Cambridge International
Science Publishing
Cambridge CB1 6AZ
Great Britain

Journal of Computational Methods in Sciences and Engineering
vol. 2, no. 3, 2, pp. 1–3
ISSN 1472–7978

The Cost/Reliability Trade-off in Verifying Approximate Solutions to Differential Equations [1]

W.H. Enright[2]

Department of Computer Science
University of Toronto

Abstract:
It is now standard practice in computational science for large scale simulations to be implemented and investigated in a problem solving environment (PSE) such as MATLAB or MAPLE. In such an environment, a scientist or engineer will formulate a mathematical model, approximate its solution using an appropriate numerical method, visualize the approximate solution and verify (or validate) the quality of the approximate solution. Traditionally we have been most concerned with the development of effective numerical software for generating the approximate solution and several efficient and reliable numerical libraries are now available for use within the most widely used PSEs. On the other hand, the visualization and verification tasks have received little attention, even though each often requires as much computational effort as is involved in generating the approximate solution.

In this paper we will investigate the effectiveness of a suite of tools that we have recently introduced in the MATLAB PSE to verify approximate solutions of ordinary differential equations. In particular we will identify and illustrate the inherent trade-off between reliability and efficiency that arises. We will use the notion of 'effectivity index', widely used by researchers in the adaptive mesh PDE community, to quantify the quality of our verification tools and illustrate the performance of these tools on a two test problems.

Keywords: verification, ODEs, defect, numerical solutions, reliability

Mathematics Subject Classification: 65L05, 65Y20

[1] This research was supported by the Natural Science and Engineering Research Council of Canada.

[2] enright@cs.utoronto.ca

1 Introduction

In the last decade there has been a significant change in the way scientific computing is carried out. Those of us developing numerical methods must be aware of this change if our software is is ever going to be widely used and have an impact. The expectations and measures we use to evaluate software must evolve to make the results more relevant and applicable to modern scientific computing.

Two environments where numerical software is now widely used are:

- Large Scale Scientific Computation:
  Characterized by simulations generating massive amounts of data and involving expensive runs on High Performance distributed systems (often located at remote sites). Applications require data compression, interactive data viewing (or data mining), remote visualization and the distributed collaboration of experts.

- Problem Solving Environments (PSEs):
  Often where students are first exposed to scientific computing and where scientists and engineers formulate their mathematical models and investigate approximate solutions of these problems.

In this investigation we focus on the latter environment but the tools and approaches we develop and the observations we make are also applicable in the former environment. We will use the numerical solution of ordinary differential equations (ODEs), in particular initial value problems (IVPs) to illustrate the tools and evaluation criteria that can be developed and implemented. The approach can be adopted in a similar fashion for other problem classes that are supported in PSEs.

In the next section we will discuss the concept of a PSE and identify the characteristics or features that make a PSE most effective. We will consider the importance of verification tools (as an essential component of a PSE) and describe four such tools that we have implemented for use with ODE solvers. In the third section we describe how one can quantify the effectiveness of these tools by introducing the notion of an 'effectivity index' (a concept which has been developed and applied in the PDE community). Finally we provide numerical results assessing the credibility of these tools on two model problems and the method ode45 of the MATLAB PSE.

2 What makes a good PSE?

In a PSE the focus is on visualizing approximate solutions of mathematical models. This generally involves graphical representation of a 'subset' of the results. (eg. Phase plane or standard plots.) It can also involve the use of colour, texture, lighting, animation and sound. Stringent accuracy is not usually necessary, but off-mesh accuracy is often required.

Ease of use and robustness is very important (as important as efficiency) in a PSE. The interface and documentation must be simple and structured in a hierarchichal fashion. Information should be hidden unless necessary to define the problem. Options and additional parameters (if required) should be specified in the same way for all methods that perform similar tasks. This includes mesh-constraints, initial guess of solution, specification of partial derivatives, and accuracy specification.

