Numerical experiments with some explicit pseudo two-step RK methods on a shared memory computer *

Cong, N.H., Podhaisky, H. and Weiner, R.

Abstract

This paper investigates the performance of two explicit pseudo two-step Runge-Kutta methods of order 5 and 8 for first-order nonstiff ODEs on a parallel shared memory computer. For expensive right hand sides the parallel implementation gives a speedup of 3–4 with respect to the sequential one. Furthermore we compare the codes with the two efficient nonstiff codes DOPRI5 and DOP853. For problems, where the stepsize is determined by accuracy rather than by stability our codes are shown to be more efficient.

Key words: Runge-Kutta methods, pseudo two-step Runge-Kutta methods, parallelism.

1 Introduction

The arrival of parallel computers influences the development of methods for the numerical solution of a nonstiff initial value problem (IVP) for systems of first-order ordinary differential equations (ODEs)

$$y'(t) = f(t, y(t)), \qquad y(t_0) = y_0, \qquad y, f \in \mathbb{R}^d. \tag{1.1}$$

Although there exist in the literature very efficient sequential numerical methods for solving this problem like multistep methods or explicit Runge-Kutta methods (cf. e.g., Butcher [6], Hairer et al. [12]), a number of parallel explicit methods (cf. e.g., Burrage [1, 2], Burrage and Suhartanto [5], Chu and Hamilton [7], Cong [8], Cong and Mitsui [11], van der Houwen and Cong [14], van der Houwen and Sommeijer [15, 16], Jackson and Nørsett [18], Lie [19], Nørsett and Simonsen [21], etc.) has been proposed for exploiting new computing facilities.

In a recent work of Cong [9], a general class of explicit pseudo two-step RK methods (EPTRK methods) for solving problems of the form (1.1) has been considered. This class of EPTRK methods is suitable for use on parallel computers and can be easily equipped with embedded and continuous formulas for an implementation with stepsize control and dense output (cf.

*This work was partly supported by DAAD, N.R.P.F.S. and QG-96-02

Cong [10]). In terms of comparing the number of f-evaluations for a given accuracy, the EPTRK methods have been shown to be much more efficient than the most efficient sequential and parallel methods currently available for solving nonstiff IVPs (cf. Cong [9, 10]).

Most numerical comparisons of parallel and sequential codes considered so far are done by means of the number of f-evaluations for a given accuracy on a sequential computer, (see e.g., [1], [5], [8]). The communication time between processors in these comparisons is ignored. In comparisons of different codes running on parallel machines, the results of the parallel codes are often disappointing. In the present work we investigate how the performance of the EPTRK methods is reflected in a real implementation on a parallel computer. This investigation is performed by means of comparing some EPTRK methods with the codes DOPRI5 and DOP853 of Hairer et al. [12] which are among the most efficient sequential nonstiff integrators for first-order ODEs. Although the class of EPTRK methods contains methods of arbitrary high order, we shall compare DOPRI5 and DOP853 with two EPTRK methods of orders 5 and 8 running on the same shared memory parallel computer to have a "fair" comparison.

The choice of an implementation on a shared memory computer is due to the fact that such a computer can consist of several processors sharing a common memory with fast data access requiring less communication times which is suited to the features of the EPTRK methods. In addition, there are the advantages of compilers which attempt to parallelize codes automatically by reordering loops and sophisticated scientific libraries (cf. e.g., Burrage and Pohl [4]).

The test problems used in our numerical comparisons differ with respect to computing costs of f-evaluations and stability requirements.

2 Explicit pseudo two-step RK methods

We consider explicit pseudo two-step RK methods (EPTRK methods) which have been recently introduced and investigated by Cong [9, 10]. For an implementation with stepsize control we shall consider variable stepsize embedded EPTRK methods.

Variable stepsize EPTRK methods

A general s-stage variable stepsize explicit pseudo two-step RK method based on an s-dimensional collocation vector $c = (c_1, \ldots, c_s)^T$ with distinct abscissas c_i has the form (cf. Cong [10])

$$Y_n = e \otimes y_n + h_n (A_n \otimes I) F(t_{n-1}e + h_{n-1}c, Y_{n-1}), \tag{2.1a}$$

$$y_{n+1} = y_n + h_n (b^T \otimes I) F(t_n e + h_n c, Y_n), \tag{2.1b}$$

where, $h_n = t_{n+1} - t_n$. The variable $s \times s$ matrix A_n and s-dimensional vector b of the method parameters are derived by the order conditions in Cong [9, 10] and given by

