CONVERGENCE PROOFS FOR NUMERICAL IVP SOFTWARE

HARBIR LAMBA* AND ANDREW M. STUART*

Abstract. The study of the running times of algorithms in computer science can be broken down into two broad types: worst-case and average-case analyses. For many problems this distinction is very important as the orders of magnitude (in terms of some measure of the problem size) of the running times may differ significantly in each case, providing useful information about the merits of the algorithm. Historically average-case analyses were first done with respect to a measure on the input data; to counter the argument that it is often difficult to find a natural measure on the data, randomised algorithms were then developed.

In this paper similar questions are studied for adaptive software used to integrate initial value problems for ODEs. In the worst case these algorithms may fail completely giving $\mathcal{O}(1)$ errors. We consider the probability of failure for generic vector fields with random initial data chosen from a ball and perform average-case and worst-case analyses. We then perform a different average-case analysis where, having fixed the initial data, it is the algorithm that is chosen at random from some suitable class. This last analysis suggests a modified *deterministic* algorithm which cannot fail for generic vector fields.

Key words. Error control, Adaptivity, Random algorithms, Convergence, Tolerance proportionality.

AMS(MOS) subject classifications. 34C35, 65L07, 65L20, 65L50.

1. Introduction.

Consider the initial value problem

$$\frac{du}{dt} = f(u), \quad u(0) = U \text{ where } f : \mathbb{R}^m \to \mathbb{R}^m \tag{1.1}$$

which is to be numerically integrated over some finite time-interval $[0, T]$. The simplest algorithms for solving this problem advance the solution with a fixed timestep Δt using, say, a Runge-Kutta method of order p. The classical numerical analysis theory then ensures that as $\Delta t \to 0$ the global error is bounded by $C(B, T)\Delta t^p$ for all initial data U in some compact set B. This convergence to the exact solution at a well-defined rate as some user-defined quantity, in this case Δt, tends to zero should be a fundamental requirement of any algorithm to solve (1.1).

However, most software used to solve (1.1) adaptively changes the timestep as the integration progresses, trying simultaneously to maximise the timesteps (for efficiency) and ensure that the error committed on each step is always less than some user-defined tolerance τ (for accuracy). Numerous comparisons have shown that such adaptive methods almost always outperform fixed timestepping methods, reaching a comparable level of accuracy with significantly less computational work. However, although these algorithms have been used for many years they are difficult to analyse because they can fail for certain initial data, because of their discontinuous nature and also because there are also many possible differences in implementation. Recent rigorous convergence results [13, 9] show that convergence to the exact solution as $\tau \to 0$ may not occur, or occur at a reduced rate, if the exact solution passes through certain, typically small, neighbourhoods of phase space. We shall regard such events as a 'failure' of the algorithm and it is our intention to quantify these cases and determine an 'average' overall behaviour of the algorithms.

*Scientific Computing and Computational Mathematics Program, Division of Mechanics and Computation, Durand 262, Stanford University, CA 94305-4040 USA. Supported by NSF Grant DMS-95-04879.

In §2 we describe a class of typical adaptive timestepping algorithms (based upon the routine ode23 from MATLAB Version 4.2) and state finite-time convergence results. We also define exactly what we mean by 'failure' of an algorithm. Then in §3 we briefly outline random analyses from other fields of computer science and numerical analysis in order to motivate our study. In §4 we present our probabilistic analysis for the adaptive algorithms in §2. We show that for a fixed algorithm and random *initial conditions* the probability of failure is small in dimensions $m > 1$ and the algorithm performs 'well' on average. This is not the case however for $m = 1$. We then show that if the initial condition is fixed and an *algorithm* is chosen at random then the same situation occurs. Finally in §5 we outline a proof demonstrating that alternating between two different algorithms can eliminate the possibility of failure for generic vector fields.

2. Convergence results for adaptive timestepping algorithms.

2.1. The algorithm. An adaptive algorithm for solving (1.1) generates a sequence $(U_n, \Delta t_n)$ where the Δt_n are the timesteps and (hopefully) $U_n \approx u(t_n)$ where $t_n = \sum_{i=0}^{n-1} \Delta t_i$. The algorithms that we shall analyse consist of two basic components not present in fixed timestep methods: a *local error estimate* and a *timestep selection strategy*.

The local error estimate is defined as the difference between two explicit Runge-Kutta approximations, $S^{(1)}$ and $S^{(2)}$, of orders p and $p-1$ respectively. For computational efficiency these are usually generated using the same intermediate stages and together form an embedded Runge-Kutta pair. Thus they can be written as

$$\eta_i = u + \Delta t \sum_{j=1}^{i-1} a_{ij} f(\eta_j), \quad i = 1, \dots, s \tag{2.1}$$

$$S^{(1)}(u, \Delta t) = u + \Delta t \sum_{j=1}^{s} b_j^{(1)} f(\eta_j), \tag{2.2}$$

$$S^{(2)}(u, \Delta t) = u + \Delta t \sum_{j=1}^{s} b_j^{(2)} f(\eta_j) \tag{2.3}$$

where s is the number of stages, A is a lower-triangular matrix, with entries a_{ij}, and $\mathbf{b}^{(i)},\ i = 1, 2$ are two different sets of weights, one for each method, with entries $b_j^{(i)}$. From now on we set $p = s = 3$ giving

$$S^{(1)}(u, \Delta t) - S(u, \Delta t) = \mathcal{O}(\Delta t^4)$$

and

$$S^{(2)}(u, \Delta t) - S(u, \Delta t) = \mathcal{O}(\Delta t^3)$$

where $S(u, t)$ is the semigroup defined by the vector field f. We can now define the local error estimate as

$$E(u, \Delta t) := \|S^{(1)}(u, \Delta t) - S^{(2)}(u, \Delta t)\| = \mathcal{O}(\Delta t^3). \tag{2.4}$$

There is in fact a 2-parameter family of such embedded pairs (that includes the pair used in MATLAB ode23) but for simplicity we shall only choose from the following 1-parameter family, written in Butcher notation as:

$$\begin{array}{c|ccc} 0 & 0 & 0 & \\ c & 0 & 0 & A \\ \frac{2}{3}(1-\frac{1}{3c}) & \frac{2}{9c} & 0 & \\ \hline 1-\frac{1}{2c} & \frac{1}{2c} & 0 & \mathbf{b}^{(2)} \\ \frac{1}{4} & 0 & \frac{3}{4} & \mathbf{b}^{(1)} \end{array}$$

where the parameter $c \in [\frac{1}{3}, \frac{2}{3}]$.