Similarly there should be a standard representation (adopted by all methods of the PSE performing similar tasks) of what is meant by an approximate solution to a particular problem. For example, an approximate solution of an ODE can be represented in a PSE by a vector of piecewise polynomials. Such a representation makes it easy to compare, visualize and manipulate the results of simulations without having to pay attention to or understand how the approximations were generated.

A PSE should provide tools to:

1. Formulate and alter a mathematical model of the problem.
2. Approximate the solution of the mathematical model.
3. Visualize the approximate solution.
4. Verify that an approximate solution is consistent with the mathematical model and the requested accuracy.
5. Verify that the mathematical model is well-posed. (That is, quantify the underlying inherent conditioning.)

Generic tools for the first three of these tasks have been widely investigated and implemented as the key components of PSEs.

We have recently ([3]) developed a class of tools that can be used for the latter two tasks when the underlying mathematical model is a system of ODEs. In this investigation we will quantify and illustrate the effectiveness of these tools.

The key assumption we make of any method which generates an approximate solution is that, when applied to the IVP,

$$y' = f(x,y),\ y(a) = y_0, \text{ on } [a,b],$$

with a specified accuracy, TOL, the numerical method, M, generates a piecewise polynomial, $z_{TOL}(x)$, defined for $x \in [a,b]$ and satisfying,

$$\|z_{TOL}(x) - y(x)\| \leq K_M TOL.$$

This assumption is satisfied by most state-of-the-art ODE software and certainly by the built-in numerical methods of MATLAB [8]. Note that K_M can depend on the method and the problem and can be interpreted as the 'numerical condition number' associated with method M.

The restrictions we imposed on the verification tools we developed include:

1. The tool must be easy to use – If possible the calling sequence should be the same as that used to generate the approximate solution with no additional information required.
2. The 'cost' associated with applying the tool should not be much greater than that involved in generating the approximate solution (or visualizing it).
3. The calling sequence should be the same for any method it is applied with. The tool may require some knowledge of the method but this information should be transparent to the user.

The four verification tools we have implemented for ODEs in MATLAB are:

Check1: A consistency check based on solving the problem with a more stringent value for TOL and returning the difference in the two approximations as a piecewise polynomial. That is we compute,

$$E_{TOL}(x) = z_{TOL}(x) - z_{TOL1}(x),$$

where $TOL1 < TOL$. The 'cost' of applying this tool is about double the cost of generating $z_{TOL}(x)$.

Check2: A consistency check based on solving the problem with an alternative method, $\tilde{M}$. The piecewise polynomial returned by this routine is,

$$\tilde{E}_{TOL}(x) = z_{TOL}(x) - \tilde{z}_{TOL}(x)$$

where $\tilde{z}_{TOL}(x)$ is the approximate solution associated with $\tilde{M}$. (Note that this check is most credible if the method $\tilde{M}$ is very different from M. For example, in our implementation of this tool, we use a variable order Adams method when assessing a Runge Kutta method from MATLAB.)

Check3: A check which computes the defect,

$$D_{TOL}(x) = z'_{TOL}(x) - f(x, z_{TOL}(x)).$$

Note that $D_{TOL}(x)$ is not a piecewise polynomial, but it is a function that can be evaluated at any point in $[a, b]$.

Check4: A consistency check based on an idea of Zadunaisky [9] where we determine a piecewise polynomial,

$$EST_{TOL}(x) = w_{TOL}(x) - z_{TOL}(x),$$

where $w_{TOL}(x)$ is the approximate solution, produced by method M (applied to the perturbed IVP, $z' = f(x, z) + D_{TOL}(x)$ with known exact solution $z_{TOL}(x)$) on the same mesh that was used to determine $z_{TOL}(x)$.

For each of the vector-valued functions returned by Check1, Check2 and Check4 we can interpret its max value for $x \in [a, b]$ to be an estimate of the maximum global error. Therefore a crude indicator of the numerical condition number is the ratio of this max value to TOL. Similarly for Check3, the maximum magnitude of the components of $D_{TOL}(x)/TOL$ can be considered an indication of the contribution of the numerical method to the overall numerical conditioning of the problem (it does not reflect the underlying mathematical conditioning of the problem).