$$(2.2a)\qquad A_n = P\ \mathrm{diag}\Big\{1, \frac{h_n}{h_{n-1}}, \ldots, \Big(\frac{h_n}{h_{n-1}}\Big)^{s-1}\Big\} Q^{-1},$$

$$(2.2b)\qquad P = \Big(p_{ij}\Big) = \Big(\frac{c_i^j}{j}\Big), \qquad Q = \Big(q_{ij}\Big) = \Big((c_i-1)^{j-1}\Big),$$

$$b^T = g^T R^{-1}, \qquad g = \Big(g_i\Big) = \Big(\frac{1}{i}\Big), \qquad R = \Big(r_{ij}\Big) = \Big(c_i^{j-1}\Big),$$

$$i = 1, \ldots, s, \qquad j = 1, \ldots, s.$$

This EPTRK method is conveniently specified by the tableau

$$\begin{array}{c|c|c} A_n & c & O \\ \hline & y_{n+1} & b^T \end{array}$$

If the stepsize ratio h_n/h_{n-1} is bounded from above (i.e., $h_n/h_{n-1} \le \Omega$) and the function f is Lipschitz continuous, then the method (2.1) is of order p and stage order q at least equal s for any collocation vector c. It has the highest order $p = s+1$ if the vector c satisfies an orthogonality relation (cf. Cong [10, Theorem 2.1]).

At each step $2s$ f-evaluations of the components $f(t_{n-1}+c_ih_{n-1}, Y_{n-1,i})$ and $f(t_n + c_ih_n, Y_{n,i})$, $i = 1, \ldots, s$ of the big vectors $F(t_{n-1}e + h_{n-1}c, Y_{n-1})$ and $F(t_ne + h_nc, Y_n)$, respectively, are used in the method. However, s f-evaluations of $f(t_{n-1} + c_ih_{n-1}, Y_{n-1,i})$, $i = 1, \ldots, s$ are already available from the preceding step. Hence, we need to compute only s f-evaluations of $f(t_n+c_ih_n, Y_{n,i})$, $i = 1, \ldots, s$, what can be done in parallel. Consequently, on a parallel implementation using s processors, just one effective sequential f-evaluation is required per step. In this way parallelisation in an EPTRK method is achieved by sharing the f-evaluations of the s components of the big vector $F(t_ne + h_nc, Y_n)$ over the available processors.
An additional computational effort consists of a recalculation of the parameter matrix A_n defined by (2.2a) when the stepsize is changed. In a parallel implementation, this recalculation can also be spread over a number of processors.

Embedded formulas

For estimating the local error used in the stepsize selection for an implementation of EPTRK methods with stepsize control, we shall apply embedding techniques. In order to equip the pth-order EPTRK method (2.1) with an embedded formula of order $\widehat{p}$, we consider a second $\widehat{p}$th-order EPTRK method based on collocation vector $\widetilde{c} = (\widetilde{c}_1, \ldots, \widetilde{c}_{\widetilde{s}})^T$ of the form (cf. Cong [10])

$$(2.3a)\qquad \widetilde{Y}_n = \widetilde{e} \otimes \widetilde{y}_n + h_n(\widetilde{A}_n \otimes I)F(t_{n-1}\widetilde{e} + h_{n-1}\widetilde{c}, \widetilde{Y}_{n-1}),$$

$$(2.3b)\qquad \widetilde{y}_{n+1} = \widetilde{y}_n + h_n(\widetilde{b}^T \otimes I)F(t_n\widetilde{e} + h_n\widetilde{c}, \widetilde{Y}_n),$$

where, $\widehat{p} < p$, the vector $\widetilde{c}$ is a subvector of the vector c, i.e., $\{\widetilde{c}_1, \ldots, \widetilde{c}_{\widetilde{s}}\} \subset \{c_1, \ldots, c_s\}$. Here $\widetilde{b}$ and $\widetilde{A}_n$ are defined by (2.2). By introducing a new vector $\widehat{b} = (\widehat{b}_1, \ldots, \widehat{b}_s)^T$ which is defined according to

$$(2.4)\qquad \text{if} \quad c_i = \widetilde{c}_j, \quad \text{then} \quad \widehat{b}_i = \widetilde{b}_j, \quad j = 1, \ldots, \widetilde{s},$$

$$\text{else} \quad \widehat{b}_i = 0, \quad i = 1, \ldots, s,$$

we obtain an embedded formula without additional f-evaluations given by

$$\widehat{y}_{n+1} = y_n + h_n(\widehat{b}^T \otimes I)F(t_n e + h_n c, Y_n). \tag{2.5}$$