The timestep selection mechanism operates as follows. Set $U_0 = U$ and let $\Delta t_{\rm init}$ be an initial estimate for the first timestep. The other input is the tolerance τ. At each step n the algorithm generates a monotone decreasing (in k) sequence of possible timesteps $\Delta t_n^{(k)}$ stopping at $k = l$ only when the local error estimate $E(U_n, \Delta t_n^{(l)})$ is sufficiently small compared to τ. This $\Delta t_n^{(l)}$ is then taken to be the timestep Δt_n and the solution U_n is advanced with this timestep to give U_{n+1}. The process repeats until $t_n = T$. The precise mechanism that we shall analyse is based upon the MATLAB Version 4.2 ode23 routine:

$$U_{n+1} = S^{(1)}(U_n, \Delta t_n), \quad U_0 = U, \tag{2.5}$$

where $\Delta t_n^{(0)} = \Delta t_{\rm init}$ if $n = 0$ or

$$\min\left(D, \theta\left(\frac{\tau\max(1, \|U_{n-1}\|_\infty)}{E(U_{n-1}, \Delta t_{n-1})}\right)^{\frac{1}{3}} \Delta t_{n-1}, T - t_n\right) \quad \text{if } n > 0, \tag{2.6}$$

$$\Delta t_n^{(k)} = \min\left(D, \theta\left(\frac{\tau\max(1, \|U_n\|_\infty)}{E(U_n, \Delta t_n^{(k-1)})}\right)^{\frac{1}{3}} \Delta t_n^{(k-1)}, T - t_n\right) \quad \text{for } k \geq 1, \tag{2.7}$$

$$\text{and } \Delta t_n = \Delta t_n^{(l)} \text{ where } l = \min_{k\geq 0}\{k : E(U_n, \Delta t_n^{(k)}) \leq \tau\max(1, \|U_n\|_\infty)\}. \tag{2.8}$$

The parameter D is the maximum allowable timestep and is *not* reduced as $\tau \to 0$, while $\theta \in (0, 1)$ is a safety factor that aims to reduce the number of (computationally wasteful) rejected steps. We shall not consider here how these parameters may be chosen in practice although the values used in MATLAB Version 4.2 ode23 are $\theta = 0.9$ and $D = T/16$. The exponent $\frac{1}{3}$ in (2.6) and (2.7) appears since $E(u, \Delta t) = \mathcal{O}(\Delta t^3)$ — this ensures that, at least for small timesteps, the estimates in the sequence $\Delta t_n^{(k)}$ are of the correct order. Note also that the local error estimate is not compared against τ directly but against $\tau\max(1, \|U_n\|_\infty)$ giving an absolute error for small U_n and a relative error for large U_n. This feature is very common in software for ODEs and causes no difficulty in the analysis. Finally note that the higher-order method $S^{(1)}$ is being used to update the solution (i.e. the algorithm is operating in extrapolation mode).

2.2. Convergence results. A convergence result for the above class of algorithms was proved in [9] and we now give a brief outline of the proof and statement of the result (as a similar proof for a modified algorithm will be used in §5).

The local error estimate can be expanded as

$$E(u, \Delta t) = \Delta t^3 \|f(u)\| \left\| \left(b_1(u) + \Delta t\|f(u)\|\tilde{b}_2(u, \Delta t)\right) \right\| \tag{2.9}$$

where $b_1(u)$ and $\tilde{b}_2(u,\Delta t)$ are bounded on bounded sets and depend upon both the vector field f and the Runge-Kutta coefficients. Also, since the algorithm is operating in extrapolation mode, the truncation error (i.e. the error made on one step) can be written as

$$T(u,\Delta t) := S^{(1)}(u,\Delta t) - S(u,\Delta t) = f(u)\mathcal{O}(\Delta t^4). \tag{2.10}$$

The crucial step is an induction argument using (2.9) proving that, for sufficiently small τ and sufficiently small Δt_{init}, the accepted timesteps are bounded from above by

$$\Delta t_n^3 \le \frac{K_1 \tau}{\|f(U_n)\|\epsilon}, \quad \forall n : 0 \le t_n \le T \tag{2.11}$$

where $K_1 = K_1(T)$ is some constant and

$$\epsilon = \inf_{t\in[0,T]} \|b_1(S(U,t))\|. \tag{2.12}$$

Thus, the truncation error at each step satisfies

$$\|T(U_n,\Delta t_n)\| \le \frac{K_2 \tau \Delta t_n}{\epsilon}. \tag{2.13}$$

for $K_2 = K_2(T)$.

A weak assumption on the set where $b_1(u)$ vanishes is also needed, namely that this set does not intersect the set of equilibria of f.

ASSUMPTION 1. *There exist* $\overline{\delta}, \overline{\epsilon} > 0$ *such that* $\forall u \in J,\ \|f(u)\| < \overline{\delta} \Rightarrow \|b_1(u)\| > \overline{\epsilon}$.

Finally, using Assumption 1 and a straightforward Gronwall argument, the global error may be shown to satisfy

$$E_n = \|u(t_n) - U_n\| \le K\frac{\tau}{\epsilon} \quad \forall n : t_n \le T$$

for some $K = K(T)$.

We therefore have the following convergence theorem, paraphrased from [9].

THEOREM 2.1. *Consider the numerical approximation of (1.1) over the time interval* $[0,T]$ *generated by the algorithm* (2.1)—(2.8) *for any choice of* c. *Let Assumption 1 hold and assume in addition that* $\|b_1(S(U,t))\| \ge \epsilon$ *for* $t \in [0,T]$ *and that* Δt_{init} *is sufficiently small. Then there exists a constant* $K = K(T)$ *such that for all sufficiently small* τ,

$$E_n = \|u(t_n) - U_n\| \le K\frac{\tau}{\epsilon} \quad \forall n : t_n \le T. \tag{2.14}$$

The theorem states that if the exact solution stays away from the set of points where the leading term $b_1(u)$ in the expansion of the local error estimate, scaled by $\|f(u)\|$, vanishes then the global error decreases like some power of τ (in this case linearly). This highly desirable property is known as *Tolerance Proportionality* (TP) [2, 11, 12] and will be our criterion for 'success'. If $b_1(u)$ does vanish along the exact trajectory then the inductive upper bound on the timestep sequence breaks down since the local error estimate is now $\mathcal{O}(\Delta t^4)$ but the actual local error is still $\mathcal{O}(\Delta t^3)$. This allows the timesteps to increase in the neighbourhood of these points. When this occurs then global convergence is still often observed but at a reduced rate, although the loss of control of the timestep sequence means that convergence can no longer be guaranteed.

Because $b_1(u)$ is a function from $\mathbb{R}^m$ to $\mathbb{R}^m$, for generic vector fields f it only vanishes at isolated points. For generic vector fields in $\mathbb{R}^m$, $m > 1$, and random initial data (using any measure that is absolutely continuous with respect to Lebesgue measure) it is therefore a measure-zero event for $b_1(u)$ to vanish along any finite time segment of the exact trajectory; note, however, that this is not the case for $m = 1$. The probability that the exact trajectory enters some neighbourhood where $b_1(u) < \epsilon$ will form an important part of our analysis in §4.

