Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

ULRICH NOWAK, SUSANNA GEBAUER

A New Test Frame for Ordinary Differential Equation Solvers

Preprint SC 97-68 (December 1997)

A New Test Frame for Ordinary Differential Equation Solvers

Ulrich Nowak, Susanna Gebauer

10th February 1998

Abstract

Benchmarking of ODE methods has a long tradition. Several sets of test problems have been developed and new problems are still collected. So, a whole variety of problems can be used to check the efficiency of a method under investigation. In general, efficiency is measured by the amount of work which is necessary to get a reliable solution for a prescribed accuracy. In order to quantify the term "amount of work" usually not only the computing time is measured but also the number of calls of functional units. To quantify the term "quality of a numerical solution" usually the l_2–norm of the true error at the final point of the integration interval is used. In our contribution we first discuss some general aspects of benchmarking. Then we present a new test frame which allows to solve typical benchmark problems with some of the state of the art integrators within a unified framework. Finally we show some results of our benchmark tests. Part of the test frame can be used interactively in the World Wide Web.

1 Introduction

During the last thirty years or so there has been a great deal of interest in developing efficient methods and computer codes for the numerical solution of initial value problems (IVPs) in ordinary differential equations (ODEs)

$$y' = f(t, y) \ , \ \ y(t_0) = y_0 \in \mathbb{R}^n \ , \ \ t \in [t_0, t_F] \ . \tag{1}$$

A quite natural question, at least from an users point of view, is the following. Which method is the best one for my problem? Probably never there will be an ultimate answer. Not only because the answer is problem dependent but also because there will never be a consistent use (and understanding) of the term "best".

Despite these problems great efforts were made to develop a methodology for fair benchmarking. An early paper which strongly influenced the field of testing and benchmarking is [13]. It deals with non–stiff problems and methods and introduces the famous *DETEST* set of test problems. This test set consists of 25 problems subdivided into 5 classes. A continuation is the paper of [5] which deals with stiff problems and methods and introduces the *STIFF DETEST* set of test problems. A refined and extended version of these benchmarking techniques is presented in [6, 7].

Over the years new test classes and individual test problems were proposed, see e.g. [8, 16, 9, 10]. And the process of collecting new interesting and challenging test problem is still ongoing, cf. [15].

Our new development aims in presenting all these test problems, and some new ones, together with a whole bunch of methods within a unified test framework, which is easy to use and easy to access. Therefore we are implementing a graphical user interface which can be accessed via the World Wide Web using a standard internet browser. We should mention that the test frame is part of a larger project of the Scientific Software group of the ZIB, namely the development of an electronic laboratory for ordinary differential equations, ODELab.

2 Specifications

In the basic paper [13] is stated that three ingredients are needed for a fair comparison of methods. A set of problems to be solved, $P = \{p_1, \ldots, \}$. A set of methods to be compared, $M = \{m_1, \ldots, \}$, and a set of comparison criteria, $\{c_1(p, m), C_1(P, m), \ldots\}$. Assume that c, C are monotonic decreasing with increasing "goodness" of the method. Then one can say a method m_i is better than a method m_j, relative to problem class P and according to criterion C_l, if $C_l(P, m_i) < C_l(P, m_j)$.

We follow this basic approach and discuss advantages and drawbacks of some specifications which were used to compare methods throughout the years.

2.1 Problems

In [13] a particular problem is specified in terms of six items, namely
$p =< f, t_0, y_0, t_F, tol, h_{max} >$. In [5], where stiff IVPs are considered, this list is extended by the items $\partial f/\partial y$ and h_0.

The first four items are out of discussion as they appear in the usual mathematical problem description, cf. (1). To efficiently solve stiff problems information on the Jacobian is needed and this indicates the $\partial f/\partial y$ term. The matrix must not necessarily be explicitly available, as most implementations of stiff methods are equipped with an internal numerical differentiation device. Somehow suspicious are the terms h_{max} (maximum stepsize) and h_0 (initial stepsize). For different reasons we drop them both.

The really hard item to specify is *tol*, the "prescribed tolerance". The first question which arises is, where do we want to get (check) the solution. From a mathematical point of view we would like to have the L_2–norm of the difference between the numerical and the true solution below the bound *tol*. As not all methods have a natural continuous solution representation, the typical requirement is that the l_2–norm of the difference at all integration points should be below *tol*. In order to equilibrate different scales (e.g. units) in different solution components and to eliminate the influence of the dimension n, the standard l_2–norm is replaced by the so–called weighted root mean square norm

$$(\sum_{j=1}^{n}(\frac{y_j^{num}(t_i) - y_j^{true}(t_i)}{y_j^w(y_i)})^2)^{1/2} \stackrel{!}{\leq} tol \ , \tag{2}$$

where y_j are the individual solution components and y_j^w are weighting values. With $y_j^w(t_i) = |y(t_i)|$, (2) is then a pure relative error requirement. For compo-

nents with $y_j(t) \approx 0$ this simple setting causes trouble and is usually replaced by

$$y_j^w(t_i) := \max\{|y_j(t_i)|, y_j^{thresh}\} \ . \tag{3}$$

The threshold values y_j^{thresh} must be part of our problem specification.