If the function f is Lipschitz continuous, then the numerical approximations at t_{n+1} defined by (2.1b) and by (2.5) give a local error estimate

$$\|y_{n+1} - \widehat{y}_{n+1}\| = O(h_n^{\widehat{p}+1}), \tag{2.6}$$

(see Cong [10, Theorem 3.1]). Thus, we have the embedded EPTRK method suited to an implementation with stepsize strategy given by (2.1a), (2.1b) and (2.5) which can be specified by the tableau

$$\begin{array}{c|c|c} A_n & c & O \\ \hline & y_{n+1} & b^T \\ & \widehat{y}_{n+1} & \widehat{b}^T \end{array}$$

The "cheap" local error estimate is then defined by (2.6). By this approach of constructing embedded EPTRK methods, there exist several embedded formulas for an EPTRK method.

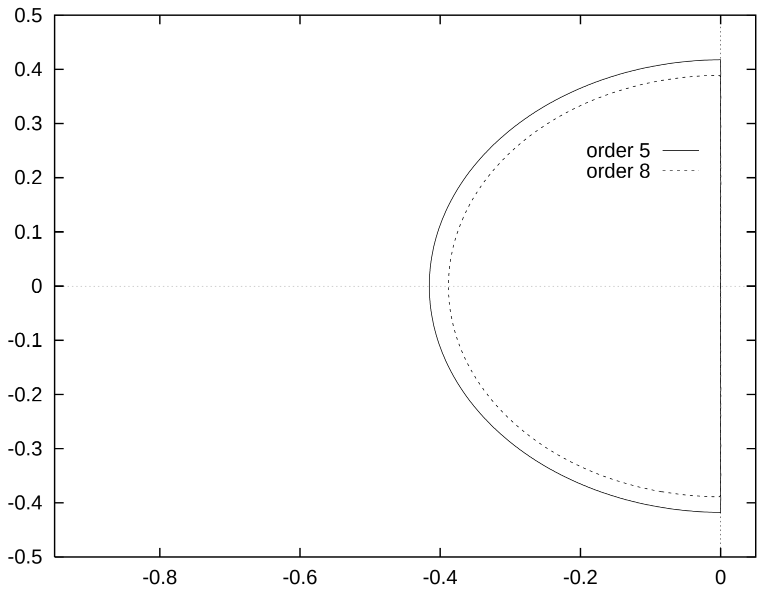

Figure 2.1: Stability regions of two EPTRK methods

Stability properties

Stability of (constant stepsize) EPTRK methods was investigated on the basis of the model test equation $y'(t) = \lambda y(t)$, where λ runs through the eigenvalues of the Jacobian matrix $\partial f/\partial y$. It is characterized by the spectral radius $\rho(M(z))$ of the $(s+1)\times(s+1)$ *amplification matrix* $M(z)$ given by (cf. Cong [9, Section 2.2])

$$M(z) = \begin{pmatrix} zA & e \\ z^2 b^T A & 1+z \end{pmatrix}. \tag{2.7a}$$

The stability region S_{stab} of an EPTRK method is defined as

$$S_{stab} := \{z : \rho(M(z)) \leq 1\}. \tag{2.7b}$$

From (2.7), it can be easily seen that zero-stability of EPTRK methods is independent of the method parameters so that the variable stepsize EPTRK method (2.1) is always stable.

Figure 2.1 shows the stability regions of two specified methods of order 5 and 8 which are used in our numerical tests (cf. section 3).

We observe that the stability regions of these two EPTRK methods are much smaller than those of the corresponding Dopri-methods. However, the scaled stability regions are comparable. Due to the additional communication times in a parallel implementation we cannot expect a speedup of our codes compared with DOPRI5 and DOP853 if the right hand sides are inexpensive to compute and if the stepsize is limited by stability requirements.

On the other hand we obtain a good speedup if

- The stepsize is limited by accuracy requirements rather than by stability (e.g. for stringent tolerances), and
- The right hand sides are expensive to compute.

This is shown in the following section.

3 Numerical experiments

Specification of the methods

For numerical comparisons we choose the codes DOPRI5 and DOP853 in the version of 24.3.93 and 20.9.93 respectively, which are based on the embedded explicit RK methods of Dormand and Prince (cf. Hairer et al. [12]). DOPRI5 and DOP853 use the 5(4) pair and the "triple" 8(5,3), respectively. DOP853 is the new version of DOPRI8 with a "stretched" error estimator (see Hairer et al. [12, p. 254]). These two codes belong to the most efficient currently existing sequential codes for nonstiff first-order ODE problems.

