BIT Numerical Mathematics (2007) 47: 137–155
DOI: 10.1007/s10543-006-0095-7
Published online: 15 November 2006 – © Springer 2006

A MULTIRATE TIME STEPPING STRATEGY FOR STIFF ORDINARY DIFFERENTIAL EQUATIONS*

V. SAVCENCO[1,**] , W. HUNDSDORFER[1] and J. G. VERWER[1]

[1] *CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands.*
email: {savcenco,willem.hundsdorfer,jan.verwer}@cwi.nl

Abstract.

To solve ODE systems with different time scales which are localized over the components, multirate time stepping is examined. In this paper we introduce a self-adjusting multirate time stepping strategy, in which the step size for a particular component is determined by its own local temporal variation, instead of using a single step size for the whole system. We primarily consider implicit time stepping methods, suitable for stiff or mildly stiff ODEs. Numerical results with our multirate strategy are presented for several test problems. Comparisons with the corresponding single-rate schemes show that substantial gains in computational work and CPU times can be obtained.

AMS subject classification (2000): 65L05, 65L06, 65L50.

Key words: multirate time stepping, local time stepping, ordinary differential equations.

1 Introduction.

Standard single-rate time integration methods for ODEs work with time steps that are varying in time but constant over the components. There are, however, many problems of practical interest where the temporal variations have different time scales for different sets of the components. To exploit these local time scale variations, one needs multirate methods that use different, local time steps over the components.

In this paper we will consider a simple multirate approach for system of ODEs

$$w'(t) = F\,(t, w(t))\ , \quad w(0) = w_0\ , \tag{1.1}$$

with given initial value in $w_0 \in \mathbb{R}^m$. The approximations at the global time levels t_n will be denoted by w_n.

Our multirate approach is based on local temporal error estimation. Given a global time step $\Delta t_n = t_n - t_{n-1}$, we compute a first, tentative approximation

* Received May 1, 2006. Accepted July 19, 2006. Communicated by Christian Lubich.
** The work of the first author is supported by a Peterich Scholarship through the Netherlands Organisation for Scientific Research NWO.

at the new time level for all components. For those components for which the error estimator indicates that smaller steps are needed, the computation is redone with $\frac{1}{2}\Delta t_n$. This refinement stage may require values at the intermediate time level of components that are not refined. These values can be obtained by interpolation or by a 'dense output' formula. The refinement is continued with local steps $2^{-l}\Delta t_n$, until the error estimator is below a prescribed tolerance for all components. Schematically, with components horizontally and time vertically, the multirate time stepping is displayed in Figure 1.1. Small time steps will be used for the more active components and larger ones for the less active components.

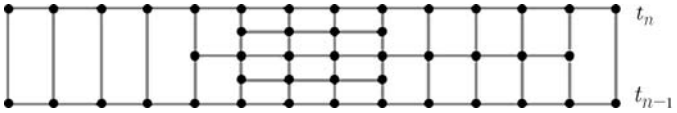

Figure 1.1: Multirate time stepping for a time slab $[t_{n-1}, t_n]$.

The intervals $[t_{n-1}, t_n]$ are called time slabs. After each completed time slab the solutions are synchronized. In our approach, these time slabs are automatically generated, similar as in the single-rate approach, but without imposing temporal accuracy constraints on all components of (1.1).

An important issue in our strategy will be to determine the size of the time slabs. These could be taken large with many levels of refinements, or small with few refinements. A decision will be made based on an estimate of the number of components at which the solution needs to be calculated, including the overhead due to repeated computations in refined sets.

The problems (1.1) in this paper are assumed to be stiff or mildly stiff. As basic integration method we will use a simple one-step Rosenbrock method. The presented strategy can be used with other methods as well, but for multistep methods additional interpolations of past values will be required in the refinement steps.

The paper is organized as follows. In Section 2 we will briefly discuss related work on multirate schemes and introduce the Rosenbrock method that will be used as our basic numerical integration method. In Section 3 the multirate time stepping is described in detail, together with the time slab strategies. The performance of the schemes is discussed in Section 4 by means of several numerical experiments. Finally, Section 5 contains the conclusions and an outlook on further work.

2 Background material and preliminaries.

2.1 Related work.

The first descriptions of automatic multirate schemes were given by Gear and Wells [6] for linear multistep methods. As noted before, with multistep methods interpolations of past values will be needed in general in the temporal refinement stages.