In the literature there was a long discussion on that topic. In most of the problems of the testsets $DETEST$ and $STIFF\ DETEST$ all the components are well scaled. So, for a long time an absolute error criterion was used to check the results in comparison tests. At best, a criterion like (2) but with $y_j^{thresh} = 1$ in (3) was used. This type of weighting was also very popular in the first generation of ODE software. It is not surprising, that applied to more realistic problems, these methods often "failed" or gave unsatisfying results.

In the test frames NSTTST and STDTST [7] all error tests use an absolute error criterion. To improve this situation one may (automatically) solve the transformed problem

$$z' = D^{-1} f(t, Dz) \tag{4}$$

where

$$z = D^{-1} y \ , \quad D = diag(y_j^{max}, \ldots, y_n^{max}) \tag{5}$$

and y_j^{max} is the maximum absolute value of component j over $[t_0, t_F]$. This modification helps for some problems but is not sufficient for the general case.

The importance of prescribing appropriate threshold values is illustrated in Fig. 1. Here we solve the well known Oregonator test problem, see e.g. [10] for a description, over the interval $[0., 700.]$. As an integrator we use the extrapolated linearly implicit Euler discretization in the implementation due to [10], but the effects are reproduceable with other stiff integrators as well. The prescribed relative tolerance (in the sense of (2,3)) is $tol = 10^{-4}$. We compare the results for 3 sets of threshold values: $y_j^{thresh} = 10^5$. This choice mimics the transformation (4,eq:trafo2). $y_j^{thresh} = 10^2$. This is, somehow, a next trial value. Finally, $y_j^{thresh} = 10^{-3}$. With the first choice (dotted lines) numerical solution and true solution have nothing in common. The second choice (dashed lines) allows the simulation of one oscillation, but with a wrong period. With the last values (solid lines) the problem can be integrated over more than one period.

In our test frame we have incorporated 23 problems. This includes – except for the discretized PDE problems – all the examples from [15, 9, 10].

Within our WWW–interface we allow the users to set up new examples. However, the Oregonator problem shows that some skill may be needed.

2.2 Methods

Concerning the term method there was, very early, a general agreement that a method should incorporate the following essential parts. A formula for computing the next approximaton, i.e. a discretization. A formula for estimating the error of the step and a strategy for accepting/rejecting the current step and choosing a new stepsize (and order).

In our test frame we have included some "oldies" as well as more recent methods. Currently the following codes are available. The multistep methods LSODE [11], LSODA [12, 18] and VODE [1] and the extrapolation methods EULSIM [4] and DIFEX1 [4]. The more recent Runge-Kutta and extrapolation

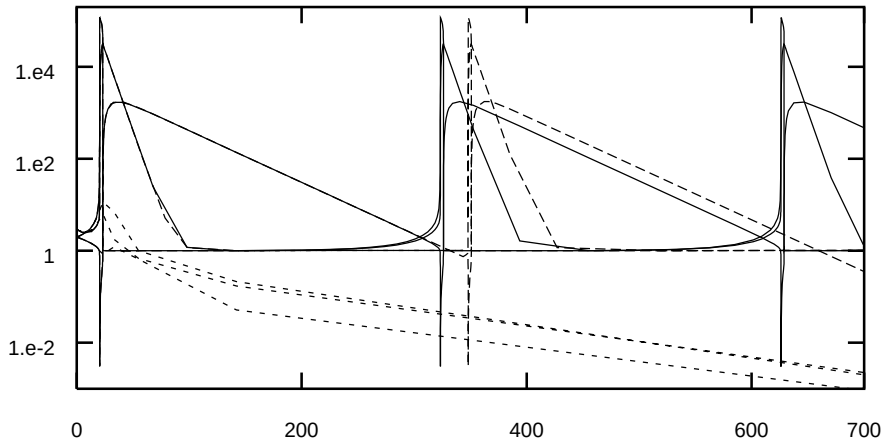

Figure 1: Three solution approximations for the Oregonator problem

codes from [9, 10], i.e. the explicit methods DOPRI5, DOP853 and ODEX and the stiff methods SDIRK4, RODAS, SEULEX, RADAU5 and RADAUP. Finally, there is access to MEBDF [2], DASSL [17] and GAM [14]. We have incorporated DAE–methods as we think it is interesting to study their performance on pure ODE problems also. In any case, we continue to incorporate further codes.