The induction argument used to prove (2.13) relies crucially upon the existence of the function $\tilde{b}_2(u,t)$, i.e. that the higher-order terms of the local error expansion in (2.9) are $\mathcal{O}(\|f(u)\|^2)$. This is automatically true for all Runge-Kutta pairs of order p and $p-1$ that use precisely p stages and hence holds for the 1-parameter family of methods in §2. If such a pair is not used then the analysis of [13] can be applied and the convergence statement (2.14) becomes

$$E_n = \|u(t_n) - U_n\| \leq K\frac{\tau}{\delta\epsilon} \quad \forall n : t_n \leq T.$$

where $\delta = \inf_{t\in[0,T]} \|f(S(U,t))\|$. Hence loss of TP (or even convergence) can now also occur in the neighbourhood of equilibria (a numerical example is presented in [9] and a different analysis near $f(u) = 0$ can be found in [6]). In fact these convergence results hold for a very wide class of algorithms under minimal assumptions on the timestep selection algorithm [8]. Thus the convergence properties of an algorithm are solely determined by the Runge-Kutta pair used and the resulting form of the local error estimate expansion.

3. Probabilistic analyses of algorithms. The results from §2.2 show that TP can be lost close to points where $b_1(u)$ vanishes and this can be regarded as a worst-case analysis. However, for most IVPs, adaptive algorithms appear to work very well, being far more efficient computationally than fixed timestepping algorithms. Therefore some kind of 'average-case' analysis is appropriate. Before re-examining the results of §2 from a probabilistic point of view we shall give a very brief review of some previous probabilistic analyses for other computational algorithms.

The *Quicksort* algorithm of Hoare [7] for sorting a string of n numbers is a good example of the difference between average-case and worst-case results. In the worst case the 'work' required is $\mathcal{O}(n^2)$ while for the 'average' case the expected amount is $\mathbb{E}(Work) = \mathcal{O}(n \ln n)$. This average-case analysis assumes that all permutations of the input string ordering are equally likely; a valid objection is that this is almost certainly not the case in many applications. However, by fixing the input sequence and instead introducing randomness into the algorithm itself, the same average-case results hold for any input, overcoming the previous objection. A more detailed discussion can be found in [3].

A second example is Smale's analysis of a (modified) Newton method [10, 1] for a polynomial of degree d; in the worst case this algorithm fails. However in the average case, with probability $1-\mu$, the amount of work to obtain an approximate zero (precisely defined in [10]) is polynomial in d and μ^{-1}. The analysis is for all monic polynomials of degree d using Lebesgue measure on the unit ball for the remaining co-efficients.

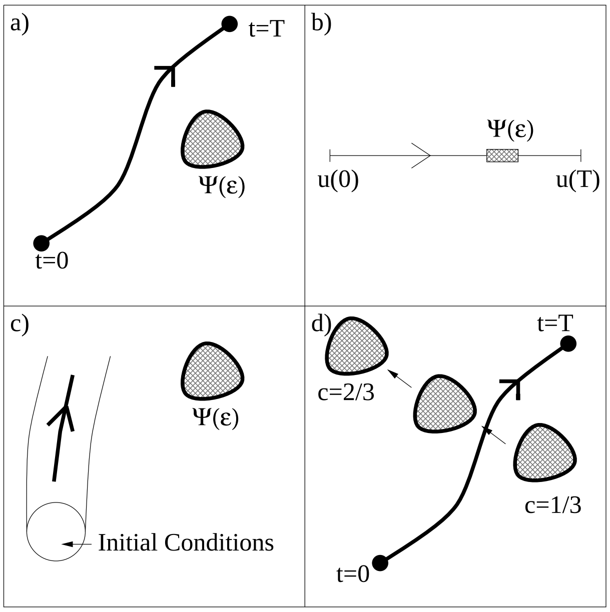

FIG. 1. *Figure (a) shows that typical trajectories avoid bad points in dimension $m > 1$ while Figure (b) shows that this is not the case for $m = 1$. Figures (c) and (d) show typical situations for random initial conditions and random algorithm parameter c respectively in $m > 1$ dimensions.*

Finally we consider an example from linear algebra which is Demmel's 1987 analysis [4] (then Edelman [5]) of the condition number $\kappa(A)$ of a random $n \times n$ matrix A. In the worst case $\kappa(A) = \infty$ while, on average, $\text{Prob}(\kappa(A) \geq x) \approx \frac{n^3}{x^2}$. This result assumes Lebesgue measure on all matrices with Frobenius norm equal to 1.

From the above analyses it can be seen that an average-case analysis often yields more information about an algorithm than a worst-case study — for example Quicksort would not be used if only the worst-case were relevant since there are other algorithms that have only $\mathcal{O}(n \ln n)$ worst-case behaviour. Also, the objection to placing a measure on the set of input data, usually chosen just to make the analysis simpler, can be overcome by instead randomising the algorithm itself and applying a probability measure to the set of algorithms instead.

4. Probabilistic analysis and examples. Theorem 2.1 shows that convergence is non-uniform with respect to initial conditions for trajectories that pass through neighbourhoods of points in $\mathbb{R}^m$ where $b_1(u)$ vanishes and it was noted in §2.2 that for generic vector fields this set will consist of isolated points. There are two cases to consider, $m = 1$ and $m \geq 2$.

Let us define

$$\Psi(\epsilon) := \{u \in \mathbb{R}^m : \|b_1(u)\| < \epsilon\}.$$

Figure 1(a) shows the typical case for $m \geq 2$. The shaded region denotes $\Psi(\epsilon)$ for some