In Günther, Kværnø and Rentrop [7] a multirate scheme was introduced which is based on partitioned Runge–Kutta methods with coupling between active and latent components performed by interpolation and extrapolation of state variables. In particular, they introduced the notion of a compound step in which the macro-step (for latent components) and the first micro-step (for the active components) are computed simultaneously. The partitioning into slow (latent) and fast (active) components is done in advance before solving the problem, based on knowledge of the ODE system to be solved (in their applications these where electrical circuits). A related scheme, based on Rosenbrock or ROW methods, was studied by Bartel and Günther [2]; this will be further discussed in Section 4.3. Some stability results for simplified versions of these schemes, applied to systems of two linear equations, with one fast and one slow component, have been presented in Kværnø [11].

An algorithm based on finite elements was proposed by Logg [12, 13]. In a single-rate approach such schemes are computationally akin to fully implicit Runge–Kutta methods. In the multirate approach this leads to very complicated implicit relations, which are difficult to solve. Additional remarks on the strategy used for this scheme can be found in Section 3.3.

Finally we mention that multirate schemes for explicit methods and non-stiff problems have been examined by Engstler and Lubich [3, 4]. In the first paper extrapolation is used, and in their strategy the partitioning into different levels of slow to fast components is obtained automatically during the extrapolation process. This approach looks quite promising, but for stiff problems and implicit methods the necessary asymptotic expansions seem difficult to obtain.

2.2 The Rosenbrock ROS2 method.

Our multirate strategy is designed for one-step methods. In this paper we will use the two-stage second-order Rosenbrock ROS2 method [9] as our basic numerical integration method. To proceed from t_{n-1} to a new time level $t_n = t_{n-1} + \tau$, the method calculates

$$
\begin{aligned}
w_n &= w_{n-1} + \tfrac{3}{2}\bar{k}_1 + \tfrac{1}{2}\bar{k}_2\,, \\
(I - \gamma\tau J)\bar{k}_1 &= \tau F(t_{n-1}, w_{n-1}) + \gamma\tau^2 F_t(t_{n-1}, w_{n-1})\,, \\
(I - \gamma\tau J)\bar{k}_2 &= \tau F(t_n, w_{n-1} + \bar{k}_1) - \gamma\tau^2 F_t(t_{n-1}, w_{n-1}) - 2\bar{k}_1\,,
\end{aligned}
\tag{2.1}
$$

where $J \approx F_w(t_{n-1}, w_{n-1})$. The method is linearly implicit: to compute the internal vectors $\bar{k}_1$ and $\bar{k}_2$, a system of linear algebraic equations is to be solved. Method (2.1) is of order two for any choice of the parameter γ and for any choice of the matrix J. Furthermore, the method is A-stable for $\gamma \geq \frac{1}{4}$ and it is L-stable if $\gamma = 1 \pm \frac{1}{2}\sqrt{2}$. In this paper we use L-stability with $\gamma = 1 - \frac{1}{2}\sqrt{2}$, since this gives smaller error coefficients in the local truncation error than the value $\gamma = 1 + \frac{1}{2}\sqrt{2}$. For the local error estimation within the variable step size control we use the embedded first-order formula

$$
\overline{w}_n = w_{n-1} + \bar{k}_1\,. \tag{2.2}
$$

We note that with our multirate approach, during the refinement step component values may be needed that are not included in this refinement. For example, components of $w(t)$ that are not refined may only be known in t_{n-1} and t_n; missing components will then be found by interpolation, and the F_t term in (2.1) will be approximated by

$$\widetilde{F}_t(t_{n-1}, w_{n-1}) \;=\; \frac{1}{\tau}(F(t_n, w_{n-1}) - F(t_{n-1}, w_{n-1}))\,. \tag{2.3}$$

This will not affect the order of the method. In all examples the exact Jacobian matrix $J = F_w(t_{n-1}, w_{n-1})$ will be used. For large practical problems a suitable approximation can be more efficient if that leads to more simple linear algebra systems.

The advantage of a Rosenbrock method is that only linear systems need to be solved. Implicit Runge–Kutta methods could also be used in our multirate approach, but then special attention should be given to the stopping criteria in Newton iterations. Making a large global time step with these methods might require many Newton iterations to get an iteration error smaller than a prescribed tolerance for the active spatial regions. But an accurate approximation is not needed there, because the numerical solution will be computed in the refinement steps. Therefore weighted norms should be used in the stopping criteria.

2.3 Variable step size control.

Let us consider an attempted step from time t_{n-1} to $t_n = t_{n-1} + \tau_n$ with step size τ_n. Suppose this is done with two methods of order p and $p-1$, giving the numerical solutions w_n and $\overline{w}_n$, respectively. By comparing w_n with $\overline{w}_n$ we obtain an estimate for the local error,

$$E_n = \|w_n - \overline{w}_n\|_\infty\,. \tag{2.4}$$

Here the maximum norm is used because we aim at errors below the tolerance for all components.