For EPTRK methods, we confine our considerations to two methods of corresponding orders 5 and 8. Although the class of EPTRK methods contains methods of arbitrarily high order (cf. Cong [9, 10]), the choice of two methods of orders 5 and 8 is simply motivated by comparing the methods of the same orders. In addition, it is also due to the fact that the maximal number of processors of the used parallel machine in shared memory mode is only 8 for our local configuration. The EPTRK codes with an example problem can be obtained from http://www.mathematik.uni-halle.de/institute/numerik/software.

The fifth-order EPTRK method is based on the collocation vector

$$c^5 = (0.089, 0.409, 0.788, 1.000, 1.409)^T, \tag{3.1a}$$

with the third-order embedded formula based on

$$(3.1b) \qquad \tilde{c}^3 = (0.788, 1.000, 1.409)^T.$$

This embedded pair of order 5(3) gives an $\mathcal{O}(h^4)$-estimate of the local truncation error lte. The eighth-order EPTRK method is based on the collocation vector

$$(3.2a) \qquad c^8 = (0.057, 0.277, 0.584, 0.860, 1.000, 1.277, 1.584, 1.860)^T,$$

with the sixth-order embedded formula using

$$(3.2b) \qquad \tilde{c}^6 = (0.584, 0.860, 1.000, 1.277, 1.584, 1.860)^T.$$

This embedded 8(6)-pair gives an $\mathcal{O}(h^7)$-estimate of the local truncation error. Note that the choice of the collocation vectors in (3.1a) and (3.2a) minimizes the principal error terms for some stage approximated values (cf. Cong [9, Theorem 2.4]). No special effort has been made to optimize the parameters of the methods. An optimal choice of the method parameters will be subject of later work. These two specified EPTRK methods were also used by Cong [10] with different embedded pairs in numerical tests on a sequential computer.

For the first step a starting procedure based on corrections until convergence of an appropriate s-stage collocation RK corrector is used. The stepsize strategy in our codes is similar to the one implemented by van der Houwen and Sommeijer [15] in PIRK methods which is also implemented in PIMRK methods by Burrage and Suhartanto [5] and in DOPRI5, DOP853 by Hairer and Wanner [12].

The new stepsize h_{n+1} is chosen as

$$(3.3) \qquad h_{n+1} = h_n \cdot \min\Big\{3, \max\Big\{0.3, 0.8 \cdot \|err\|^{-1/p^*}\Big\}\Big\},$$

where p^* is the local order of the embedded formula and

$$\|err\| = \sqrt{\frac{1}{d}\sum_{i=1}^{d}\Big(\frac{lte_i}{atol + rtol|y_{n,i}|}\Big)^2}$$

In our tests we used $rtol = atol = tol$. For $\|err\| > 1$ the step will be rejected. The constants 3 and 0.3 serve to keep the stepsize ratios h_{n+1}/h_n to be in the interval $\big[0.3,\ 3\big]$. These two new EPTRK codes of order 5 and 8 will be denoted by EPTRK5 and EPTRK8, respectively.

Specifications of the problems

We apply DOPRI5, DOP853, EPTRK5 and EPTRK8 to the following test problems of large dimension:

BRUSSELATOR: As a first test example we consider a two-dimensional reaction-diffusion equation based on the interaction between two chemical species. This equation is known as the diffusion Brusselator equation (see Hairer et al. [12], also Burrage [3] and Hochbruck et

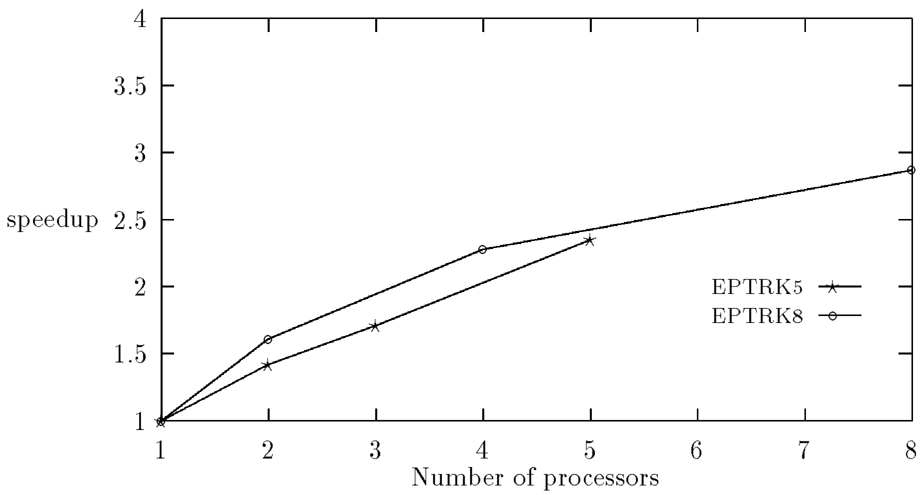

Figure 3.2: Speedup for the Brusselator.

