 10^{-2} for all three integrations. The maximum difference in the step sizes chosen in the integrations at orders 2 and 3 was about 10^{-5}, in the integration at orders 2 and 4 about 10^{-5}, and in the integrations of orders 3 and 4 about 10^{-7}. This computation and others we have done confirm that the step sizes are independent of the formula whenever the tolerances are small enough that the asymptotic analysis is applicable.

2.3. *Efficiency*

One way to measure efficiency is by the cost of the integration for a given tolerance. For a formula with fixed cost per step, a conventional measure of the cost of integrating from a to b is the number of steps taken,

$$N = \sum_{j=1}^{N} \frac{1}{h_j} h_j.$$

Viewing the expression for N as a Riemann sum leads to

$$N \approx \int_a^b \frac{1}{\theta(t)H}\,\mathrm{d}t = \tau^{-1} \int_a^b \|y'(t)\|\,\mathrm{d}t.$$

Because H is proportional to τ, this approximation of the cost is proportional to τ^{-1}, regardless of the order of the method being used. When controlling the local error of a method of order p, the cost is proportional [12] to $\tau^{-1/p}$ for EPUS and XEPUS and proportional to $\tau^{-1/(p+1)}$ for EPS and XEPS. The cost is higher when controlling the change in the solution because for a given τ, the integration involves more steps than when controlling the local error. In this measure of efficiency no account is taken of the fact that the smaller step sizes result in more accuracy, and when it is, the approach becomes competitive.

A second measure of efficiency is to consider the cost as a function of the global error μ of the numerical solution. We have seen that when controlling the change in the solution, $\mu \sim \tau^p$, or put differently, to achieve a global error of μ we need a tolerance τ proportional to $\mu^{1/p}$. The cost of achieving this accuracy, as seen above, is proportional to τ^{-1}. The cost of achieving a global error of μ is therefore proportional to $\mu^{-1/p}$. When controlling the local error by the criterion of EPS or EPUS, the efficiency in this sense is also proportional to $\mu^{-1/p}$ [12]. In this gross measure of efficiency, control of the change in the solution is just as efficient as control of the local error. Local extrapolation provides a global error of μ at a cost proportional to $\mu^{-1/(p+1)}$, which is rather more efficient. But this is not a possibility with control of the change in the solution, because there is no companion formula of higher order used for estimating the local error.

No attention has been paid to the constants that arise in these comparisons. Experience with codes based on local error control says that the general conclusions drawn from such arguments are quite useful, but may not be valid for a given problem. We conclude that control of the change

in the solution "works" in the sense that automatic selection of the step size does control the error, the error behaves in a regular way, and in a standard measure of efficiency, the control is just as efficient as control of the local error. The fact that the step sizes selected do not depend on the formula makes it clear that they are not optimal for any given formula. A numerical example in Section 4 illustrates this.

3. Stability

It is generally recognized that control of the local error stabilizes a numerical integration. More specifically, instability is manifested in a growth of the local error that eventually causes the step size algorithm to reduce the step size until the integration is stable. The theory that supports this observation is fragmented and of restricted applicability. Nonetheless, it is easily seen that it applies to control of the change in the solution. The analysis in the paper [10] that first pointed out this property of step size selection algorithms is easily modified to accomodate control of the change in the solution, and some of the arguments seen in later papers are directly applicable. Adams methods in their various forms are often studied in terms of backward differences of the solution. Although attention is focussed on the higher differences that appear in the local error estimates, the analysis of [17] shows that instability is revealed in the first difference just as in the higher ones. Interestingly, the result [13] of most general applicability in terms of methods and equations shows that instability will be revealed by the first difference of the solution. A striking consequence of step size control is pointed out in [10]. When solving a stiff problem, the step size will be reduced as necessary to keep the computation stable, but when the step size is small enough for stability, an efficient algorithm will increase the step size because the accuracy is easy to achieve. As a consequence, the step size will, on average, be about as big as possible, meaning that the product of the step size and the dominant eigenvalue of the local Jacobian will be near the boundary of the stability region of the method. This implies that the cost of the integration will be determined by the stability properties of the formula, hence will depend weakly on the accuracy desired (and achieved).