Having the estimate E_n and a tolerance *Tol* specified by the user, two cases can occur: $E_n > Tol$ or $E_n \le Tol$. In the first case we decide to reject this time step and to redo it with a smaller step size τ_{new}, where we aim at $E_{new} = Tol$. In the second case we decide to accept the step and to continue the integration from t_n to t_{n+1}. In both cases we continue with a time step of size

$$\tau_{new} \;=\; \vartheta\,\tau_n\,\sqrt[p]{Tol/E_n}\,, \tag{2.5}$$

where the safety factor $\vartheta < 1$ serves to make the estimate conservative so as to avoid repeated rejections.

This form of variable step size selection is standard; see for example [8, 14]. We will use it in two ways in our multirate approach: to select the time slabs and to determine the components for which smaller step sizes are to be taken.

3 Multirate time stepping strategy.

The time integration interval $[0, T]$ will be partitioned into synchronized time levels $0 = t_0 < t_1 < \ldots < t_N = T$. The length of the time slab $[t_{n-1}, t_n]$ will be denoted by Δt_n.

3.1 Strategy I: uniform treatment within time slabs.

3.1.1 Processing of one time slab.

Consider a single time slab $[t_{n-1}, t_n]$, as illustrated in Figure 1.1. Suppose that the approximation w_{n-1} at time t_{n-1} is known, and that we want to obtain an approximation w_n at the new time level. First we perform a single step with step size Δt_n and using an error estimator we determine the components for which the computation of the solution should be refined, that is, performed with a smaller time step. We refine for those components for which the estimated local error is larger than the prescribed tolerance *Tol*. This set of components is denoted as $\mathcal{J}_1$.

Refinement is done by doubling of the number of time steps for the selected set of components. So for all components in $\mathcal{J}_1$ we recalculate the solution using two steps of size $\frac{1}{2}\Delta t_n$. After this refinement phase we have the numerical solution for the set of components $\mathcal{J}_1$ at time levels $t_{n-1/2}$ and t_n. We then define $\mathcal{J}_2$ as the subset of components from $\mathcal{J}_1$ in which the estimated local error is still larger than the tolerance at either $t_{n-1/2}$ or t_n, and for all components from $\mathcal{J}_2$ we recalculate the solution using four time steps of size $\frac{1}{4}\Delta t_n$. This is repeated until the error estimator indicates that there is no need of smaller steps anymore. The processing of a time slab is then finished.

The interface, at the transition between the solutions obtained using different time step sizes, should be treated properly. For some components from the refinement set we will need solution values of components where we do not refine.

For example, in a first refinement step the solution is advanced for a part of the components using the halved time step $\frac{1}{2}\Delta t_n$. For the Rosenbrock method (2.1) this will require the values of the components at the time levels t_{n-1}, t_n and $t_{n-1/2}$. At time level t_{n-1} and t_n these values are available from the solution that has been computed with the coarse step Δt_n. At the intermediate time level $t_{n-1/2}$ we use interpolation based on the information available at t_{n-1} and t_n; this information consists of approximate solution values w_k and approximate derivative values $w'_k = F(t_k, w_k)$ for $k = n-1, n$.

In our tests, with the second-order method (2.1), we have examined linear interpolation based on w_{n-1} and w_n, and quadratic interpolation based on w_{n-1}, w'_{n-1} and w_n. For the numerical experiments presented in Section 4 both interpolations gave nearly identical results.

In general, the order of the interpolation should be related to the order of the time stepping method. With a basic integration method of order p, the error in one step will be $\sim \Delta t_n^{p+1}$. Interpolation with a q-th order polynomial will introduce an interpolation error $\sim \Delta t_n^{q+1}$ at the components in which we interpolate. Since we are interested in the errors in the maximum norm, the

choice $q = p$ is natural. On the other hand, it was observed, also for higher-order methods, that taking $q = p - 1$ often produces an order of accuracy equal to p for the whole scheme, due to damping and cancellation effects. A proper analysis for these effects is lacking at present.

3.1.2 Choosing the size of the time slabs.

The size of the time slabs will be determined automatically while advancing in time. When we are done with the processing of the n-th time slab of size Δt_n, the size of the next time slab is taken as

$$\Delta t_{n+1} = 2^{s_{n+1}} \tau^*_{n+1}\,, \tag{3.1}$$

where s_{n+1} is the estimated (desired) number of levels of refinement for the $(n+1)$-st time slab, and τ^*_{n+1} is the optimal step size which would give us an estimated error smaller than the given tolerance if we were to use a single-rate approach for the next time step from t_n to $t_n + \tau^*_{n+1}$. Both s_{n+1} and τ^*_{n+1} will be estimated using information from the last time slab. In general, s_{n+1} may not coincide with the actual number of levels of refinement that will be taken; we will usually refine until the estimated error is smaller than the imposed tolerance. The estimations for s_{n+1} and τ^*_{n+1} will be discussed in the next subsections.

