Error Estimation for Numerical Differential Equations

Donald J. Estep[1], Sjoerd M. Verduyn Lunel[2], and Roy D. Williams[3]

Given a capable human being and a computer, it is possible to make an approximation to the solution of a nonlinear differential equation. However, with the (usually correct) assumption that the equation is analytically intractable, the result of the computation is not the exact solution; indeed, it may be so far from the exact solution as to be completely useless. We are interested in the relationship between the effort expended by the human and the computer and the quality of the computed approximation to a partial or ordinary differential equation. To be specific, we would like to think in terms of a cost-benefit analysis. The cost of the computation is a combination of the human effort and computer resources used to obtain the approximation. The benefit includes not only the computed approximation, but it also includes an estimate of the *quality* of the approximation, that is, an error estimate. It is our opinion that in computational science, as with the experimental sciences, results should always be presented with some estimate of their accuracy. In addition, however, there is another facet to error estimation: one cannot even attempt a cost-benefit analysis or efficiency comparison of methods without an error estimate to evaluate the results.

Let $y(t)$ be the exact solution of a partial or ordinary differential equation

$$\frac{\partial y}{\partial t} + f(y,t) = 0,$$

with suitable initial and boundary conditions, where f may involve space derivatives. We compute a numerical approximation Y with respect to a method of lines discretization of the space-time domain. We use $h(x,t)$ and $k(t)$, respectively, to denote the size of the space and time mesh at position (x,t). We emphasize that these may vary at different points in time and space, for example the timestep from time t to $t+k$ may change from one step to the next. We compute an approximation by choosing a function from a finite dimensional space (such as polynomials) that approximately satisfies the differential equation in both space and time. For example, the numerical solution might satisfy an approximation of the differential equation in the form of a difference scheme or it might be chosen to be the best solution of the equation among all functions in the finite dimensional space in a finite element method. The error is $e = y - Y$ and this article is about estimating its norm $||e||$. The norm may be chosen to emphasize error at some particular time and point in space, an average over the whole space-time domain, or something else.

In the classical *a priori* error theory, we attempt to estimate the error before computing the approximation by first bounding the error produced in a step and then assuming that

[1]School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332, USA, estep@math.gatech.edu

[2]Faculteit Wiskunde en Informatica, Universiteit van Amsterdam, Plantage Muidergracht 24, 1018 TV Amsterdam, Netherlands, verduyn@fwi.uva.nl

[3]Center for Advanced Computing Research, California Institute of Technology, Pasadena, CA 91125, USA, roy@cacr.caltech.edu

errors accumulate in the worst possible way to bound the global error. We measure the error of the approximation over a single timestep as an interpolation error, $\mathcal{I}$, that is bounded using Taylor's theorem

$$\mathcal{I} \leq C_n \|h^n D^n y\| + C_m k^m \|D^m y\|,$$

where D^n and D^m denote a combination of space and time derivatives of the appropriate order for the accuracy, C_n and C_m are constants that depend on the method for computing Y, and the norms are localized to the space domain and the current timestep.

We must then bound the effects of the accumulation of such errors over all the subsequent intervals in order to bound the total error. We can analyze the effect of perturbations in a solution of a nonlinear problem by studying the differential equation obtained by linearizing around the solution. The coefficient matrix for the linearized problem is the Jacobian matrix $\partial f/\partial y$. If we write J_{max} as the maximum norm of the Jacobian over "all possible values of y", we can obtain a global error bound:

$$||e|| \leq e^{J_{\max} t} \max \mathcal{I}.$$

This is the traditional form of the *a priori* error bound, where the maximum is over all previous timesteps. We call the exponential prefactor a *stability factor* because it reflects the stability of the solution to perturbations. It depends on both the solution and the numerical method. In this *a priori* error bound, the stability factor is an exponential, meaning that the error bound grows rapidly.

We would like to point out two things about this kind of result. The first is the list of unknown, unknowable quantities: the high-order derivatives of the exact solution in the Taylor estimate, and the range of the solution that is required to compute J_{max}. The second is that the result is pessimistic; it assumes that the error always accumulates constructively rather than destructively, hence the growth of a perturbation is as rapid as possible. It is this kind of error bound which spits in our faces by telling us that error of a well-crafted CFD computation, for example, grows as something like $\exp(10^{15}t)$. In the face of this, we can choose between giving up computation altogether or paying no attention to the error bound.