Each of these verification tools are scalable with respect to the dimension of the problem and the length of the integration (in the sense that the cost associated with applying the tool remains a small multiple of the cost associated with generating the approximate solution). Each is also straightforward to apply in a parallel computing environment where the underlying numerical method and the verification tool can be run simultaneously on independent processors with very little communication required between the processors.

In the numerical solution of partial differential equations (PDEs) there has been a similar (although more advanced) effort in developing efficient numerical methods and corresponding verification tools for special classes of problems. In the last decade there has been considerable interest in developing verification tools based on à posteriori error estimates for assessing the validity of a particular numerical solution. Such estimates are intended to be computed as a post processing task, and have been implemented and packaged with some of the widely used PDE PSEs. In particular, the finite element and adaptive mesh refinement research communities have adopted this point of view (see for example [1] and [2]). They have introduced the concept of an 'effectivity index' to quantify the credibility of their tools. We will adopt this approach in quantifying the credibility of the ODE verification tools we have introduced.

3 Effectivity Indices for ODE Verification Tools

For the ODE verification tool, Check3, the only error that arises is due to round-off and therefore this verification check will be reliable at all reasonable error tolerances. Because it does involve the subtraction of near equals (the pointwise evaluation of the defect) it will become unreliable at stringent tolerances (values of TOL near the unit round-off).

Each of the other verification tools (Check1, Check2 and Check4) can be interpreted as an estimate of the global error, $EST \approx TRUERR$. In this case the natural definition for the 'Effectivity Index' is,

$$EI = ||EST||/||TRUERR||.$$

Clearly, the closer this value is to 1, the more accurate and useful it will be. Too large a value indicates an over-estimate while too small a value indicates an under-estimate and the possibility that an inaccurate approximate solution will not be recognized. We will compute the value of EI for Check1, Check2 and Check4 for two test problems, a range of values of $ATOL$, and monitor its behaviour over the entire range of integration ($x \in [a, b]$).

In our numerical results we report,

$$CEST \equiv \frac{\max_{x\in[x_0,x_F]} \|EST(x)\|}{TOL}.$$

$$MnEI \equiv \min_{x\in[x_0,x_F]} EI(x).$$

$$MxEI \equiv \max_{x\in[x_0,x_F]} EI(x).$$

Our main objective, in developing these verification tools, is to produce an inexpensive but reasonably reliable tool for assessing the credibility of a particular approximate solution. For Check1 and Check2 the 'reliability' of these tools can be improved by setting the accuracy parameter associated with the secondary solution to be more severe. We have been somewhat arbitrary in setting the default for these values but, as the results of the next section will show, the overall reliability of these tools is quite good and not very sensitive to this choice. For Check4, the reliability of the tool is sensitive to the definition of the piecewise polynomial associated with the approximate solution and we have assumed that this is part of the definition of the method. For ode45 there are several possible choices for how this piecewise polynomial can be defined and the MATLAB implementation, while appropriate for most purposes, does not necessarily result in a small magnitude defect (especially at relaxed tolerances). Alternative definitions for this piecewise polynomial (for the same underlying discrete Runge-Kutta formula) are available which, at a small increase in cost per step, will result in a smaller magnitude defect and hence a more reliable verification tool.

4 Numerical Results and Conclusions

The first test problem corresponds to a recently discovered stable orbit that arises in the simulation of the restricted three-body problem (where the orbits are planar). The bodies have equal mass (in our case we assume $m_1 = m_2 = m_3 = 1.0$) and, with the appropriate starting conditions, will follow the same figure-eight orbit as a periodic steady-state solution.