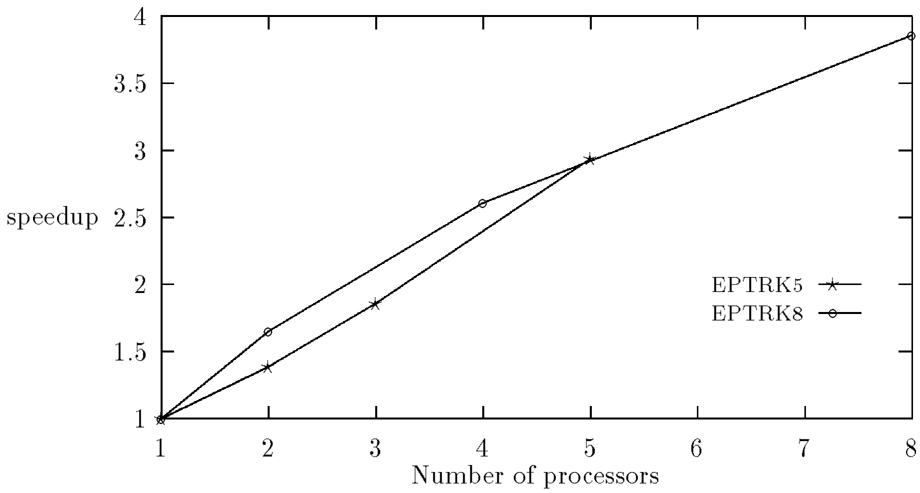

Figure 3.3: Speedup for DIFFU2 with $\beta = 1$.

al. [13]). It is defined on the unit square and takes the form

$$\begin{aligned} \frac{\partial u}{\partial t} &= B + u^2 v - (A+1)u + \alpha\Big(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\Big), \\ \frac{\partial v}{\partial t} &= Au - u^2 v + \alpha\Big(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\Big), \end{aligned} \tag{3.4}$$

with initial conditions

$$\begin{aligned} u(0,x,y) = 0.5 + y, \qquad v(0,x,y) = 1 + 5x, \\ A = 3, \qquad B = 1, \qquad \alpha = 2 \cdot 10^{-4}, \end{aligned}$$

and Neumann boundary conditions

$$\frac{\partial u}{\partial n} = 0, \quad \frac{\partial v}{\partial n} = 0 \quad \text{on} \quad \partial\Omega, \quad \Omega = [0,1] \times [0,1], \quad t \in [0,1]. \tag{3.5}$$

Here u and v denote chemical concentrations of reaction products, A and B are concentrations of input reagents which are taken to be constant and $\alpha = d/L^2$ where d is a diffusion coefficient and L is a reactor length. Second-order central differencing of (3.4) leads to a system of coupled nonlinear ODEs of dimension $2N^2$, where N is the number of grid points. In our tests we used $N = 100$, so that for the given value of α the resulting system of ODEs is nearly nonstiff.

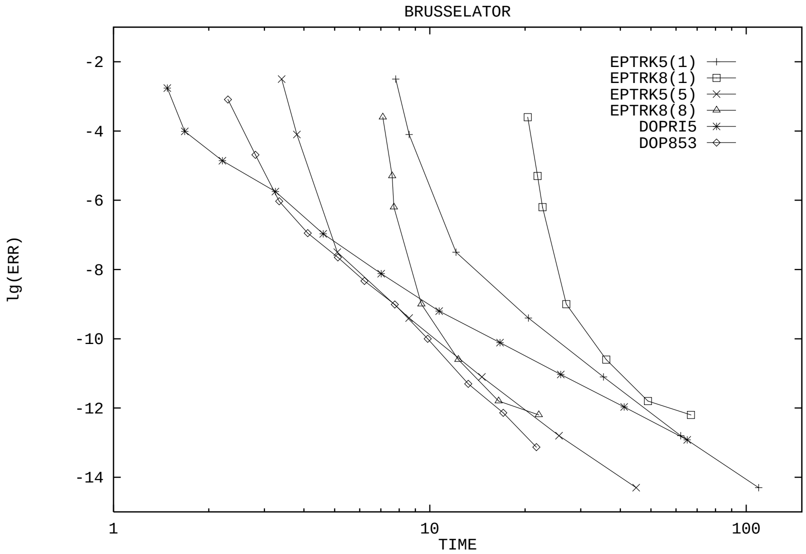

Figure 3.4: Results for the Brusselator

DIFFU2: The second test example is the two-dimensional diffusion equation depending on two parameters α and β (cf. Weiner et al. [22])

$$\begin{aligned} \frac{\partial u}{\partial t} =& \alpha \triangle u + f(t, x, y, \alpha, \beta), \\ \Omega =& [0,1]^2, \qquad t \in [0,1]. \end{aligned} \tag{3.6}$$