For the first time slab we use $s_1 = 0$, meaning that we would like to make a single time step with an estimated error less than the prescribed tolerance *Tol* at all components. The size of the first time slab Δt_1 is estimated using a small prescribed test step size τ_0 together with the step size control formula

$$\Delta t_1 \;=\; \vartheta\, \tau_0 \;\sqrt[p]{Tol/E_0}\,, \tag{3.2}$$

where the safety factor ϑ, the tolerance *Tol* and the order p of the method are as in (2.5), and E_0 is the maximum norm of the estimated local error for the test step from 0 to τ_0. In the numerical experiments presented in this paper we use the ROS2 method ($p = 2$) with $\vartheta = 0.9$ and $\tau_0 = 10^{-4}$.

If the time integration is near an output point or the endpoint T, it should be verified whether $t_n + \Delta t_{n+1} > T$, and in that case we reset $\Delta t_{n+1} = T - t_n$.

In our implementation an additional check of the new time slab size Δt_{n+1} is made. This is to cover a situation where shortly after the last accepted time level t_n the solution suddenly becomes active. When after the global time step of size Δt_{n+1} has been performed it turns out that refinement is needed for each component, then the size of the time slab is deemed too large. In that case a smaller size Δt_{new} will be selected by making a new estimate τ^*_{new} based on the newly available information and we also set $s_{new} = \max(0, s_{n+1} - 1)$. Such rejections will only occur in exceptional situations, with the sudden appearance of new active terms in the equations.

3.1.3 Estimation of τ^*_{n+1}.

Using the information available from the n-th time slab we can estimate the value of τ^*_{n+1} for the next time slab. This is done using the standard step size

control technique; the only difference is that for each component we use the information from the last available local time steps from the last time slab $[t_{n-1}, t_n]$. For example, in the time slab depicted in Figure 3.1, in order to estimate τ^*_{n+1}, we will use the information from the hatched areas where the last local time steps before t_n have been taken.

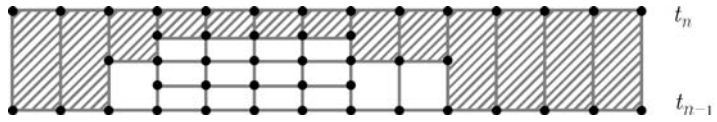

Figure 3.1: Time steps used for the estimation of τ^*_{n+1}.

After each level of refinement we know which components we already have refined (recall that for the k-th level of refinement this set of components is denoted by $\mathcal{J}_k$) and which components we ought to refine in the next level of refinement. Therefore, after the k-th level of refinement, for all components in $\mathcal{J}_k \setminus \mathcal{J}_{k+1}$, we estimate

$$\tau^{(k)}_{n+1} \;=\; \vartheta\, 2^{-k} \Delta t_n \, \sqrt[p]{Tol/E_k} \tag{3.3}$$

based on the local step sizes $2^{-k}\Delta t_n$ in the k-th level of refinement and on E_k, which is the maximum norm of the estimated error for the last time step at this level of refinement. The estimate in (3.3) represents the step size which would give us a local error smaller than the tolerance for all components from $\mathcal{J}_k \setminus \mathcal{J}_{k+1}$ if all is going well. The safety factor ϑ makes the estimate conservative.

After having finished with all levels of refinement we determine τ^*_{n+1} by

$$\tau^*_{n+1} \;=\; \min\left(\tau^{(0)}_{n+1}, \tau^{(1)}_{n+1}, \tau^{(2)}_{n+1}, \dots\right). \tag{3.4}$$

Expression (3.4) gives us an estimate of a step size with which we expect a local error smaller than the tolerance for all the components.

3.1.4 Estimation of s_{n+1}.

The estimation of s_{n+1} will be based on the anticipated amount of work needed to cover a unit of time. The multirate approach will introduce component-time points where the solution is computed several times, and this should be taken into account of course.

We suppose that the amount of work required for advancing one time step in m components is proportional to m^r with $r \geq 1$. In the experiments presented in this paper we use the two-stage Rosenbrock method (2.1) as our time integration method. At each stage of this method one vector-function evaluation is done and one system of linear algebraic equations with a band matrix is solved. Therefore, in this paper we can consider $r = 1$.