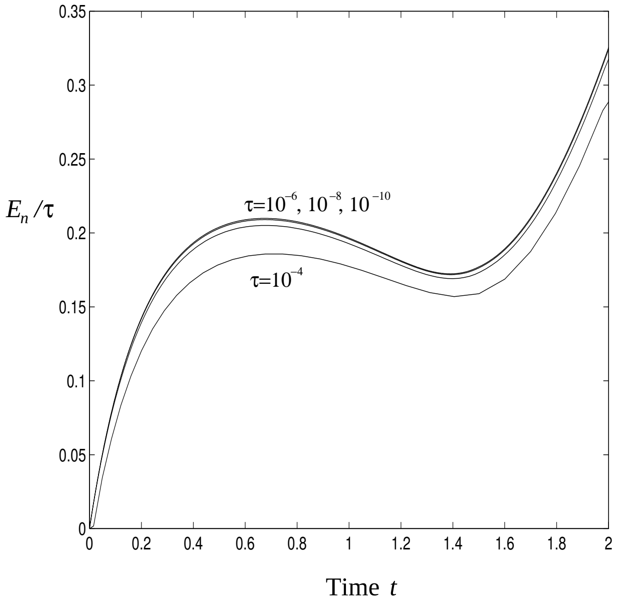

FIG. 2. *The plotted curves are seen to converge as $\tau \to 0$, implying Tolerance Proportionality.*

small ϵ and most trajectories will not enter it, implying TP with error constant at most K/ϵ. Figure 2 shows a numerical example of this situation. The initial value problem

$$\frac{du}{dt} = u(1 - u - v), \quad u(0) = 2,$$
$$\frac{dv}{dt} = v, \quad v(0) = 0.1$$

was solved over the time interval $[0, 2]$ using the algorithm (2.1)—(2.8) with $c = \frac{1}{2}$. The quantity $\|U_n - u(t_n)\|/\tau$ is plotted against t_n for $\tau = 10^{-4}, 10^{-6}, 10^{-8}$ and 10^{-10}. If TP is occurring then the curves should converge to a limit as $\tau \to 0$ which is indeed seen to be the case.

For $m = 1$ however, the situation is much worse (see Figure 1(b)). Whole neighbourhoods of initial conditions will pass through $\Psi(0)$ resulting in TP failure. Figure 3 shows the convergence behaviour for the IVP

$$\frac{du}{dt} = u - u^2, \quad u(0) = 0.5$$

over the time interval $[0, 5]$ again using $c = \frac{1}{2}$. TP is lost at $t \approx 0.8$ when the solution passes through a value of u for which $b_1(u) = 0$ and the results are similar for all nearby initial conditions. For other numerical examples and a more detailed analysis, see [2].

We now start our average-case analysis for dimensions $m \geq 2$ by considering trajectories chosen at random from a ball in $\mathbb{R}^m$ with $m \geq 2$ (see Figure 1(c)). It is clearly important to

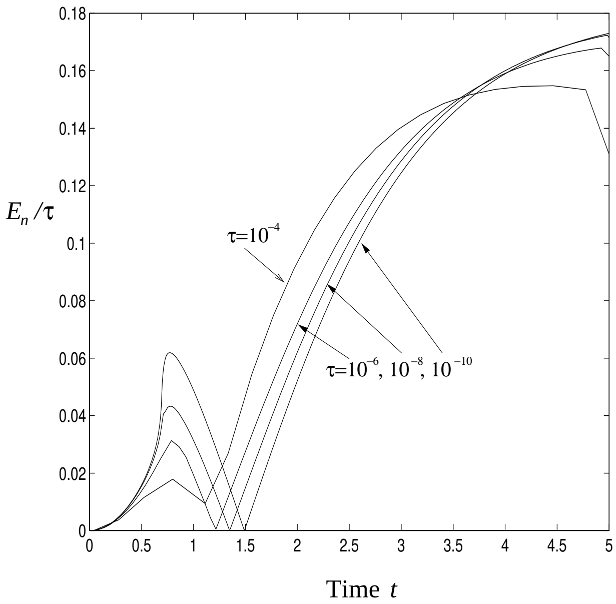

FIG. 3. *Tolerance Proportionality does not occur since the exact solution passes through a point where $b_1(u) = 0$ at $t \approx 0.8$.*

estimate the probability that a trajectory remains outside $\Psi(\epsilon)$ since Theorem 2.1 states that the global error constant is then bounded by K/ϵ for some constant $K(T)$. Mathematically this may be done by estimating the probability that the image of $\Psi(\epsilon)$ under $S(\bullet, -t),\ t \in [0, T]$ intersects the ball of initial conditions. To do this it is simplest to assume that the set $\Psi(\epsilon)$ can be inscribed in finitely many disjoint balls of $\mathcal{O}(\epsilon)$ which are all disjoint from the set of equilibria. This holds for generic $f(u)$ within the class of sufficiently smooth functions [13] but we now state it as an assumption.

ASSUMPTION 2. *There is a constant $\epsilon_c > 0$ such that, for each $\epsilon \in [0, \epsilon_c)$, the set $\Psi(\epsilon)$ is the disjoint union of a countable set of neighbourhoods $\{\Psi_i\}_{i=1}^{M}$ with $M \leq \infty$ each containing a point $z_i \in \mathbb{R}^m$ at which $b_1(z_i) = 0$. Thus, for each $\epsilon \in [0, \epsilon_c)$,*

$$\Psi(\epsilon) = \bigcup_{i=1}^{M} \Psi_i, \quad \Psi_i \bigcap \Psi_j = \emptyset \quad \forall i \neq j.$$

Furthermore, for any finite integer M_0 there exists a constant C_1 such that, for all $\epsilon \in [0, \epsilon_c)$

$$\Psi_i \subseteq B(z_i, C_1\epsilon),\ i = 1, \ldots, M_0.$$

Based upon the analysis in [13, 9] we can state the following result.

THEOREM 4.1. *Let Assumptions 1 and 2 hold and suppose that U is chosen randomly with respect to Lebesgue measure from a ball in $\mathbb{R}^m$ and $m > 1$. Then, for sufficiently large*

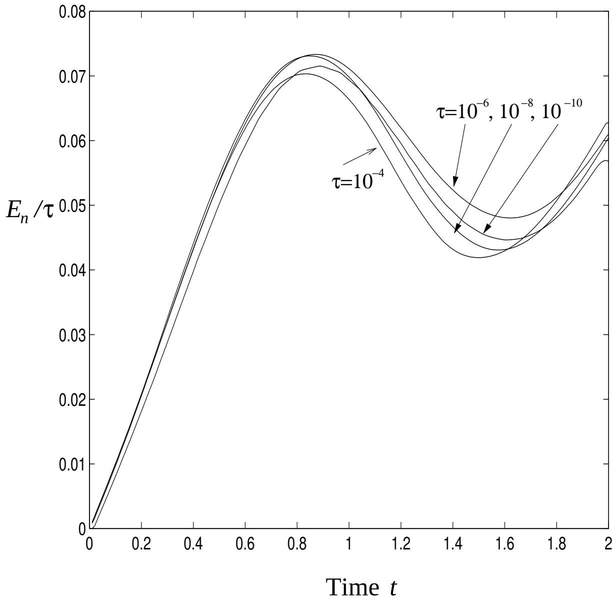

FIG. 4. *The errors averaged over 100 runs with random initial conditions are plotted and Tolerance Proportionality appears to hold.*

R, there exists $C(T) > 0$ such that

$$\mathrm{Prob}(\inf_{t\in[0,T]} \|b_1(S(U,t))\| \geq \frac{1}{R}) \geq 1 - CR^{1-m}.$$

Combining this with Theorem 2.1 gives a probabilistic convergence result — essentially the global error is $\mathcal{O}(R\tau)$ with probability greater than $1 - \mathcal{O}(R^{1-m})$. Note that the probability decreases as the dimension m of the problem increases. This average-case situation is (crudely) reflected in Figure 4 where for each value of τ the IVP

$$\frac{du}{dt} = u + v - u(u^2 + v^2)$$
$$\frac{dv}{dt} = -u + v - v(u^2 + v^2)$$

is numerically integrated (once again using $c = \frac{1}{2}$) over the time interval $[0, 2]$ for 100 different initial conditions chosen randomly from the square $0.3 \leq u, v \leq 0.7$. For each run a piecewise-linear numerical solution $U(t)$ is generated by interpolation of the output from the algorithm and the quantity $\|U(t) - u(t)\|/\tau$, averaged over all the runs, is plotted against t. Tolerance Proportionality, in some average-case sense, clearly occurs.