We now describe an alternative way to compute error bounds (see [1] and [3] for more details). The goal is to obtain a *sharp* error bound: one that is within a factor two or three of the true error rather than different by hundreds of orders of magnitude. This superior bound entails more computation because a linear system must be solved for each error estimate, in addition to the solution of the nonlinear differential equation itself. It is an *a posteriori* bound, meaning that the error is computed after the computation rather than before. The bound has two parts: one measures the errors committed locally at each point in space-time; and the other measures the effects of the accumulation of such errors. We begin with the local behavior.

In the *a priori* analysis, we ask *"How close is the approximation to the true solution?"*. This question is answered using Taylor's theorem, and the result, as stated above, is a formula that is difficult to compute practically. Instead let us decide what we *can* compute, and then relate this to the error norm. We ask the question *"How well does the approximation satisfy the differential equation?"*. The answer is given by the residual $\mathcal{R}$, defined as:

$$\mathcal{R} = \|\dot{Y} + f(Y,t)\|.$$

This quantity is computable, since it is written in terms of low-order derivatives of the approximation. $\mathcal{R} = \mathcal{R}_x + \mathcal{R}_t$ decomposes into space and time residual errors naturally.

The other part of the error bound, measuring the accumulation of error, is a property of the equation and its true solution. The formulae relating residuals to error may be written generically as:

$$||e|| \leq S(t) \max \mathcal{R}.$$

where S is a stability factor and the maximum is over all steps.

There are different stability factors for different kinds of error, such as the error in discretization, and the error in initial conditions. These may be computed at some time t by integrating the linearized system:

$$\begin{cases} \dot{z}(s) + (\partial f/\partial y)(y(s), s)z(s) = 0, \quad t > s \geq 0, \\ z(t) = e(t) \end{cases}$$

where the Jacobian of f is evaluated at the solution y of the nonlinear system. The integration is backwards in time from t to zero, then S is expressed as a time-integral of a norm of z.

We would like to draw an analogy between these error bounds and the process of solving a system of linear equations $Ax = b$, where A is a matrix and x and b are vectors. To determine the quality of an approximate solution X, the direct approach – estimating the error $\|X - x\|$ – is of course not computable. The alternative is to work with the residual $\|AX - b\|$, which is computable. The relationship between error and residual is determined by the *condition number* of the matrix: the error is bounded by condition number times residual. The stability factor plays exactly this role for differential equations: it is the "condition number" of the solution of the differential equation that we approximate.

The stability factors are not explicitly computable, because they are a property of the (unknown) solution. Instead, we compute approximate stability factors from the numerical approximation of the differential equation; justification of these approximations and the study of how to compute them efficiently is an area of current research [1][3].

Mesh Adaptivity

We would like to choose a dynamically adaptive discretization which is as coarse as possible to reduce the required computer resources, but at the same time keeping the error bound below a desired tolerance. The error bound is written as a stability factor multiplied by a maximum of the local residual error. As a result, we can be efficient by keeping the local residual error roughly constant on each local component of the space-time discretization. Having done this, we can evaluate the stability factor, and hence the error bound.

The procedure above provides the error bound after the computation is finished, because we cannot compute either residual or stability factor until then. If, however, we want to specify an error bound in advance, we must use a predictor-corrector strategy. If the computed error bound is larger than the specified error bound, the computation is redone with a finer discretization; if too small, we change our strategy for subsequent computations. See [1].

This Seems Rather Complicated

But surely this is all much more difficult than it needs to be – why not simply run a canned solver with smaller and smaller timesteps, and using finer and finer spatial meshes, until eventually "it seems to have converged"? The end result of this approach is an approximation together with numerical evidence that the approximation is not changing with the mesh spacings. Of course it may indeed be a very accurate approximation, but we do not know this because there is no estimate of error. On the other hand, it may be very inaccurate: there are plenty of examples in the literature of systems that are not pathological, but where the approach of reducing the discretization until there is no further change can fail because of false plateaux, providing a mirage of convergence. This "brute force" method also seems to be *inefficient* in the following sense: if the approximation is more accurate than the desired accuracy, then perhaps we should have used less resources and obtained a result that is sufficiently but not excessively accurate. On the other hand, if the approximation is not accurate enough, then it is not of much use and we should not have spent our resources computing it.