Suppose the n-th time slab has been processed using s_n levels of refinement, and that in the k-th level of refinement m_k components were refined, where $m_0 = m$. Since 2^k time steps were taken at this level of refinement to cover the

time slab, the amount of work involved with the k-th level of refinement is $2^k m_k^r$. The amount of work per time unit for the processing of the entire time slab is therefore considered to be

$$C = \frac{1}{\Delta t_n}\big(m_0^r + 2m_1^r + \cdots + 2^{s_n} m_{s_n}^r\big). \tag{3.5}$$

In order to estimate the optimal amount of work per time unit we also study two hypothetical (virtual) computations for this last time slab. In the first case we consider what would have happened if we had taken the size of the time slab 2^l times smaller than Δt_n, and in the second case what would have happened if we had taken the size twice as large as Δt_n. In both cases we can estimate the amount of work per unit time, and this can be compared to the actual amount C. This information will then be used for the next time slab.

For the first hypothetical case, let us assume we go back to the n-th time slab and redo it with $\Delta t'_n = \frac{1}{2^l}\Delta t_n$, that is, 2^l times smaller than the actual Δt_n. Then we would start with a time step of size $\Delta t'_n$ on the whole spatial domain ($m_0 = m$ points). The number of components involved in the first refinement, with two steps of size $\frac{1}{2}\Delta t'_n = \frac{1}{2^{l+1}}\Delta t_n$, can be estimated to be m_{l+1}, because that was the number of components used in the actual computation with this time step. In the same way we can estimate that in the k-th level of refinement we would refine in m_{l+k} components and that $s_n - l$ levels of refinement would be used. Hence, the amount of work per time unit for this hypothetical case would be approximately

$$C' = \frac{1}{\Delta t'_n}\big(m_0^r + 2m_{l+1}^r + \cdots + 2^{s_n - l} m_{s_n}^r\big)\,. \tag{3.6}$$

If $C' < C$, we estimate that this hypothetical step would have given an improvement in the amount of work, compared to the actual computation that has been done.

LEMMA 3.1. *Let $\rho = \left(\frac{1}{2}\right)^{1/r}$. The value of C' in* (3.6) *attains its minimum for*

$$l_* = \max\{\, l \,:\, m_l > \rho m\}\,. \tag{3.7}$$

PROOF. Denote the right-hand side of (3.6), with $\Delta t'_n = 2^{-l}\Delta t_n$, by C'_l. Then it is easily seen that

$$C'_{l-1} < C'_l \quad (\text{resp. } C'_{l-1} > C'_l) \quad \Longleftrightarrow \quad m_l < \rho m \quad (\text{resp. } m_l > \rho m)\,. \tag{3.8}$$

For the value l_* in (3.7) we have

$$m = m_0 \geq m_1 \geq \cdots \geq m_{l_*} > \rho m \geq m_{l_*+1} \geq \cdots \geq m_{s_n}\,.$$

It thus follows from (3.8) that

$$C'_0 > C'_1 > \cdots > C'_{l_*} \quad \text{and} \quad C'_{l_*} \leq C'_{l^*+1} \leq \cdots \leq C'_{s_n}\,,$$

which provides the proof of the lemma. □

If $l_* > 0$, then an improvement in amount of work per unit step could have been obtained if the n-th time slab had been done with fewer levels of refinement and a smaller size of the time slab. Therefore, for the $(n+1)$-st time slab we try to improve the performance by taking

$$s_{n+1} = s_n - l_* \,. \tag{3.9}$$

If $l_* = 0$, there was apparently no need to decrease the number of levels of refinement. But then more efficiency might be possible with a time slab of larger size (with more levels of refinement) than in the actual computation. This leads us to the second hypothetical case.

If the size of the n-th time slab had been two times larger than Δt_n, that is $\Delta t_n'' = 2\Delta t_n$, then one time step of size $\Delta t_n''$ for all the components ($m_0 = m$ points) would have been performed, followed by refinement steps. Suppose that the first level of refinement would have involved m_* components. The second level of refinement then would take four time steps of size $\frac{1}{4}\Delta t_n'' = \frac{1}{2}\Delta t_n$. In the processing of the original time slab of size Δt_n we needed time steps of this size in m_1 components. Therefore, it can be assumed that for the second level of refinement in the virtual step, refinement would also take place on m_1 components. Similarly, the k-th level of refinement can be assumed to involve m_{k-1} components. In total we would have s_n+1 levels of refinement. The amount of work per time unit for this case would thus be approximately

$$C'' \,=\, \frac{1}{\Delta t_n''}\big(m_0^r + 2m_*^r + 2^2 m_1^r + \cdots + 2^{s_n+1} m_{s_n}^r\big). \tag{3.10}$$

In this case, taking the size of the time slab two times larger than Δt_n, would give us an expected improvement in work per time unit if $C > C''$, that is,

$$m_* \,<\, \rho\, m\,, \quad \rho \,=\, \big(\tfrac{1}{2}\big)^{1/r}\,. \tag{3.11}$$