However, the objection to this average-case approach is the same as for the Quicksort analysis — initial data are not usually chosen at random from some mathematically convenient measure. Thus we instead consider the average-behaviour for a *fixed* initial condition if the algorithm is chosen randomly from some class. The class we shall choose is given by picking $c \in [\frac{1}{3}, \frac{2}{3}]$ at random. As is shown in Figure 1(d), for generic vector fields, the points z_i in $\Psi(0)$ will move transversely to the vector field as c varies. For simplicity we shall also require that the quantities $dz_i(c)/dc$ exist, leading to the following assumption.

ASSUMPTION 3. *Assumption 2 holds for each $c \in [\frac{1}{3}, \frac{2}{3}]$. Furthermore, for all $c \in [\frac{1}{3}, \frac{2}{3}]$ each $\frac{dz_i(c)}{dc}$ exists and*

$$\frac{\frac{dz_i(c)}{dc}}{\|\frac{dz_i(c)}{dc}\|} \neq \frac{f(z_i(c))}{\|f(z_i(c))\|}.$$

A similar analysis to that for the case of random initial conditions gives the following theorem.

THEOREM 4.2. *Let Assumptions 1 and 3 hold and suppose also that c is chosen randomly from the uniform distribution on $[\frac{1}{3}, \frac{2}{3}]$ with the corresponding algorithm used to integrate* (1.1) *for $m \geq 2$. Then, for sufficiently large R, there exists $C(T) > 0$ such that*

$$\text{Prob}(\inf_{t\in[0,T]} \|b_1(S(U,t))\| \geq \frac{1}{R}) \geq 1 - CR^{-1}.$$

Together with Theorem 2.1 this gives a second probabilistic convergence result. As a numerical example, the same IVP used in Figure 4 with fixed initial condition $(U, V) = (0.4, 0.3)$ was integrated 100 times with c chosen uniformly from $[\frac{1}{3}, \frac{2}{3}]$. The same averaged quantity $\|U(t) - u(t)\|/\tau$ is plotted against time in Figure 5 and again averaged TP occurs.

5. A modified deterministic algorithm. The results of the last section have shown that adaptive algorithms may fail, in the sense of not obeying TP, but that the probability is small for problems of dimension $m > 1$. However, the random algorithm analysis suggests a *deterministic* modification that guarantees TP for *all* trajectories for a generic class of vector fields for problems of all dimensions, including $m = 1$.

The basic idea is to choose two different embedded Runge-Kutta pairs (that is, to choose two different values of c) and switch between them at each timestep. For typical vector fields, the sets $\Psi(\epsilon)$, $\epsilon \ll 1$ for each pair will not intersect each other and as we prove below, this leads to significantly stronger convergence results.

The proof is essentially the same as was outlined in §2.2: an induction provides an upper bound on the timestep sequence and then a Gronwall argument gives the desired convergence result. We shall concentrate on proving the upper bound on the timesteps since this is the key step and simply refer the reader to the proofs of the other necessary Lemmas. For clarity, we shall only prove the convergence result for modifications of the algorithms defined in §2; but the proof can easily be extended to modifications of other existing algorithms.

We begin by precisely stating the modified 'switching' algorithm We define the first embedded Runge-Kutta pair, corresponding to $c = c_1$, as $S_1^{(1)}$ and $S_1^{(2)}$ and the second pair, corresponding to $c = c_2$, as $S_2^{(1)}$ and $S_2^{(2)}$. The methods $S_i^{(1)}$, $i = 1, 2$ are the ones used to advance the solution. For definiteness the first pair of methods will be used on odd numbered timesteps and the second pair on even steps. We shall define $i_n = 1$ if n is odd and $i_n = 2$ if n is even.

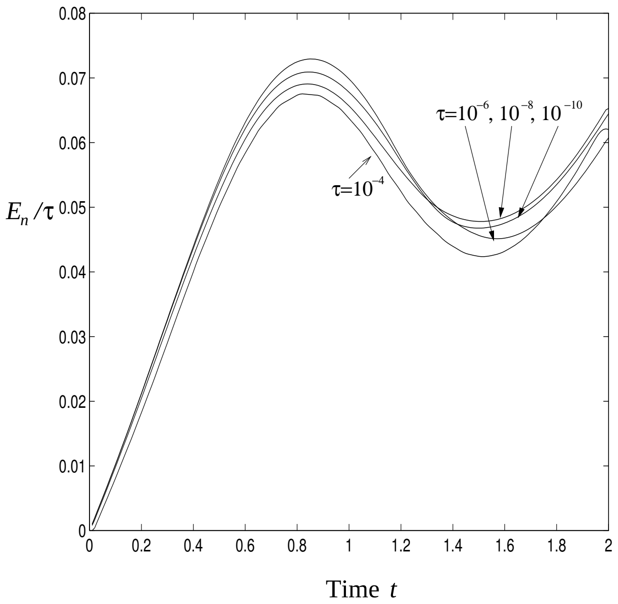

FIG. 5. *The errors for a fixed initial condition but with random c are averaged over 100 runs and Tolerance Proportionality appears to hold.*

We write the two sets of local error estimates as $E_1(u, \Delta t)$ and $E_2(u, \Delta t)$ and so

$$E_{i_n}(u, \Delta t_n) = \|S_{i_n}^{(1)}(u, \Delta t_n) - S_{i_n}^{(2)}(u, \Delta t_n)\|. \tag{5.1}$$

From [9] the local error estimates $E_1(u, \Delta t)$ and $E_2(u, \Delta t)$ can be expanded as

$$E_i(u, \Delta t) = \Delta t^3 \|f(u)\| \ \left\| \left\{ b_{i,1}(u) + \Delta t \|f(u)\| \tilde{b}_{i,2}(u, \Delta t) \right\} \right\|, \quad i = 1, 2. \tag{5.2}$$