In terms of the cost-benefit analysis, the brute force method has a human cost, which is the most expensive part of the human-computer team. The human must evaluate the results of each run, decide if "convergence" has occurred, and if not, choose a new mesh spacing. Surely it would be easier if the computer ran a little longer, yet came up with both an approximation to the solution and a quantitative error estimate.

Numerical Results

We present some computations that illustrate the ideas in this article. All of the results were computed with the error norm at 1% or better of the solution norm. Because stability factors depend only on the equation and its solution, they are not only useful for making sharp error bounds, but are also important tools for understanding the behavior of a solution of a nonlinear problem, which in general means not only knowing the values of the solution, but also knowing how perturbations to the solution behave. The stability factors are a probe to do this. To illustrate the great variety of stability factors encountered in nonlinear models, we present them for a selection of nonlinear problems. In each case, the classical *a priori* bound, which grows exponentially, is misleading.

1. For the dissipative problem in Fig. 1(a), the standard *a posteriori* bound gives a stability factor that converges to a fixed constant. With the proper choice of scheme, the error actually tends to zero, and a specialized analysis for this problem can be done. The *a priori* error factor however gives a bound of order $\exp(100t)$.

2. The stability factor for the Kepler problem, Fig. 1(b), grows on average quadratically, but oscillates around this average with the period of the satellite, with the peaks coinciding with perihelion of the orbit. The stability factor increases and decreases exponentially inside each period, whereas the *a priori* estimate "sees" only the exponential increase.

3. In the Lorenz system, Fig. 1(c), solutions oscillate between revolving around one of the two fixed points, with the chaotic nature reflected in the seemingly random number of revolutions around one fixed point before switching to the other. The *a priori* error bound predicts error growth on the order of $\exp(90t)$. However, when a solution trajectory is orbiting one of the two fixed points (which is most of the time), then the error growth is on the order of $t^{1.4}$. When a solution switches from orbiting one fixed point to the other, then the error grows rapidly. The first time this happens

in the plotted solution is at $t = 16$, and the growth is reflected in the growth of $S(t)$. This rapid increase in sensitivity causes the long time growth to be exponential on average at a rate of approximately $\exp(.9t)$.

4. The forced Duffing equation has two regimes of solutions. Certain initial conditions lead to a chaotic solution, while the rest of the solutions converge to a stable periodic solution. In Fig. 1(d), we plot a chaotic solution while 1(e) illustrates a solution that converges to the periodic solution. The *a priori* error analysis treats both solutions the same, predicting the usual exponential growth, yet it is clear from the figures that the stability of the chaotic and periodic solutions are vastly different.

The Bistable Equation

The bistable problem with Neumann boundary conditions in one dimension reads:

$$\begin{cases} y_t - \epsilon^2 y_{xx} = y - y^3, & 0 < x < 1, 0 < t, \\ y_x(0,t) = y_x(1,t) = 0, & 0 < t, \\ y(x,0) = y_0(x), & 0 < x < 1. \end{cases} \tag{1}$$

The bistable equation is an illustrative example of nonlinear relaxation to equilibrium in the presence of competing stable steady states. It is a model of many physical phenomena, including motion of domain walls in ferromagnetic materials. The stable steady states are $y \equiv 1$ and $y \equiv -1$, which are minimizers of the energy functional $\int_0^1 \left(\epsilon^2 y_x^2/2 + (y^2-1)^2/4\right) dx$. For generic initial data, $\lim_{t\to\infty} y(x,t)$ is one of these steady states. However, this convergence can be extremely slow because solutions of (1) can exhibit dynamic *metastability*. In general, y forms a pattern of transition layers between the values 1 and -1, where the thickness of each layer is of order ϵ. The subsequent time scale for substantial motion of the layers is $\exp(Cd/\epsilon)$, where $C = O(1)$ and d is the minimum distance between layers or between the layers and the boundaries. Metastable solutions are not local minimizers of the energy, and thus are always dynamic. After a metastable period, one or more of the layers disappear in a relatively quick transient. The system then forms a new metastable pattern. This repeats until the eventual convergence to a steady state. The standard *a priori* analysis predicts that errors accumulate at an exponentially growing rate on the order of $\exp(Ct/\epsilon^2)$ ($\approx \exp(1000t)$ for the ϵ chosen below), essentially ruling out accurate computation. The *a posteriori* error bound suggests that accurate computation is possible for long times including both transients and metastable periods.