Here Dirichlet boundary conditions and the right-hand side function $f(t, x, y, \alpha, \beta)$ are determined by the exact solution given by

$$u(t, x, y) = \sin(\pi x) \sin(\pi y)(1 + 4xy \sin(\beta t)). \tag{3.7}$$

Fourth-order central differencing of (3.6) yields a system of linear ODEs of dimension N^2. In this example we choose $N = 69$, $\alpha = 0.001$ and use different values of β. The components of the right hand side of this problem are more expensive than those in the Brusselator. With increasing β the stepsize is more and more restricted by accuracy requirements.

The computations were performed on a HP-Convex S-Class-Server with 16 PA-8000 processors of 180 MHz and 3 GB global shared memory. The EPTRK methods were implemented in sequential and parallel modes, the number in brackets indicates the number of processors used.

At first we compare the results for the EPTRK methods in parallel implementation with different numbers of processors. Figure 3.2 and Figure 3.3 show the achieved speedup (for $tol = 10^{-5}$). For Diffu2 with its expensive right hand side, we observe a good speedup of about 4 for EPTRK8(8) and of about 3 for EPTRK5(5).

Next we compare our methods with DOPRI5 and DOP853 for the Brusselator and Diffu2 with different values of β. Figures 3.4 – 3.8 show the computing time versus the achieved accuracy ERR at the endpoint t_e of the integration interval,

$$ERR = \sqrt{\frac{1}{d}\sum_{i=1}^{d}\Big(\frac{u_i(t_e) - \tilde{u}_i(t_e)}{1 + |\tilde{u}_i(t_e)|}\Big)^2}.$$

The reference solution $\tilde{u}(t_e)$ was obtained by DOP853 with $tol = 10^{-13}$.

For crude tolerances at the Brusselator the stepsize in the EPTRK methods is restricted by stability, the codes DOPRI5 and DOP853 are much more efficient. For more stringent accuracy requirements the parallel version of the EPTRK codes becomes comparable, especially EPTRK5(5) is more efficient than DOPRI5 (see Figure 3.4).

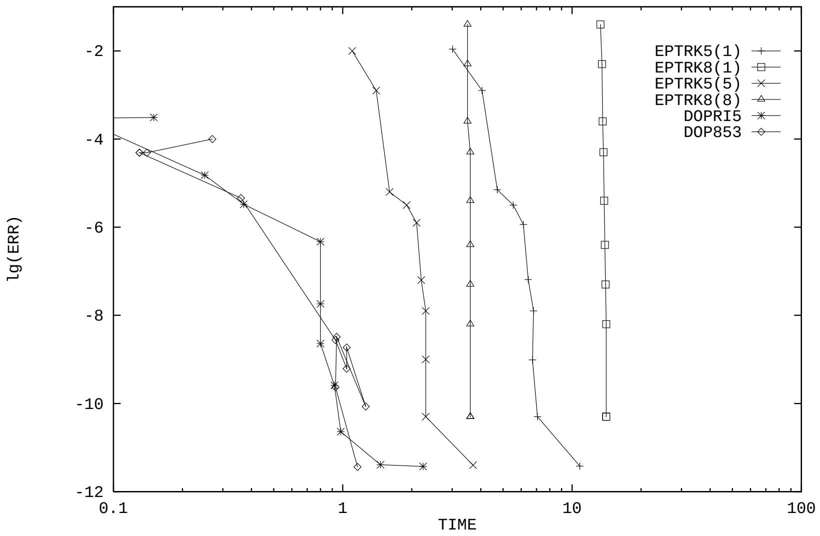

Figure 3.5: Results for DIFFU2 with $\beta = 1$

For small values of β in Diffu2 the stepsize in the EPTRK codes is restricted by stability, what can be observed by the nearly vertical lines in Figure 3.5. With increasing β EPTRK5(5) and EPTRK8(8) become more and more efficient in stringent accuracy ranges. And for $\beta = 1000$, where only accuracy limits the stepsize, they are superior to DOPRI5 and DOP853 in both parallel and sequential modes (cf. Figures 3.6– 3.8).