The two spatial coordinates of the j^{th} body are y_{1j}, y_{2j} for $j = 1, 2, 3$. Each of the six coordinates satisfy a second-order differential equation,

$$y''_{ij} = \sum_{k=1,k\neq j}^{3} m_k \left(\frac{y_{ik} - y_{ij}}{d^3_{jk}}\right),$$

where

$$d^2_{kj} = \sum_{i=1}^{2} (y_{ik} - y_{ij})^2, \;\; k, j = 1, 2, 3.$$

When this system is re-written, as a first order system, the dimension of the problem is 12 and the initial conditions, at $x = 0$, are given by,

$$\begin{aligned} y_{11} &= -0.97000436, & y'_{11} &= 0.466203685, \\ y_{21} &= 0.24308753, & y'_{21} &= 0.43236573, \\ y_{12} &= 0.0, & y'_{12} &= -0.93240737, \end{aligned}$$

$$
\begin{aligned}
y_{22} &= 0.0, & y'_{22} &= -0.86473146,\\
y_{13} &= 0.97000436, & y'_{13} &= 0.466203685,\\
y_{23} &= -0.24308753, & y'_{23} &= 0.43236573.
\end{aligned}
$$

We solve the problem for $x \in [0, 20]$.

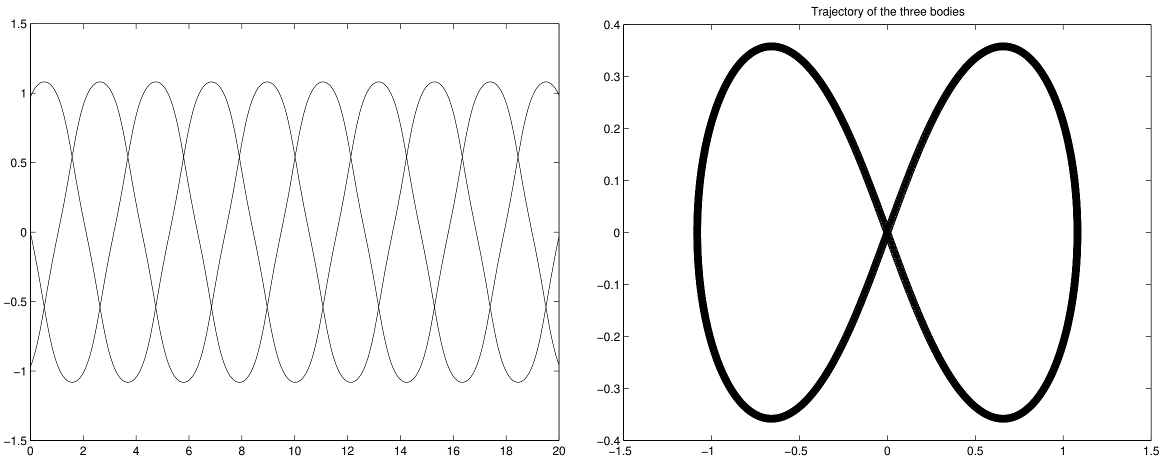
The second problem is the well-known Lorenz problem which arises in the study of dynamical systems and is known to have solutions which are potentially poorly conditioned,

$$
\begin{aligned}
y'_1 &= 10(y_2 - y_1),\\
y'_2 &= y_1(28 - y_3) - y_2,\\
y'_3 &= y_1 y_2 - \frac{8}{3} y_3,
\end{aligned}
$$

with

$$y_1(0) = 15,\ y_2(0) = 15,\ y_3(0) = 36, \ \text{and } x \in [0, 20].$$

Using ode45 to investigate solutions of these problems allows us to illustrate the ability of our tools to detect potential conditioning difficulties. Figures 1-2 present standard visualizations of numerical solutions to these problems determined by ode45.