To illustrate the stabilization provided by control of the change of the solution, the model stiff problem $y' = -100y$, $y(0) = 10^{-3}$ was solved on $[0, 50]$ using RK4. The solution decays so rapidly that the step size is almost wholly determined by stability. The results reported in Table 1 for the maximum magnitude, YMAX, of the numerical solution show that over a wide range of absolute error, AE, and relative error, RE, tolerances, the computation remained stable. It is seen that the cost as measured by the number of function evaluations, NFE, depended weakly on the tolerances. The product of the average step size, HAVG, and the eigenvalue -100 was near the stability boundary of the classical four stage, fourth order formula used, namely -2.78.

4. Difficult problems

The numerical examples of this selection explore the capabilities of control of the change of the solution for solving difficult problems. We have already seen that it responds in the desired way to stiffness. Problem 10 of the set of test problems assembled by Krogh [6] is often used to test step

Table 1
A stiff problem solved with RK4 and control of the change in the solution

AE	RE	NFE	YMAX	HAVG* (−100)
$1\cdot10^{-2}$	$1\cdot10^{-1}$	12,169	$4\cdot10^{-2}$	− 2.69
$1\cdot10^{-3}$	$1\cdot10^{-2}$	12,083	$3\cdot10^{-3}$	− 2.70
$1\cdot10^{-4}$	$1\cdot10^{-3}$	11,981	$3\cdot10^{-4}$	− 2.72
$1\cdot10^{-5}$	$1\cdot10^{-4}$	11,966	$4\cdot10^{-5}$	− 2.71
$1\cdot10^{-6}$	$1\cdot10^{-5}$	12,011	$3\cdot10^{-6}$	− 2.71
$1\cdot10^{-7}$	$1\cdot10^{-6}$	11,989	$3\cdot10^{-7}$	− 2.72
$1\cdot10^{-8}$	$1\cdot10^{-7}$	12,200	$3\cdot10^{-8}$	− 2.67
$1\cdot10^{-9}$	$1\cdot10^{-8}$	12,113	$4\cdot10^{-9}$	− 2.70

size selection algorithms because it requires a considerable variation in the step size:

$$y_1'' = 2y_2' + y_1 - \frac{\mu^*(y_1+\mu)}{r_1^3} - \frac{\mu(y_1-\mu^*)}{r_2^3}, \quad y_1(0) = 1.2,\ y_1'(0) = 0,$$

$$y_2'' = -2y_1' + y_2 - \frac{\mu^* y_2}{r_1^3} - \frac{\mu y_2}{r_2^3}, \quad y_2(0) = 0,$$

$$y_2'(0) = -1.0493575098031990726.$$

Here $r_1 = ((y_1+\mu)^2 + y_2^2)^{1/2}$, $r_2 = ((y_1-\mu^*)^2 + y_2^2)^{1/2}$, $\mu = 1/82.45$, and $\mu^* = 1-\mu$. This restricted three body problem has initial conditions that were determined in a high precision computation so as to result in a periodic solution of period $T = 6.19216933131963970674$. The periodicity of the solution allows us to assess the accuracy of a numerical integration by comparing the values computed at the end of the period to the initial values. An absolute error control is appropriate. To compare control of the change in the solution to control of the local error, we solved this problem for a range of tolerances with a well-known Runge–Kutta code, RKF45 [18], that implements a Fehlberg (4, 5) pair. Because this code does local extrapolation, we had to use the fifth order formula in the code based on control of the change in the solution so as to obtain comparable results. The maximum absolute error in the four solution components was measured at the end of the period. Fig. 1 shows the logarithm of the maximum absolute error at the end of the interval of integration plotted against the cost in function evaluations for the two codes. It is considerably more efficient to control the local error — we could hardly have expected a different outcome for a step size selection that does not depend on the formula — but it is possible to solve a difficult problem with acceptable efficiency.

To illustrate the solution of stiff problems, we solved van der Pol's equation with parameter 1000 on the interval $[0, 3000]$. The initial conditions $y(0) = 2$, $y'(0) = 0$ are close to a slowly varying portion of the limit cycle. It is suggested in [11] that this relaxation oscillation strains step size selection algorithms because on this interval the portions of the cycle where the solution changes slowly are connected by three internal boundary layers where the solution changes almost