We still need an estimate for m_*. Let $v = w_n - \bar{w}_n$ be the difference between one step in the embedded Rosenbrock method (2.1), (2.2) computed in the n-th time slab with step size Δt_n, and let $E_n = \|v\|_\infty$ be the norm of this estimated local error. Then $E_n \sim (\Delta t_n)^p$, with order $p = 2$ for the present Rosenbrock combination. In the first stage of our hypothetical step, starting from t_{n-1} with time step $2\Delta t_n$, we would expect an estimated local error of size $2^p E_n$. Consider the index set

$$\mathcal{I}_1 = \{\, i \,:\, |v_i| > 2^{-p}\, Tol\}\,, \tag{3.12}$$

where v_i is the i-th component of the vector v. Then m_* will be approximately equal to the number of elements $|\mathcal{I}_1|$ in this set. This estimate of m_* can be determined during the actual processing of the time slab without significant extra work. If

$$|\mathcal{I}_1| \,<\, \rho m\,, \tag{3.13}$$

then it is expected that a larger time slab with more refinement levels would have been more efficient. For the next time slab we then take $s_{n+1} = s_n + 1$. We note that a larger increase of refinement levels could be considered in a similar way, but it seems better to be conservative about this, because $s_{n+1} = s_n + 1$ will already lead (approximately) to a doubling of the size of the time slab (if $\tau^*_{n+1} \approx \tau^*_n$).

Summarizing, after having completed the n-th time slab with s_n levels of refinement, we choose for the next time slab

$$s_{n+1} = \begin{cases} s_n + 1 & \text{if (3.13) is satisfied}, \\ s_n - l_* & \text{if (3.13) is not satisfied}, \end{cases} \tag{3.14}$$

where $l_* \geq 0$ is defined by (3.7). Together with (3.1) and (3.4) this determines the size Δt_{n+1} of the new time slab. The actual number of levels of refinements will be determined by the error estimations. The s_{n+1} in (3.14) is merely an indication for this. In our experiments the s_{n+1} was usually equal to the number of levels of refinements, but sometimes it was one more or one less.

3.2 Strategy II: recursive two-level approach.

The time slab processing strategy presented in the above generally works very well, but in some cases a modification is desirable.

It may happen that the strategy takes very large time slabs with a large number of refinement levels. Then the smallest time steps are used throughout the entire time slab. Although this is only for a subset of components, it can be inefficient if the local temporal variation changes drastically inside this large time slab. Then the small time steps may be needed only in some part of the time slab $[t_{n-1}, t_n]$. In such a situation our strategy can be improved by applying the refinements not on the whole time slab but just for the required, smaller time intervals.

Let us consider a time slab $[t_{n-1}, t_n]$ with known approximation w_{n-1}. As before, we start with a single step $\Delta t_n = t_n - t_{n-1}$, and use the error estimator to determine the components where we should refine in time. For those components the time slab is divided into two smaller sub-slabs with size $\frac{1}{2}\Delta t_n$. Next, each of these sub-slabs is processed separately, in a similar way as the initial 'global' time slab. This is a recursive processing strategy, which stops when the error estimator indicates that there is no need of further subslabs. A simple illustration for two levels of refinement is given in Figure 3.2.

This modified time slab size processing strategy is considered in combination with a slightly modified time slab estimation. In the modified version the time

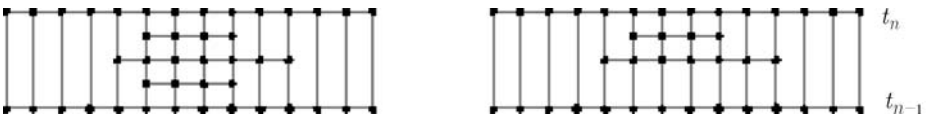

Figure 3.2: Example of a time slab created by the original strategy I (left) and the modified strategy II (right).

slabs have a different structure; they are no longer uniform over the whole time slab. Therefore, not all the rationale from the previous time slab size estimation strategy can be used directly for the modified version.

The size of a time slab can still be determined using the same formula

$$\Delta t_{n+1} = 2^{s_{n+1}} \tau^*_{n+1}. \tag{3.15}$$

The value for τ^*_{n+1} can be determined using exactly the same procedure as in our original multirate strategy. The desired number of the levels of refinement s_{n+1} was determined on the basis of values of the number of components $m_0, m_1, \ldots, m_{s_n}$ in the levels of refinement for the n-th time slab. For the modified strategy these numbers of components are not constant anymore over the time slab. Still, for the new time slab we have as a first guess that the refinement will proceed uniformly as in the last local steps before t_n. Therefore, the estimations of the amount of work is done in the same way as before, but now with values of m_l based on the last available local steps before t_n. Using these values m_l we can determine the desired number of levels of refinement following the same procedure and rationale as in our original strategy. The size of the time slab obtained in this way is the optimal size which can be obtained based on the last information from the previous time slab.