Rather than use the mixed relative/absolute error function $\tau \max(1, \|U_n\|_\infty)$ from §2 we generalise (as in [9]) and instead test the local error estimate against a more general function $\sigma(\tau, U_n)$ where, for every compact set $J \subset \mathbb{R}^m$, there exist constants $C_1(J)$ and $C_2(J)$ such that

$$0 < C_1(J) \leq \sigma(\tau, u)/\tau \leq C_2(J) < \infty \ \ \forall \tau > 0 \ \text{ and } \ \forall u \in J. \tag{5.3}$$

Integration timesteps Δt_n are now chosen by the following algorithm:

$$U_{n+1} = S_{i_n}^{(1)}(U_n, \Delta t_n), \quad U_0 = U, \tag{5.4}$$

where $\Delta t_n^{(0)} = \Delta t_{\mathrm{init}}$ if $n = 0$ or

$$\min\left(D, \alpha\Delta t_{n-1}, \theta\left(\frac{\sigma(\tau, U_{n-1})}{E_{i_{n-1}}(U_{n-1}, \Delta t_{n-1})}\right)^{\frac{1}{3}} \Delta t_{n-1}, T - t_n\right) \text{ if } n > 0, \tag{5.5}$$

$$\Delta t_n^{(k)} = \min\left(D, \alpha\Delta t_{n-1}, \theta\left(\frac{\sigma(\tau, U_n)}{E_{i_n}(U_n, \Delta t_n^{(k-1)})}\right)^{\frac{1}{3}} \Delta t_n^{(k-1)}, T - t_n\right) \text{ for } k \geq 1, \tag{5.6}$$

$$\text{and } \Delta t_n = \Delta t_n^{(l)} \text{ where } l = \min_{k \geq 0}\{k : E_{i_n}(U_n, \Delta t_n^{(k)}) \leq \sigma(\tau, U_n)\}. \tag{5.7}$$

Note that we have now introduced a maximum stepsize ratio $\alpha > 1$ which ensures that $\Delta t_{n+1}/\Delta t_n \leq \alpha$ and is a common feature of many codes. While its introduction into the algorithm (2.1)—(2.8) does not lead to improved convergence properties, it will be essential to the improved convergence results that we shall prove for the switching algorithm.

We assume that the exact solution lies inside some compact set $J \subset \mathbb{R}^m$. We also make the following assumption which holds for generic $f(u)$.

Assumption 4. *The zeros of the continuous functions $b_{1,1}(u)$ and $b_{2,1}(u)$ in the compact set J are distinct and*

$$\epsilon^* = \frac{1}{2}\min_{u \in J}(\max(\|b_{1,1}(u)\|, \|b_{2,1}(u)\|)) > 0.$$

Note that ϵ^*, which will appear in our global error estimate, depends only upon f, c_1 and c_2 and not upon the initial conditions.

We define the constants

$$C_{i,4} = \sup_{(u,t) \in J \times [0,D]} \|\tilde{b}_{i,2}(u,t)\| < \infty, \quad i = 1, 2$$

for $i = 1, 2$. We then set $C_4 = \max(C_{1,4}, C_{2,4})$. The finiteness of the $C_{i,4}$ uses the fact that we are only considering Runge-Kutta pairs of order 2 and 3 with precisely 3 stages. The proof can be found in [9].

We also define the sets

$$\Psi_i(\epsilon^*) := \{u \in \mathbb{R}^m : \|b_{i,1}(u)\| < \epsilon^*\}, \quad i = 1, 2$$

and

$$J_{i,\epsilon^*} = J \setminus \Psi_i(\epsilon^*), \ i = 1, 2.$$

From Assumption 4 it follows that

$$\gamma := \frac{1}{2} \inf_{u \in \Psi_1(\epsilon^*),\, v \in \Psi_2(\epsilon^*)} (\|u - v\|) > 0. \tag{5.8}$$

We will need the following lemma, proved in Lemma 3.8 of [9].

LEMMA 5.1. *For* $U_n \in J$, $\|f(U_{n+1})\| \leq \overline{K}\|f(U_n)\|$ *for some* $\overline{K}(J, D) > 0$.

We also need the following Lemma which proves that for sufficiently small timesteps Δt it is impossible to move between the sets $\Psi_i(\epsilon)$ in a single step. The proof is given in the Appendix.

LEMMA 5.2. *If* $U_n \in J$ *then there exists* $\lambda(J) > 0$ *such that if*

$$\Delta t_n \leq \frac{\lambda}{\|f(U_n)\|}$$

then

$$\|U_{n+1} - U_n\| \leq \gamma.$$

The next two lemmas prove the upper bound on the sequence of accepted timesteps Δt_n. Once again the proofs are given in the Appendix.

LEMMA 5.3. *Let* $U_n \in J_{i_n,\epsilon^*}$. *Then all* $\Delta t_n \in (0, D]$ *satisfying*

$$E_{i_n}(U_n, \Delta t_n) \leq \sigma(\tau, U_n) \text{ and } \Delta t_n \leq \min(\frac{\lambda}{\|f(U_n)\|}, \frac{\epsilon^*}{2C_4\|f(U_n)\|})$$

also satisfy

$$\Delta t_n^3 \leq \frac{2\sigma(\tau, U_n)}{\|f(U_n)\|\epsilon^*}. \tag{5.9}$$

Also, if $U_{n+1}, U_{n+2} \in J$, *then for all sufficiently small* τ,

$$\Delta t_{n+1} \leq \min(\frac{\lambda}{\|f(U_{n+1})\|}, \frac{\epsilon^*}{2C_4\|f(U_{n+1})\|}), \tag{5.10}$$

$$\Delta t_{n+1}^3 \leq \frac{2\overline{K}C_2\alpha^3\sigma(\tau, U_{n+1})}{C_1\|f(U_{n+1})\|\epsilon^*} \text{ and} \tag{5.11}$$

$$\Delta t_{n+2} \leq \min(\frac{\lambda}{\|f(U_{n+2})\|}, \frac{\epsilon^*}{2C_4\|f(U_{n+2})\|}). \tag{5.12}$$

LEMMA 5.4. *Let* $U_n \in J$ *for all* $n = M(\tau), \ldots N(\tau)$, *and* $N \geq M + 2$. *Then if* $U_M \in J_{i_M,\epsilon^*}$ *and*

$$\Delta t_M \leq \min(\frac{\lambda}{\|f(U_M)\|}, \frac{\epsilon^*}{2C_4\|f(U_M)\|}) \tag{5.13}$$

it follows that for τ *sufficiently small and for all* $n = M, \ldots, N$

$$\Delta t_n \leq \min(\frac{\lambda}{\|f(U_n)\|}, \frac{\epsilon^*}{2C_4\|f(U_n)\|}) \tag{5.14}$$

TABLE 1

The number of accepted steps, rejected steps and number of flops are displayed for the algorithm (2.1)—(2.8) with $c = \frac{1}{2}$ and $c = \frac{1}{3}$ separately and then when these values are used together in the alternating algorithm.