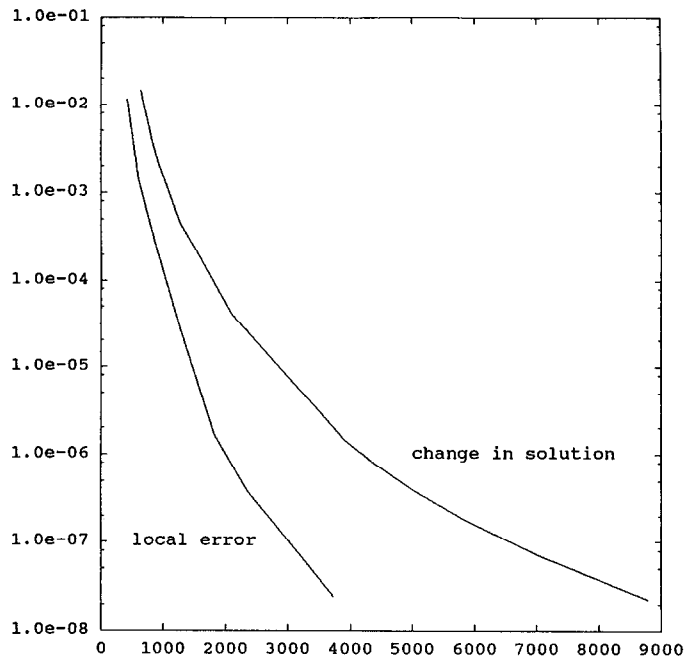

Fig. 1. Efficiency of error controls for a three body problem.

discontinuously. As a method appropriate for stiff problems we chose the semi-implicit, one-step method of Wolfbrandt [21]. The formula requires an approximate Jacobian at each step; for the sake of simplicity, we provided the analytical Jacobian. Because the method is of order two, only modest tolerances are appropriate. The problem was integrated without difficulty for a range of tolerances. For example, with AE = 0.05 and RE = 0.05 the code solved the problem at a cost of 1149 successful steps and 66 failed steps. The solution with AE = 0.01 and RE = 0.05 cost 1503 successful steps and 53 failed steps. When plotted, it was only slightly different from the other solution and was indistinguishable from the solution obtained with AE = 0.01 and RE = 0.01 at a cost of 5638 successful steps and 41 failed steps. Evidently the control is able to deal with very sharp changes in the solution.

We now take up a couple of difficult problems that test the robustness of step size selection schemes. The paper [19] compares the performance of the most effective codes of the time when applied to these problems. Not all were able to integrate the problems, and some did so very inefficiently. Problem 11 of Krogh's test set [6] is a problem with an integrable singularity, namely $y' = \frac{2}{3}x^{-1/3}$ for $x \neq 0$, $y' = 0$ for $x = 0$. It is pointed out in [19] that codes based on control of the local error with a criterion of error per unit step generally cannot integrate such an equation from $x = -1$ to $+1$ because when stepping across $x = 0$ with step size h, the local error is itself $\mathcal{O}(h)$. The change in the solution is also $\mathcal{O}(h)$, but it is compared to a given tolerance τ, rather than the $h\tau$ of EPUS, so is capable of taking a successful step. The difficulty here is that $y'(x)$ tends to $-\infty$ as x tends to 0, causing the step size to decrease as the integration approaches the origin. The code does manage to step past the origin, but an error is made that persists for the remainder of the

integration. In our experiments with RK4, the code did not fail catastrophically, but the results after passing the origin were not very accurate. This behavior is better than that of some of the codes studied in [19] and worse than others.

The solution of the problem

$$y'' = -y - \operatorname{sgn}(y) - 3\sin(2x), \quad 0 \leqslant x \leqslant 8\pi, \quad y(0) = 0, \quad y'(0) = 3,$$

where $\operatorname{sgn}(y) = +1$ if $y \geqslant 0$, $= -1$ if $y < 0$, has jump discontinuities at a spacing of $\frac{1}{2}\pi$. Although some of the codes compared in [19] failed or performed very poorly on this problem, our experimental code that controlled the change in the solution computed with RK4 provided acceptable solutions. This is not to say that it is as efficient as one of the better codes based on control of the local error, just that the step size control is capable of dealing with this kind of difficulty. With AE = 0.1 and RE = 0.1, the code solved the problem at a cost of 2853 successful steps and no failed steps. The maximum absolute error in $y(x)$ at any mesh point was $5 \cdot 10^{-2}$. Similarly, with AE = 0.05 and RE = 0.05, the integration costed 5741 successful steps and no failed steps, and the numerical solution was in error by no more than $2 \cdot 10^{-2}$. Comparing these computations to similar ones made with RKF45 revealed an interesting difference in the step sizes chosen. RKF45 must use a relatively small step size to pass a discontinuity because the local error does not behave as expected at such a point (the order of the formula is reduced). However, because the solution itself does not change rapidly, control of the change in the solution does not result in exceptionally small step sizes at these points. The control chooses relatively small step sizes near the extrema of the oscillation where the derivative of the solution has relatively sharp changes. RKF45 does not use exceptionally small step sizes there because the solution is smooth and the local error of the formula behaves as expected by the code.