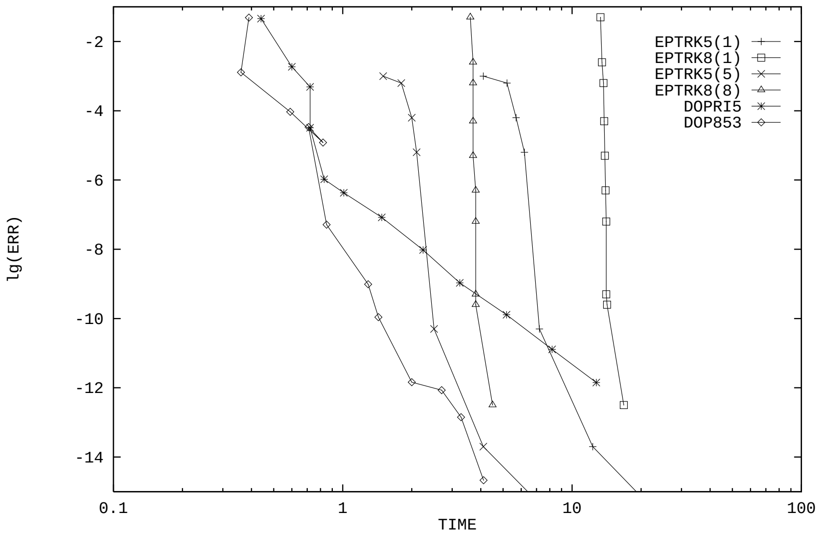

Figure 3.6: Results for DIFFU2 with $\beta = 10$

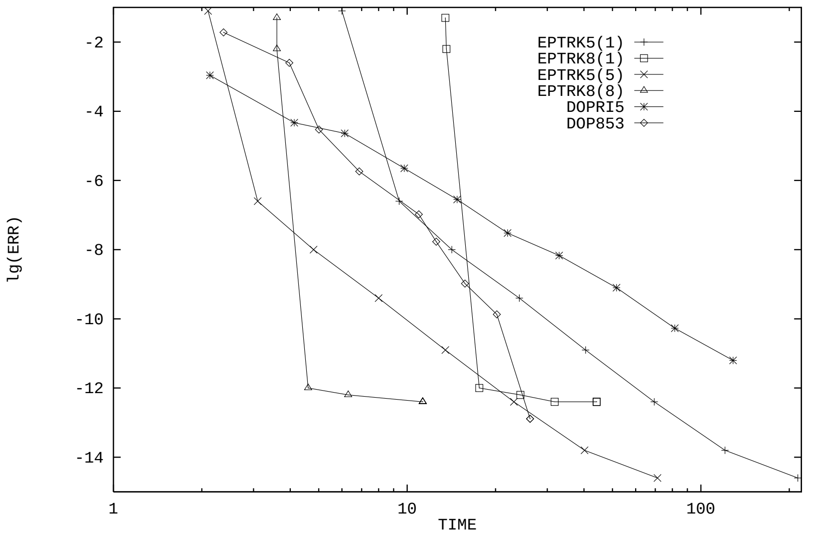

Figure 3.7: Results for DIFFU2 with $\beta = 100$

4 Conclusion

In this paper we have tested the efficency of a class of parallel explicit pseudo two-step RK methods by comparing two of the most efficient sequential codes, DOPRI5 and DOP853, with two new codes EPTRK5 and EPTRK8 of the same orders based on this new class of methods.

The results of our tests show that for inexpensive right hand sides and mildly stiff problems

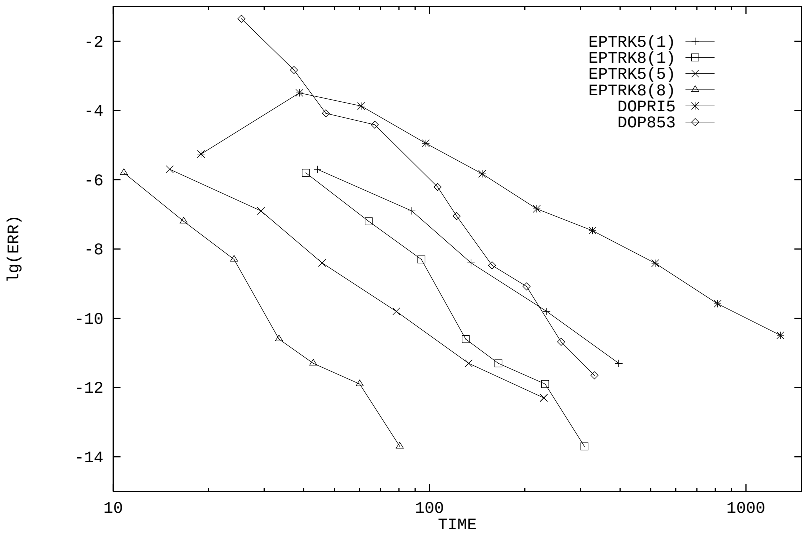

Figure 3.8: Results for DIFFU2 with $\beta = 1000$

DOPRI5 and DOP853 are superior. This is due to the communication costs and the small stability region of the EPTRK methods. On the other hand, for problems with expensive f-evaluations and when accuracy is more important than stability, the proposed parallel EPTRK methods are more efficient.