In the general test frame we use the default settings of the codes for internal numerical parameters which may be modified by the user. Most of the codes use a *Rtol*/*Atol* pair of values (vectors) to control the error according to

$$(\sum_{j=1}^{n}(\frac{\Delta y_j(t_i)}{Atol_i + Rtol_i|y_j(t_i)|})^2)^{1/2} \stackrel{!}{\leq} 1 \ . \tag{6}$$

By setting $Atol_j = y_j^{thresh} \times tol$ the accuracy requirement (2,3) and the internal accuracy check coincide very well.

2.3 Comparison Criteria

The comparison criteria should be chosen such that both the efficiency and the reliability of a method are reflected. The costs are usually measured by the total CPU time needed to solve a problem. A more detailed understanding allows the counting of the number of calls of the main functional units of a method, i.e. NFCN – evaluations of the right hand side of (1), NJAC – evaluations of the Jacobian matrix, NDEC – decomposition of matrices NSOL – linear system solves. Typically the number of overall integration steps is also monitored. A distinction of successful and rejected steps can help to judge the performance of the method. Note that counting calls to functional units has the advantage to describe efficiency without being effected by the special (fast/slow) implementation of the units.

The hard part is the attempt to check for reliability. One may distinguish two philosophies. First, one may check the true error (in a meaningful norm). Alternatively, one may check what the solvers control, i.e. check if the approximation error of one step is below the prescribed tolerance. This approach, counting deceptions, is advocated in [13, 5], but was not widely accepted. In

[6, 7] both approaches are offered, including sophisticated statistical devices, e.g. an estimation of the expected work for a specific prescribed tolerance. Although used, e.g. in [3], the drawback of using at best an absolute l_2 or l_∞ norm, hindered a general acceptance.

In our test frame we follow the current tendency in benchmarking and offer only some simple tools to display error over time and work over achieved precision. As the type of the norm may strongly influence the results we offer several, always weighted, norms for selection, e.g. local l_2 and l_∞ as well as global (in time) l_2, l_∞ and a discretized L_2 norm.

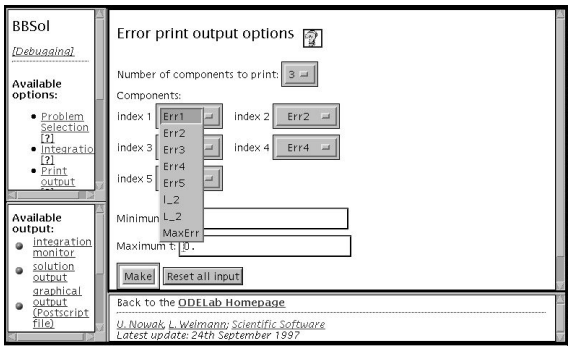

Figure 2: Web GUI form for error output selection

3 Example

For the Oregonator problem the errors at the integration points are computed and displayed in printable form by filling out the WWW–form presented in Fig. 2. Part of the header and trailer of the generated data file are:

```
Error Data in Printable Form

 Problem    =  Oregon
 Integrator =  SEULEX

 ......

 Maximum Maximum (L-infinity) Error:      1.406E+00
  (at t =  6.28879E+02 for j =   1)
 l2-l2 Error:                             7.591E-02
 l2 of integrated Error**2 (=L2-Error):   9.572E-03
```

Note that for this specific example the different norms vary over two orders of magnitude.

Comparison runs over a range of tolerances can be automatically performed also. The results are then arranged in a work precision diagram. A typical example is given in Fig. 3. Herein, the results for the required tolerances $Rtol = 10^{-2}, 10^{-3}, \ldots, 10^{-8}$ are depicted.

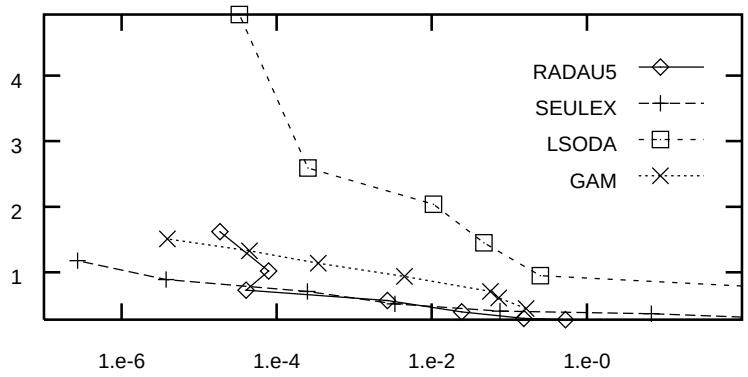

Figure 3: CPU time over achieved precision for the Oregonator problem

4 Conclusion

In the field of testing ODEs there was, very early, a general acceptance of some essential points. A problem should be reasonably realistic and should not exhibit special features like occurence of singularities. A whole set of test problems should be used which covers a broad spectrum of problem types. A solution method should consist of a discretization and an associated control strategy. Benchmarking with optimization by hand was not viewed as being helpful.