3.3 Comparison to existing time slab strategies.

Another time slab strategy has been presented by Jansson and Logg [10] for the multi-adaptive Galerkin time-stepping algorithm of Logg [12, 13]. In their strategy a time slab is created by first computing a desired time step for all components. The size of the time slab is then taken as $\Delta t = \theta \tau_{max}$ with τ_{max} the maximum over the desired time steps and $\theta \in (0, 1)$ a fixed parameter. The components are then partitioned into two sets. The components in the group with large time steps are integrated with time step $\theta \Delta t$. The remaining components are processed by a recursive application of the same procedure.

In this multi-adaptive Galerkin approach, the resulting implicit systems for all refinements are solved simultaneously. This is the main difference with our approach. We first solve the coarse step, and then, successively, the refined steps. This leads to some overhead because in the refined regions the solution is computed repeatedly. On the other hand, with our approach the implicit systems are all relatively simple; basically the same as in a single-rate approach for (1.1) but with fewer points m_k in the refined steps. The dimension of the implicit systems in the approach of Logg will be very much larger than m, the number of components in (1.1), so these systems will be very hard to solve. For this reason a damped functional (fixed point) iteration is used in [10], but that can easily lead to a very large number of iterations per time slab.

In our case the size of a time slab is computed from the minimum time step over the components and an expected number of levels of refinement. In our strategy the sizes of the time slabs and the numbers of levels of refinement are automatically adjusted to get an optimal amount of work per time unit.

4 Numerical experiments.

In this section we will present numerical results for several test problems. We consider the behaviour of both our strategies: Multirate I (with uniform treatment within time slabs) and Multirate II (with the recursive two-level approach). The results are compared to the single-rate approach, also using the Rosenbrock pair (2.1) and (2.2).

As measure for the amount of work we consider primarily the number of components for which the solution is computed during the whole integration, where the fact that with our multirate approach some solution components will be computed several times at certain time levels is taken into account. For practical purposes the CPU time is more relevant, but this depends strongly on the programming language and environment. Some resulting computing times for a C-program will be discussed.

As mentioned before, the amount of work per step for m components in these experiments is estimated as m^r with $r = 1$. Tests with $r = 2$, which is obviously a wrong value here, produced quite similar results. In general, the choice of r will depend on the problem and linear algebra solver. The tests with $r = 2$ indicate that an optimal estimate for r is not critical for the performance of our multirate schemes.

One of the test problems is an ODE system from circuit analysis, the other two are obtained from partial differential equations (PDEs) by standard second-order central discretization of the spatial derivatives on fixed uniform grids (fourth-order central differences were also tried and the results were very similar). The resulting semi-discrete systems are simply considered as ODE test problems in these numerical experiments.

For the results reported here we used quadratic interpolation to obtain missing component values. Linear and cubic interpolation were also tried and the results were nearly identical; this simply indicates that the interpolation errors are not significant in these tests. Linear interpolation could potentially lower the order of accuracy, which is two for the ROS2 method, and therefore quadratic interpolation is our preferred interpolation here. As mentioned before, with a higher order basic time stepping method, also the order of interpolation should be increased. For a number of Runge–Kutta and Rosenbrock methods dense output formulas are available [8] which can also be considered.

The errors presented in the tables below are the maximum errors over the components at the output times T, with respect to a time-accurate ODE reference solution. The reference solutions have been computed by using very small tolerance values.

4.1 An ODE system obtained from semi-discretization: a reaction-diffusion problem with traveling wave solution.

For our first test problem we consider the semi-discrete system obtained from the reaction-diffusion equation

$$u_t = \epsilon u_{xx} + \gamma u^2(1-u), \tag{4.1}$$

for $0 < x < L$, $0 < t \leq T$. The initial- and boundary conditions are given by

$$u_x(0,t) = u_x(L,t) = 0\,, \qquad u(x,0) = (1+\mathrm{e}^{\lambda(x-1)})^{-1}\,, \tag{4.2}$$

where $\lambda = \frac{1}{2}\sqrt{2\gamma/\epsilon}$. If the spatial domain had been the whole real line, then the initial profile would have given the traveling wave solution $u(x,t) = u(x - \alpha t, 0)$ with velocity $\alpha = \frac{1}{2}\sqrt{2\gamma\epsilon}$. In our problem, with homogeneous Neumann boundary conditions, the solution will still be very close to this traveling wave provided the end time is sufficiently small so that the wave front does not come close to the boundaries. The parameters are taken as $\gamma = 1/\epsilon = 100$ and $L = 5$, $T = 3$. In space we used a uniform grid of $m = 1000$ points and standard second-order differences, leading to an ODE system in $\mathbb{R}^m$. An illustration of the semi-discrete solution at various times is given in Figure 4.1 with (spatial) components horizontally.