References

[1] K. Burrage, *Efficient block predictor-corrector methods with a small number of corrections*, J. Comput. Appl. Math. **45** (1993), 139-150.

[2] K. Burrage, *Parallel methods for initial value problems*, Appl. Numer. Math. **11** (1993), 5-25.

[3] K. Burrage, *Parallel and Sequential Methods for Ordinary Differential Equations*, (Clarendon Press, Oxford, 1995).

[4] K. Burrage and B. Pohl, *Implementing an ODE code on distributed memory computers*, Computers Math. Applic. **28** (1994), 235-252.

[5] K. Burrage and H. Suhartanto, *Parallel iterated methods based on multistep Runge-Kutta methods of Radau type*, Advances in Computational Mathematics **7** (1997), 37-57.

[6] J.C. Butcher, *The Numerical Analysis of Ordinary Differential Equations, Runge-Kutta and General Linear Methods*, (Wiley, New York, 1987)

[7] M.T. Chu and H. Hamilton, *Parallel solution of ODE's by multi-block methods*, SIAM J. Sci. Statist. Comput. **8** (1987), 137-157.

[8] N.H. Cong, *Parallel iteration of symmetric Runge-Kutta methods for nonstiff initial-value problems*, J. Comput. Appl. Math. **51** (1994), 117-125.

[9] N.H. Cong, *Explicit pseudo two-step Runge-Kutta methods for parallel computers*, submitted for publication.

[10] N.H. Cong, *Continuous variable stepsize explicit pseudo two-step RK methods*, submitted for publication.

[11] N.H. Cong and T. Mitsui, *A class of explicit parallel two-step Runge-Kutta methods*, Japan J. Indust. Appl. Math. **14** (1997), 303-313.

[12] E. Hairer, S.P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations, I. Nonstiff Problems*, (2. ed., Springer-Verlag, Berlin, 1993).

[13] M. Hochbruck, C. Lubich and H. Selhofer, *Exponential integrators for large systems of differential equations*, Tech. Rept., Mathematisches Institut, Universität Tübingen, 1995.

[14] P.J. van der Houwen and N.H. Cong, *Parallel block predictor-corrector methods of Runge-Kutta type*, Appl. Numer. Math. **13** (1993), 109-123.

[15] P.J. van der Houwen and B.P. Sommeijer, *Parallel iteration of high-order Runge-Kutta methods with stepsize control*, J. Comput. Appl. Math. **29** (1990), 111-127.

[16] P.J. van der Houwen and B.P. Sommeijer, *Block Runge-Kutta methods on parallel computers*, Z. Angew. Math. Mech. **68** (1992), 3-127.

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

[18] K.R. Jackson and S.P. Nørsett, *Parallel Runge-Kutta methods*, Manuscript (1988).

[19] I. Lie, *Some aspects of parallel Runge-Kutta methods*, Report No. 3/87, Division Numerical Mathematics, University of Trondheim, Norway (1987).

[20] H. De Meyer, M. Van Daele and G. Vanden Berghe, *On the implementation of parallel iterated Runge-Kutta methods on a transputer network*, Appl. Numer. Math. **13** (1993), 155-163.

[21] S.P. Nørsett and H.H. Simonsen, *Aspects of parallel Runge-Kutta methods*, in: A. Bellen, C.W. Gear and E. Russo, eds., Numerical Methods for Ordinary Differential Equations, Proceedings L'Aquilla 1987, Lecture Notes in Mathematics **1386**, (Springer, Berlin 1989).

[22] R. Weiner, B.A. Schmitt and H. Podhaisky, *ROWMAP - a ROW-code with Krylov techniques for large stiff ODEs*, Appl. Numer. Math. **25** (1997), 1-16.

Nguyen Huu Cong
Faculty of Mathematics, Mechanics and Informatics
Hanoi University of Sciences
90 Nguyen Trai
Dong Da, Hanoi
Vietnam

Helmut Podhaisky, Rüdiger Weiner
FB Mathematik und Informatik
Martin-Luther-Universität
Halle-Wittenberg
Postfach
D-06099 Halle/Saale
Germany
e-mail: rw@mail.mathematik.uni-halle.de