	$c = \frac{1}{2}$ / $\frac{1}{3}$ / Alternating $c = \frac{1}{2}$ and $\frac{1}{3}$		
τ	# steps	# rej. steps	flops
10^{-4}	23 / 24 / 25	1 / 2 / 2	1329 / 1459 / 1556
10^{-6}	93 / 95 / 99	2 / 2 / 3	5436 / 5648 / 6083
10^{-8}	421 / 427 / 445	3 / 3 / 1	24554 / 25332 / 26967
10^{-10}	1942 / 1969 / 2054	3 / 2 / 1	112772 / 116262 / 124311

and

$$\Delta t_n^3 \le \frac{2\sigma(\tau, U_n)}{\|f(U_n)\|\epsilon^*} \max(1, \frac{\alpha^3 C_2 \overline{K}}{C_1}). \tag{5.15}$$

To obtain a convergence result it is necessary to prove that the upper bound (5.15) holds for all $n \ge 0$ and so we now consider the choice of the initial timestep estimate Δt_{init}. If we choose $\Delta t_{\text{init}} \to 0$ as $\tau \to 0$ it can be seen that, for sufficiently small τ, the conditions of Lemma 5.4 must hold with either $M = 0$ or $M = 1$. If it holds with $M = 0$ then (5.15) does indeed hold for all n. If not then (5.15) holds for all $n \ge 1$ and for sufficiently small τ,

$$\Delta t_0 \le \min(\frac{\lambda}{\|f(U_0)\|}, \frac{\epsilon^*}{2C_4\|f(U_0)\|})$$

implying that (5.15) also holds for $n = 0$ by Lemma 5.3. We now have upper bounds on the entire timestep sequence giving corresponding upper bounds on the truncation errors at each step. A Gronwall argument identical to that in [9] gives the following convergence result.

THEOREM 5.1. *Consider the numerical approximation of (1.1) over the time interval $[0, T]$ generated by the algorithm (5.4)–(5.7) and let Assumptions 1 and 4 hold. Then there exists a constant $K = K(T)$, such that for all sufficiently small τ and sufficiently small Δt_{init},*

$$E_n = \|u(t_n) - U_n\| \le K\frac{\tau}{\epsilon^*}.$$

The algorithm (5.4)–(5.7) is merely the simplest implementation of the alternating methodology. We can in fact choose $\Delta t_n^{(0)}$ to be strictly less than the expression in (5.5) and the statements and proofs of Lemmas 5.3, 5.4 and Theorem 5.1 are unchanged. Efficiency considerations, such as attempting to reduce the number of (computationally wasteful) rejected timesteps, suggest other choices for $\Delta t_n^{(0)}$. The one that we shall use for our numerical comparisons is

$$\Delta t_n^{(0)} = \min\left(D, \alpha\Delta t_{n-1}, \Delta t_{n-1}^{(0)}, \theta\left(\frac{\sigma(\tau, U_{n-1})}{E_{i_{n-1}}(U_{n-1}, \Delta t_{n-1})}\right)^{\frac{1}{3}} \Delta t_{n-1}, T - t_n\right) \text{ if } n \ge 1. \tag{5.16}$$

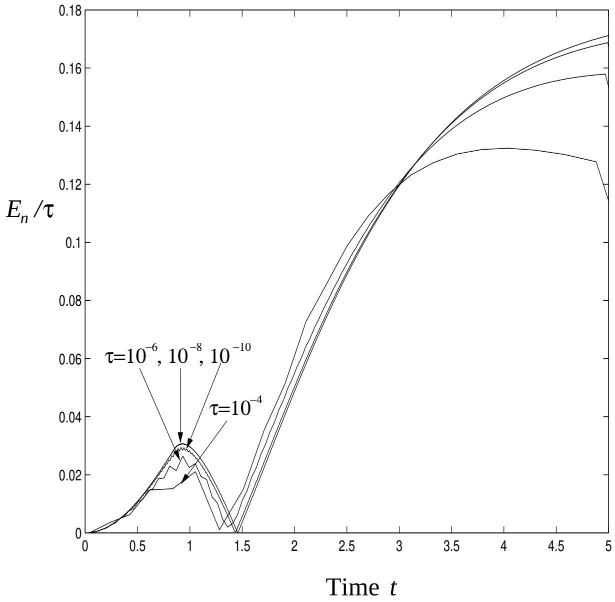

FIG. 6. *This Figure should be compared with Figure 3. By alternating between $c = \frac{1}{2}$ and $c = \frac{1}{3}$ Tolerance Proportionality has been recovered in accordance with Theorem 5.1.*

TABLE 2
The same statistics as Table 1 are displayed for a more typical problem where TP holds individually for $c = \frac{1}{2}$ and $c = \frac{1}{3}$. The statistics for the alternating algorithm are almost identical to those for the non-alternating algorithms.

	$c = \frac{1}{2}$ / $\frac{1}{3}$ / Alternating $c = \frac{1}{2}$ and $\frac{1}{3}$		
τ	# steps	# rej. steps	flops
10^{-4}	32 / 32 / 33	0 / 0 / 0	3013 / 3044 / 3190
10^{-6}	135 / 138 / 138	0 / 0 / 0	13068 / 13497 / 13703
10^{-8}	619 / 629 / 630	0 / 0 / 0	60016 / 61615 / 62657
10^{-10}	2863 / 2910 / 2911	0 / 0 / 0	277684 / 285153 / 289617

This choice should reduce the number of rejections that occur when the two error estimates E_1 and E_2 are significantly different in magnitude.

We now return to the numerical example of Figure 3 where TP breakdown occured for $c = \frac{1}{2}$; this breakdown also occurs for $c = \frac{1}{3}$ but at different values of u. We now integrate using the switching methodology with $c = \frac{1}{2}$, $c = \frac{1}{3}$ and (5.16) in place of (5.5). The maximum timestep ratio is set at $\alpha = 5$ and the convergence behaviour is plotted in Figure 6 and shows that TP has indeed been restored. Table 1 records the number of timesteps used in the numerical solution, the number of rejected steps and the total number of flops (as calculated by the MATLAB *flops* variable) for the alternating algorithm and the algorithm (2.1)—(2.8) using $c = \frac{1}{2}$ and $c = \frac{1}{3}$ separately.

The above numerical integration represents a severe test for the new algorithm. Nevertheless TP has been restored, with a corresponding improvement in accuracy, at the cost of just a few percent extra computational effort.