The comparison criteria used and advocated were quite different. Nevertheless, the comparison of methods accompanied the method development in a fruitful way. Users of ODE software had and have the chance to inform themselves on merits and drawbacks of methods in a comparable way.

A decade after invention of the public test domain software NSTTST and STDTST we try to set up a new test frame where we concentrate on the implementation of a large set of problems, the embedding of many methods and the proper computation of some widely accepted indicators for efficiency and reliability. To minimize the amount of human efforts using the frame we offer a WWW access. Try `http://galois.zib.de:8001/public/odelab/` to see the first experimental version.

References

[1] P. N. Brown, G. D. Byrne, A. C. Hindmarsh: VODE: A Variable Coefficient ODE Solver. SIAM J. Sci. Stat. Comput. 10, 1039–1051, (1989)

[2] J. R. Cash, S. Considine: An MEBDF Code for Stiff Initial Value Problems. ACM Trans. Math. Softw. 18, 2, 142–155, (1992)

[3] J. R. Cash, S. Semnani: A Modified Adams Method for NonStiff and Mildly Stiff Initial Value Problems. ACM Trans. Math. Softw. 19, 1, 63–80, (1993)

[4] P. Deuflhard: Recent Progress in Extrapolation Methods for ODEs. SIAM Review 27, 505–535, (1985)

[5] W. H. Enright, T. E. Hull, B. Lindberg: Comparing Numerical Methods for Stiff Systems of ODEs. BIT 15, 10–48, (1975)

[6] W. H. Enright, J. D. Pryce: Two FORTRAN Packages for Assessing Initial Value Methods. ACM Trans. Math. Softw. 13, 1, 1–27, (1987)

[7] W. H. Enright, J. D. Pryce: ALGORITHM 648, NSDTST and STDTST: Routines for Assessing the Performance of IV Solvers. ACM Trans. Math. Softw. 13, 1, 28–34, (1987)

[8] B.A. Gottwald, G. Wanner: A Reliable Rosenbrock Integrator for Stiff Differential Equations. Computing 26, 2, (1981), 355–360

[9] E. Hairer, S. P. Nørset, G. Wanner: Solving Ordinary Differential Equations I: Nonstiff Problems. Springer Verlag, (1987)

[10] E. Hairer, G. Wanner: Solving Ordinary Differential Equations II: Stiff and Differential–algebraic Problems. Second Revised Edition, Springer Verlag, (1996)

[11] A. C. Hindmarsh: LSODE and LSODI, Two New Initial Value Ordinary Differential Equation Solvers. ACM SIGNUM Newsletter 15, 10–11, (1980)

[12] A. C. Hindmarsh: ODEPACK, a Systematized Collection of ODE Solvers. In Scientific Computing, R.S. Stepelman et al. (eds.) North–Holland, Amsterdam, 55–64, (1983)

[13] T. E. Hull, W. H. Enright, B. M. Felen, A. E. Sedgewick: Comparing Numerical Methods for Ordinary Differential Equations. SIAM J. Numer. Anal.9, 4, 603–637, (1972)

[14] F. Iavernaro, F. Mazzia: Solving Ordinary Differential Equations by Generalized Adams Methods: properties and implementation techniques. Proc. of NumDiff8, Appl. Numer. Math. (to appear)

[15] Parallel IVP Solvers Group of CWI: Test Set for IVP Solvers. URL: http://www.cwi.nl/cwi/projects/IVPtestset.shtm

[16] P. Kaps, S. W. H. Poon, T.D. Bui: Rosenbrock Methods for Stiff ODEs: A Comparison of Richardsen Extrapolation and Embedding Techniques. Computing 34, 1, (1985), 17–40

[17] L. R. Petzold: A Description of DASSL: a differential–algebraic system solver. Proc. IMACS World Congress, Montreal, Canada (1982)

[18] L. R. Petzold: Automatic Selection of Methods for Solving Stiff and Nonstiff Systems of Ordinary Differential Equations. SIAM J. Sci. Stat. Comput. 4, 136–148, (1983)

[19] L.F. Shampine: Evaluation of a Test Set for Stiff ODE Solvers. ACM Trans. Math. Softw. 7, 409–420, (1981)