We discretize (1) in space using a second-order finite element method based on piecewise linear functions and lumped mass quadrature and using a finite element approximation in time. We plot the evolution of a solution starting from data with two metastable "wells" with $\epsilon = .03$ in Fig. 2(a). The left well of the initial data is slightly thinner than the right and collapses by the sides coming together around time 41, while the well on the right collapses at time 141. The solution exhibits metastability during the time before 41 and between the two transients. In Fig. 2(b), we plot the stability factor $S(t)$. S reflects the alternation of metastable periods, where $S(t)$ is of order 1 and slowly increasing, with the transient periods, which are predicted by the increase in S. After the transient, the stability factor drops precipitously, indicating that the subsequent solution is essentially independent of any previous error accumulation. When the solution finally converges to the uniform equilibrium state, the stability factor is one, and all previous error due to accumulation is removed.

Thermoviscoplastic Shear Band Formation

When a fluid undergoes shear, there is viscous heating and a consequent rise in temperature. The shear rate and heating may be spread uniformly through the sheared fluid or may be localized in a narrow "shear band", depending on how the viscosity of the fluid varies with temperature. Formation of a shear band depends on the viscosity decreasing sufficiently quickly with temperature, in which case all the shear and consequent heating is concentrated in the band. Most of the material is very viscous and undergoing no deformation, and there is a thin, hot, low viscosity layer lubricating the shear. The main features of this phenomenon have been modeled by assuming a power-law dependence of viscosity on shear rate; in the following example, we choose an inverse-square dependence. If the shear happens very quickly, such as in a plastic-forming die or at a geological fault during an earthquake, then thermal diffusion cannot dissipate the heat quickly. In the limiting case of no thermal diffusion, there is a debate ([2]) over whether the shear-band model has singular solutions in finite time.

We transform the momentum and energy equations as follows:

$$\begin{cases} s_\tau - \frac{1}{r}e^{\tau/3}s_{xx} = \frac{2}{3}s\big(1-3\sqrt{r}s^2\big), \\ r_\tau \qquad\qquad\;\; = -\frac{2}{3}r\big(1-3\sqrt{r}s^2\big), \\ s_\tau(0,\tau)=0, s_\tau(1,\tau)=0, \\ s(x,0)=s_0(x), r(x,0)=r_0(x), \end{cases}$$

for s and r representing the transformed shear stress and temperature respectively, while τ is the logarithm of the time. This non-physical form has certain advantages for mathematical analysis, including the existence of invariant regions and a conserved quantity. We discretize with a specially constructed finite difference scheme that preserves this structure.

The initial data has a smooth bump of height 1 and width 0.04 centered at $x = 0.8$, as well as a small undulating perturbation throughout the x-domain. The perturbation was added to test the structural stability of the shear band: we find that the band has the same form with or without the perturbation.

In Fig. 3, we show the results of the numerical solution: the transformed temperature, r, is plotted with a log scale with 8 orders of magnitude, against x and τ. We can see the temperature increasing in a narrow band where the initial conditions provided a nucleation point. The computation was done with a time- and space-adaptive mesh, which has the effect of maximizing the efficiency of the computation by concentrating computer resources where necessary, which turns out to be at the peak.

In Fig. 4(a), we plot the maximum height of r versus τ for computations made on fixed meshes with $32, 64, ...,$ and 16384 space nodes, together with the results of a computation made with adaptive error control. The fixed mesh computations are plotted with dashed lines and the solid line shows the results of the adaptive computation. Using fixed meshes leads to incorrect conclusions about the rate of growth of the peak height of r unless an extremely fine discretization is used. Figure 4(b) illustrates how the number of elements used in the adaptive computation increases with time. We get better results with far fewer nodes in space. The placement of the nodes is visible in the plot of the solution in Fig. 3.