References

[1] G.L. Baker and J.P. Gollub, *Chaotic Dynamics an Introduction* (Cambridge Univ. Press, Cambridge, 1991).

[2] T.S. Chua and P.M. Dew, The simulation of a gas transmission network using a variable-step integrator, Rept. 148, Dept. Computer Studies, Univ. of Leeds, United Kingdom, 1981.

[3] C.A.J. Fletcher, *Computational Galerkin Methods* (Springer, Berlin, 1984).

[4] T.E. Hull, W.H. Enright, B.M. Fellen and A.E. Sedgwich, Comparing numerical methods for ordinary differential equations, *SIAM J. Numer. Anal.* **9** (1972) 603–637.

[5] P. Kaps and P. Rentrop, Generalized Runge–Kutta methods for order four with stepsize control for stiff ordinary differential equations, *Numer. Math.* **38** (1979) 55–68.

[6] F.T. Krogh, On testing a subroutine for the numerical integration of ordinary differential equations, *J. ACM* **20** (1973) 545–562.

[7] A. Ralston, Runge–Kutta methods with minimum error bounds, *Math. Comput.* **16** (1962) 431–437.

[8] R.E. Scraton, Some L-stable methods for stiff differential equations, *Internat. J. Comput. Math. Sec. B* **9** (1981) 81–87.

[9] M.J. Shah, *Engineering Simulation Using Small Scientific Computers* (Prentice-Hall, Englewood Cliffs, NJ, 1976).

[10] L.F. Shampine, Stiffness and non-stiff differential equation solvers, in: L. Collatz et al., Eds., *Numerische Behandlung von Differentialgleichungen*, ISNM **27** (Birkhauser, Basel, 1975) 287–301.

[11] L.F. Shampine, Evaluation of a test set for stiff ODE solvers, *ACM Trans. Math. Software* **7** (1981) 409–420.

[12] L.F. Shampine, The step sizes used by one-step codes for ODEs, *Appl. Numer. Math.* **1** (1985) 95–106.

[13] L.F. Shampine, Practical improvements to SIMPR codes for stiff ODEs, *Appl. Math. Comput.* **31** (1989) 328–345.

[14] L.F. Shampine, Tolerance proportionality in ODE codes, in: A. Bellen et al., Eds., *Numerical Methods in ODEs*, Lecture Notes in Math. **1386** (Springer, Berlin, 1989) 118–136.
[15] L.F. Shampine and L.S. Baca, Error estimators for stiff differential equations, *J. Comput. Appl. Math.* **11** (1984) 197–207.
[16] L.F. Shampine and M.K. Gordon, Some numerical experiments with DIFSUB, *SIGNUM Newsletter* **1** (1975) 24–26.
[17] L.F. Shampine and M.K. Gordon, *Computer Solution of Ordinary Differential Equations: the Initial Value Problem* (Freeman, San Francisco, 1975).
[18] L.F. Shampine and H.A. Watts, Practical solution of ordinary differential equations by Runge–Kutta methods, Rept. SAND 76-0585, Sandia National Laboratories, Albuquerque, NM, 1976.
[19] L.F. Shampine, H.A. Watts and S.M. Davenport, Solving nonstiff ordinary differential equations — the state of the art, *SIAM Rev.* **18** (1976) 376–411.
[20] F.H. Speckhard and W.L. Green, *A Guide to Using CSMP — the Continuous System Modelling Program* (Prentice-Hall, Englewood Cliffs, NJ, 1976).
[21] T. Steighaug and A. Wolfbrandt, An attempt to avoid exact Jacobian and nonlinear equations in the numerical solution of stiff differential equations, *Math. Comput.* **33** (1979) 521–534.
[22] H.A. Watts, Starting step size for an ODE solver, *J. Comput. Appl. Math.* **9** (1983) 177–191.
[23] H. Zedan, A preliminary assessment of the local error estimates of some Rosenbrock-type methods, Rept. CS-83-06, School of Math., Comp. Sci., Univ. of Bristol, United Kingdom, 1983.
[24] J.A. Zonneveld, *Automatic Numerical Integration* (Mathematisch Centrum, Amsterdam, 1964).