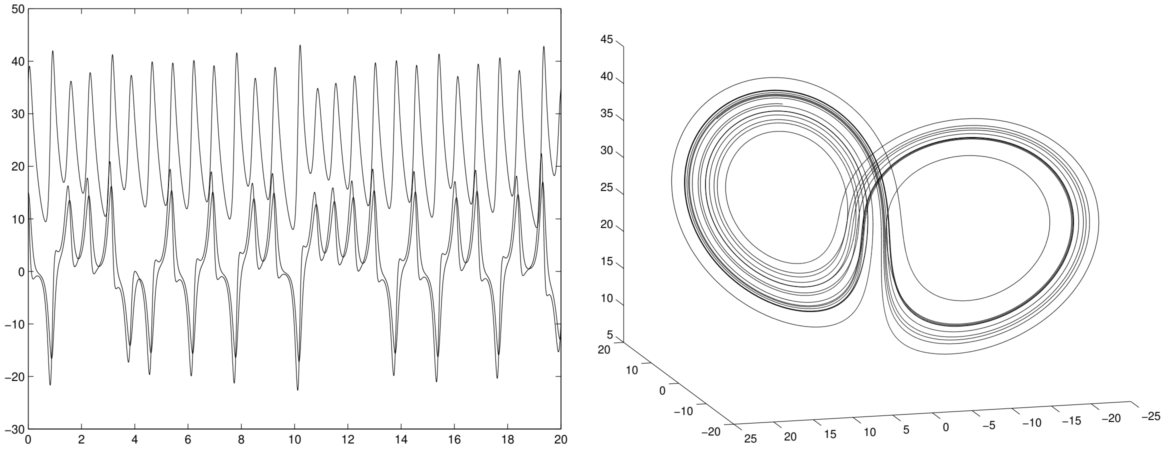


(a) Solution Plot

(b) Phase Portrait

Figure 1: The approximate solution produced by ode45 for the Three Body problem

Tables 1–2 and Figure 3 report statistics on the behaviour of the effectivity indices quantifying the verification tools we have implemented on these two test problems. As these tables indicate the tools are able to provide a consistent and reliable estimate of the conditioning of the mathematical model over a range of accuracy requests. Note that even on poorly conditioned problems, such as the Lorenz problem (where the condition number is exponential in the length of the integration interval), all three tools reflect this. While the effectivity index values associated with Check1 are



(a) Solution Plot (b) Phase Portrait

Figure 2: The approximate solution produced by ode45 for the Lorenz problem

TOL	CEST	MnEI	MxEI	CEST	MnEI	MxEI	CEST	MnEI	MxEI
	Check1			Check2			Check4		
10^{-3}	801	.67	1.01	778	.56	15.5	1078	.42	3.59
10^{-4}	1370	.98	1.01	1450	.81	5.43	1411	.39	1.53
10^{-6}	120	.99	1.02	188	.90	11.9	76	.62	1.22
10^{-8}	190	.98	1.03	247	.63	5.80	223	.21	2.75
10^{-9}	220	.99	1.02	220	.99	1.02	258	.28	2.09
10^{-10}	230	.13	1.00	218	.19	1.92	273	.05	1.32

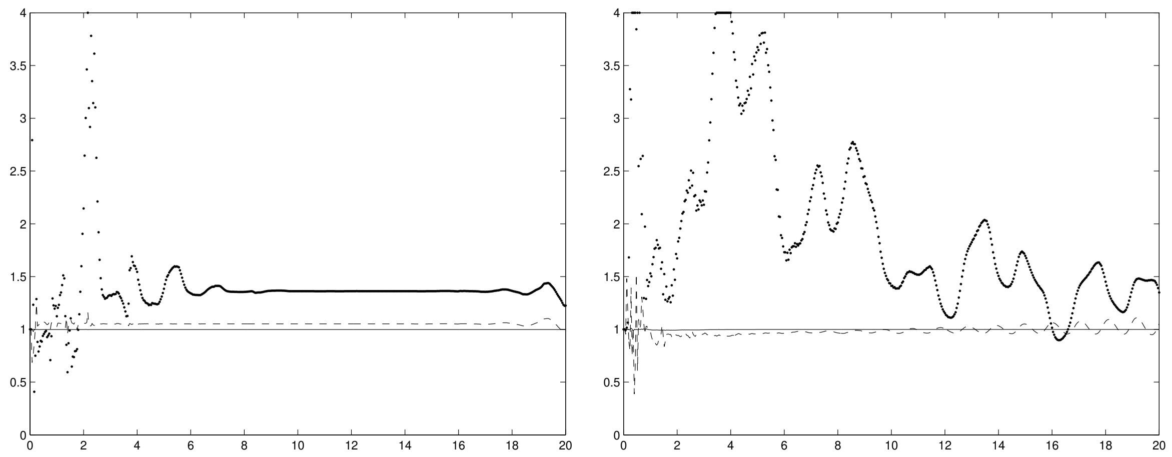
Table 1: Effectiveness Indices for the Three Body problem.

particularly good (all values very close to 1), the values are also quite reasonable for the other tools as well (the respective estimates of the norm of the maximum error is rarely out by a factor of 5).