Acknowledgments

The work of D. Estep is supported by the National Science Foundation, contract numbers DMS-9208684 and INT-9302016. The work of R. D. Williams is supported by Caltech, the Concurrent Supercomputing Consortium, and the Center for Research on Parallel Computation (a National Science Foundation Science and Technology Center, Grant 292-3-51393 under NSF CCR-8809615).

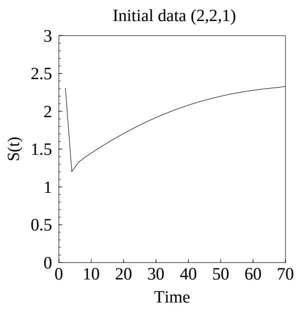
References

[1] D. J. Estep and R. D. Williams, *Accurate Parallel Integration of Large Sparse Systems of Differential Equations*, to be published in Math. Models. Meth. Appl. Sci., also available on the Internet from `http://www.cacr.caltech.edu/~roy/cards/`.

[2] M. Bertsch, L. A. Peletier, and S. M. V. Lunel, *The Effect of Temperature-dependent Viscosity on Shear-flow of Incompressible Fluids*, SIAM J. Math. 22 (1991), 328 - 343.

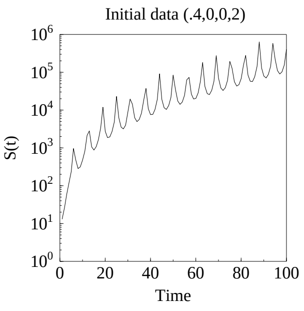
[3] D. J. Estep, *A posteriori error bounds and global error control for approximations of ordinary differential equations.* SIAM J. Numer. Anal. 32 (1995), 1–48.

(a) A stiff dissipative problem with three different time scales. Zero is a stable solution.



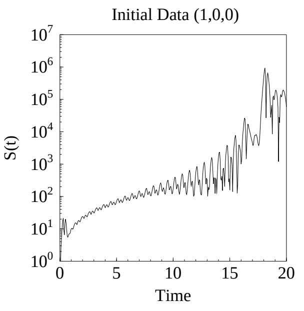
$$\begin{cases} \dot{y}_1 = -.01y_1 - .99y_2 + .99y_3, \\ \dot{y}_2 = -y_2 + 99y_3, \\ \dot{y}_3 = -100y_3. \end{cases}$$

(b) The Kepler two-body problem for a satellite in an eccentric, periodic orbit around a planet.



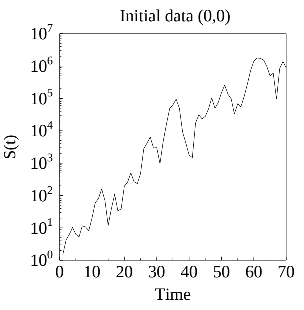
$$\begin{cases} \dot{y}_1 = y_3, \\ \dot{y}_2 = y_4, \\ \dot{y}_3 = -y_1/(y_1^2 + y_2^2)^{3/2}, \\ \dot{y}_4 = -y_2/(y_1^2 + y_2^2)^{3/2}. \end{cases}$$

(c) The Lorenz system, which is a well-known model for chaotic behavior.



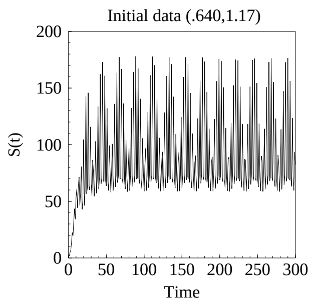
$$\begin{cases} \dot{y}_1 = -10y_1 + 10y_2, \\ \dot{y}_2 = 28y_1 - y_2 - y_1y_3, \\ \dot{y}_3 = -\frac{8}{3}y_3 + y_1y_2. \end{cases}$$

(d) A chaotic solution of the forced Duffing problem, which is another model for chaotic behavior.



$$\begin{cases} \dot{y}_1 = y_2, \\ \dot{y}_2 = y_1 - y_1^3 - .15y_2 + .3\cos(t). \end{cases}$$

(e) A solution of the forced Duffing problem that converges to the stable periodic solution.



$$\begin{cases} \dot{y}_1 = y_2, \\ \dot{y}_2 = y_1 - y_1^3 - .15y_2 + .3\cos(t). \end{cases} .$$