Finally, we use the alternating algorithm to repeat the integration from Figure 2 and in Table 2 make a similar comparison of the work done. In this example the original algorithm exhibits TP for both $c = \frac{1}{2}$ (see Figure 2) and $c = \frac{1}{3}$ and TP also occurs for the alternating algorithm. Again, the increase in computational effort is small. More sophisticated alternatives to (5.16), as well as a detailed investigation into possible choices for the Runge-Kutta pairs, should lead to even more efficient algorithms.

APPENDIX

Lemma 5.2.

Proof. Let us consider the first advancing Runge-Kutta method $S_1^{(1)}$. From Lemma 4.2.6 in [14] there exist positive constants $b_j(J,D), c_j(J,D),\ j = 1,\ldots,3$ such that

$$\begin{aligned}\|S_1^{(1)}(U_n,\Delta t_n) - U_n\| &\le \Delta t_n \|f(U_n)\| \left(\sum_{j=1}^{3} b_j(1 + c_j \Delta t_n)\right)\\ &\le \Delta t_n \|f(U_n)\| \left(\sum_{j=1}^{3} b_j(1 + c_j D)\right).\end{aligned}$$

So there exists $\lambda_1(J) > 0$ such that for all $U_n \in J$,

$$\Delta t_n \le \frac{\lambda_1}{\|f(U_n)\|} \Rightarrow \|S_1^{(1)}(U_n,\Delta t_n) - U_n\| \le \gamma.$$

There exists a similar constant $\lambda_2 > 0$ for the second advancing Runge-Kutta method $S_2^{(1)}$ and choosing $\lambda = \min(\lambda_1,\lambda_2)$ completes the proof. ∎

Lemma 5.3.

Proof. We first prove (5.9). If $E(U_n,\Delta t_n) \le \sigma(\tau,U_n)$ and $U_n \in J_{i_n,\epsilon^*}$ then

$$\Delta t_n^3 \left\|\Big(b_{i_n,1}(U_n) + \Delta t_n \|f(U_n)\| \tilde{b}_{i_n,2}(U_n,\Delta t_n)\Big)\right\| \le \frac{\sigma(\tau,U_n)}{\|f(U_n)\|}.$$

But

$$\begin{aligned}\left\|\Big(b_{i_n,1}(U_n) + \Delta t_n \|f(U_n)\| \tilde{b}_{i_n,2}(U_n,\Delta t_n)\Big)\right\| &\ge \|b_1(U_n)\| - \Delta t_n \|f(U_n)\| \|\tilde{b}_{i_n,2}(U_n,\Delta t_n)\|\\ &\ge \epsilon^* - C_4 \Delta t_n \|f(U_n)\|\\ &\ge \epsilon^*/2.\end{aligned}$$

Thus $\epsilon^* \Delta t_n^3 \le \frac{2\sigma(\tau,U_n)}{\|f(U_n)\|}$ and the result follows. The proof of (5.10) and (5.11) uses the maximum timestep ratio α. We have from (5.9) that

$$\Delta t_{n+1}^{(0)} \leq \alpha \left(\frac{2\sigma(\tau, U_n)}{\|f(U_n)\|\epsilon^*} \right)^{\frac{1}{3}}$$
$$\leq \alpha \left(\frac{2C_2\overline{K}\tau}{\|f(U_{n+1})\|\epsilon^*} \right)^{\frac{1}{3}}$$

which proves (5.11) and finally this is

$$\leq \min(\frac{\lambda}{\|f(U_{n+1})\|}, \frac{\epsilon^*}{2C_4\|f(U_{n+1})\|})$$

for sufficiently small τ using the fact that $\frac{1}{3} < 1$ which proves (5.10). The proof of (5.12) is precisely the same as for (5.10) but with α^2 replacing α. $\square$

Lemma 5.4.

Proof. By Lemma 5.3, (5.14) and (5.15) hold for $n = M, M+1$ and so we proceed by induction. We suppose that (5.14) and (5.15) hold for $n = M, M+1, \ldots, p < N$ and show that it holds for $p+1$. Suppose first that $U_p \in J_{i_p,\epsilon^*}$. Then by the inductive hypothesis and Lemma 5.3 with $n = p$ the result holds for $n = p+1$. So suppose instead that $U_p \notin J_{i_p,\epsilon^*}$. Then, by the inductive hypothesis and Lemma 5.2, $\|U_{p-1} - U_p\| \leq \gamma$ and so $U_{p-1} \in J_{i_{p-1},\epsilon^*}$. Thus by Lemma 5.3 with $n = p-1$, for sufficiently small τ,

$$\Delta t_p \leq \frac{\lambda}{\|f(U_p)\|}$$

and so $\|U_{p+1} - U_p\| \leq \gamma$. Therefore $U_{p+1} \in J_{i_{p+1},\epsilon^*}$ and Lemma 5.3 applies with $n = p+1$, proving the result for $p+1$. This completes the proof. $\square$

REFERENCES

[1] L. Blum, F. Cucker, M. Shub, and S. Smale, *Complexity and real computation*, Springer-Verlag, New York, 1997.
[2] M. Calvo, D. Higham, J. Montijano, and L. Rández, *Stepsize selection for tolerance proportionality in explicit Runge-Kutta codes*, Advances in Computational Mathematics, 7 (1997), pp. 361–382.
[3] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*, McGraw-Hill,, 1990.
[4] J. Demmel, *On condition numbers and the distance to the nearest ill-posed problem*, Num. Math., 51 (1987), pp. 251–289.
[5] A. Edelman, *On the distribution of a scaled condition number*, Math. Comp., 58 (1992), pp. 185–190.
[6] D. Higham, R. Wain, and A. Humphries, *Phase space error control for dynamical systems*. Research Report CMAIA97-05, University of Sussex, 1997.
[7] C. Hoare, *Quicksort*, Computer Journal, 5 (1962), pp. 10–15.
[8] H. Lamba, *Dynamical systems and adaptive time-stepping ODE solvers*. SCCM Technical report, Stanford University, 1998.
[9] H. Lamba and A. Stuart, *Convergence results for the MATLAB ode23 routine*. To appear, BIT.
[10] S. Smale, *The fundamental theorem of algebra and complexity theory*, Bull. AMS, 4 (1981), pp. 1–35.
[11] H. Stetter, *Considerations concerning a theory for ODE solvers*. Numerical Treatment of Differential Equations. Lecture Notes in Mathematics 631, Springer-Verlag, Berlin, 1976.

[12] ———, *Tolerance proportionality in ODE codes.* Proc. second Conf. on Numerical Treatment of Ordinary Differential Equations, Humboldt University, 1980.
[13] A. Stuart, *Probabilistic and deterministic convergence proofs for software for initial value problems*, Numerical Algorithms, 14 (1997), pp. 227–260.
[14] A. Stuart and A. Humphries, *Dynamical systems and numerical analysis*, CUP, 1996.