From the reported results it is also clear that, when the errors are potentially underestimated

TOL	CEST	MnEI	MxEI	CEST	MnEI	MxEI	CEST	MnEI	MxEI
	Check1			Check2			Check4		
10^{-3}	$3.8 \cdot 10^4$	.20	2.22	$4.0 \cdot 10^4$	.06	6.21	$3.3 \cdot 10^4$	.18	6.21
10^{-4}	$4.0 \cdot 10^5$	.26	18.1	$3.7 \cdot 10^5$	.10	6.83	$3.6 \cdot 10^5$	.09	8.44
10^{-6}	$3.6 \cdot 10^7$	.92	1.12	$3.1 \cdot 10^7$	.30	14.3	$3.6 \cdot 10^7$	.28	3.77
10^{-8}	$5.1 \cdot 10^8$	.99	1.00	$2.8 \cdot 10^9$	.70	9.85	$9.1 \cdot 10^8$	.39	2.05
10^{-9}	$4.4 \cdot 10^8$	.98	1.00	$5.7 \cdot 10^8$	.41	4.33	$4.4 \cdot 10^8$	.69	1.24
10^{-10}	$4.3 \cdot 10^8$	.24	1.08	$9.4 \cdot 10^8$	.60	10.4	$4.5 \cdot 10^8$	.18	1.19

Table 2: Effectiveness Indices for the Lorenz Problem.



(a) Three Body problem at $TOL = 10^{-6}$ (b) Lorenz problem at $TOL = 10^{-8}$

Figure 3: Graphs of the Effectiveness Indices over the range of integration for the Verification tools. The solid line corresponds to Check1, dashed line to Check2 and dotted line to Check4.

(small values of $MnEI$), the reported values are usually large in magnitude (the corresponding relative error is estimated to be much greater than 100%) and this is a reliable signal that the approximate solution cannot be trusted. The results summarized in Figure 2 illustrate that the tools provide reliable and credible information about the approximate solution over the entire interval of integration.

These results indicate that verification tools, such as those we have developed, can provide inexpensive confirmation that a particular approximate solution can be trusted. It is also clear that, when the reported error is small enough for the approximation to be accurate to a few significant digits, then the associated estimate of the condition number is generally a good indication of the underlying numerical conditioning.

References

[1] M. Ainsworth and J. Tinsley Oden, A posteriori error estimation in finite element analysis, Wiley – Interscience, New York, 2000.

[2] I. Babuska and T. Strouboulis, The finite element method and its reliability, Oxford University Press, Oxford, 2001.

[3] W.H. Enright, Tools for the Verification of Approximate Solutions to Differential Equations, to appear?.

[4] W.H. Enright, K.R. Jackson, S.P. Nørsett and P.G. Thomsen, Interpolants for Runge-Kutta formulas, ACM Trans. Math. Soft., 12, 1986, pp.193-218.

[5] D.J. Higham and N.J. Higham, *Matlab Guide*, SIAM, 2000.

[6] The MathWorks, Inc., MATLAB 6.0, Natick MA, 2000.

[7] L.F. Shampine, I. Gladwell and S. Thompson, *Solving ODEs in in the MATLAB Problem Solving Environment*, To Appear, 2002.

[8] L.F. Shampine and M.W. Reichelt. The MATLAB ODE suite, SIAM J. Sci. Comp., 18, 1997, pp.1-22.

[9] P.E. Zadunaisky, On the estimation of errors propagated in the numerical integration of ordinary differential equations, Numer. Math., 27, 1976, pp. 21-39.