Figure 1: The Stability Factor Gallery

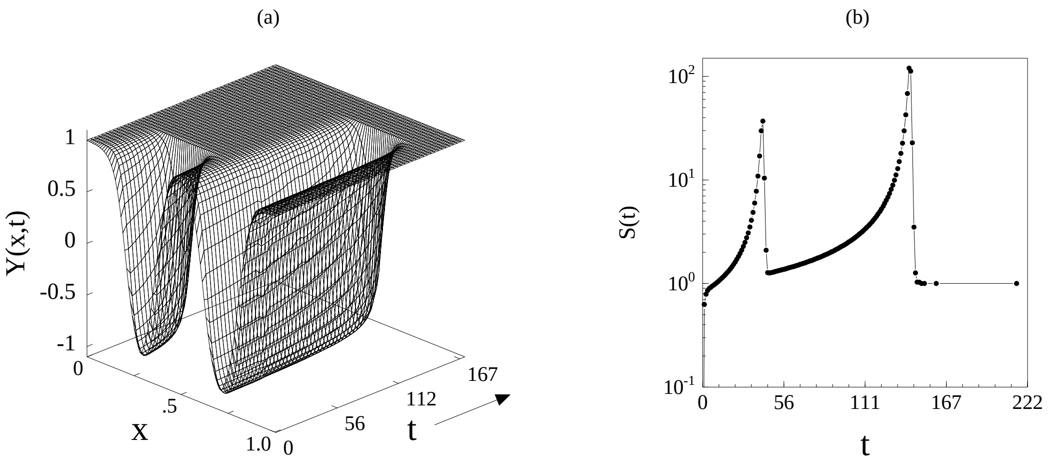


Figure 2: Evolution of a metastable solution of the bistable equation and its stability factor, $\epsilon = 0.03$. Transients are marked by rapid growth of the stability factor. During the metastable periods of extremely slow change, the solution is relatively stable to perturbations. When it has converged to 1, it is insensitive to the accumulation of error.

Figure 3: Evolution of a shear band. The initial profile is a smooth bump of size 1 at $x = 0.8$, together with an undulating perturbation. The shear band is structurally stable to this perturbation, and grows at the same rate as without the perturbation.

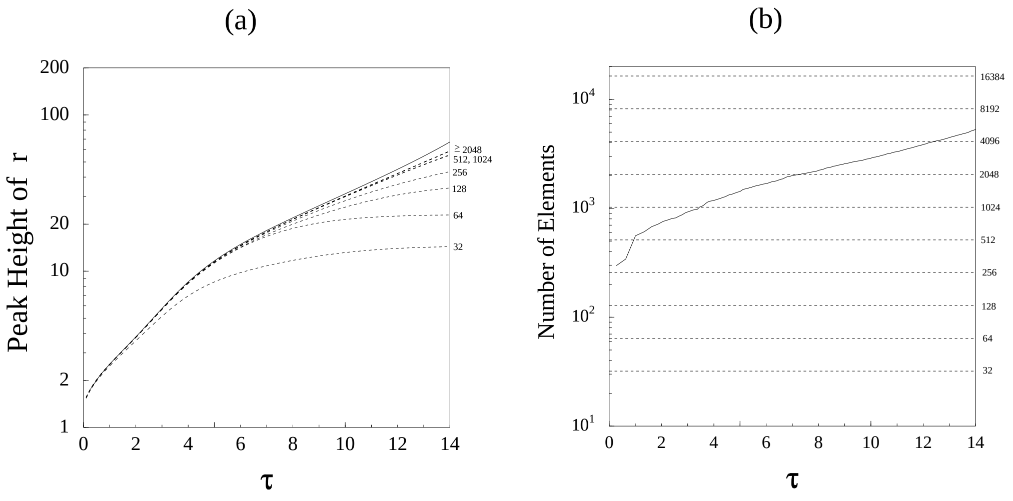


Figure 4: (a) Peak heights in r versus time for a series of fixed mesh discretizations (dashed lines) and one adaptive discretization (solid line). (b) The number of elements versus time for the different computations. The adaptive computation gives the correct dynamics using fewer elements and less computer time than the fixed mesh computations.