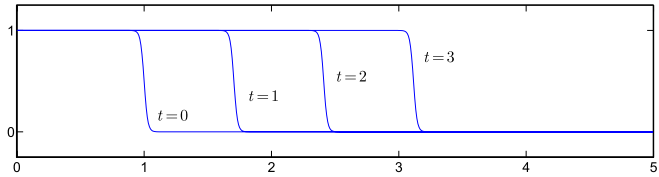

Figure 4.1: Traveling wave solution for problem (4.1)–(4.2) at various times.

In Table 4.1 the errors (in the maximum norm with respect to the reference ODE solution at time T) and the amount of work (number of space-time points for the integration interval $[0,T]$) are presented for different tolerances. From these results it is seen that a substantial improvement in amount of work is obtained for this problem. For the single-rate scheme, the number of space-time points where the solution is computed is almost seven times larger. Moreover, the error behavior of the multirate scheme is very good. We have roughly a proportionality of the errors and tolerances, and the errors of the multirate scheme are approximately the same as for the single-rate scheme.

Table 4.1: Errors and work amount for (semi-discrete) problem (4.1)–(4.2).

	Single-rate		Multirate I		Multirate II	
Tol	error	work	error	work	error	work
10^{-3}	$3.2 \cdot 10^{-3}$	818818	$3.4 \cdot 10^{-3}$	188138	$2.1 \cdot 10^{-3}$	124356
$5 \cdot 10^{-4}$	$1.9 \cdot 10^{-3}$	1128127	$1.9 \cdot 10^{-3}$	246962	$2.2 \cdot 10^{-3}$	149763
10^{-4}	$4.8 \cdot 10^{-4}$	2431429	$5.1 \cdot 10^{-4}$	411466	$5.4 \cdot 10^{-4}$	308685
$5 \cdot 10^{-5}$	$2.5 \cdot 10^{-4}$	3408405	$2.7 \cdot 10^{-4}$	550723	$2.7 \cdot 10^{-4}$	428549
10^{-5}	$5.3 \cdot 10^{-5}$	7528521	$5.5 \cdot 10^{-5}$	1153759	$5.7 \cdot 10^{-5}$	1064115

Measurements of CPU times (for a C-program) showed that for this problem the single-rate scheme was approximately four times more expensive than the multirate schemes. This factor four is less than the factor seven in space-time

points; this is due to overhead with the multirate schemes for determining the time slabs and refinement regions.

The multirate strategy II (recursive two-level approach) works somewhat better for this problem than strategy I, in particular for the larger tolerances. In Figure 4.2 the space-time grid is shown on which the solution was calculated for strategy I with tolerance value $Tol = 2 \cdot 10^{-2}$. (With this large tolerance the structure of the grid is better visible than with small tolerances.) One nicely sees that the refinements move along with the steep gradient in the solution. From the more detailed picture (enlargement on part of the domain), it is seen that there is some redundancy in the fine level computations: in each time slab the fine level domains form a rectangle, and this is the reason why the strategy II is more efficient for this problem. Figure 4.3 shows the space-time grid for strategy II, again with $Tol = 2 \cdot 10^{-2}$.

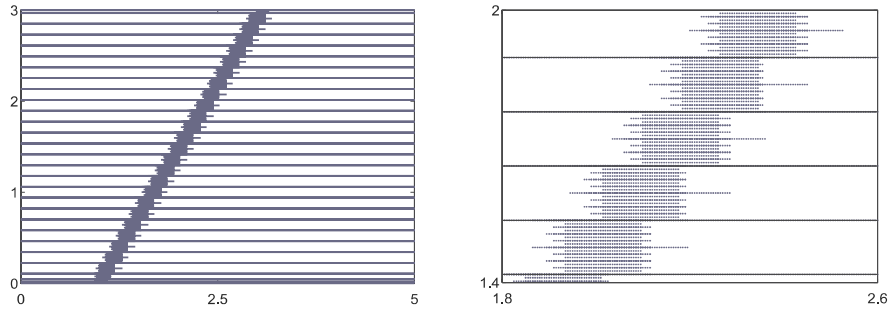

Figure 4.2: Space-time grid for problem (4.1)–(4.2) with strategy I. The right picture gives an enlargement for a part of the domain.

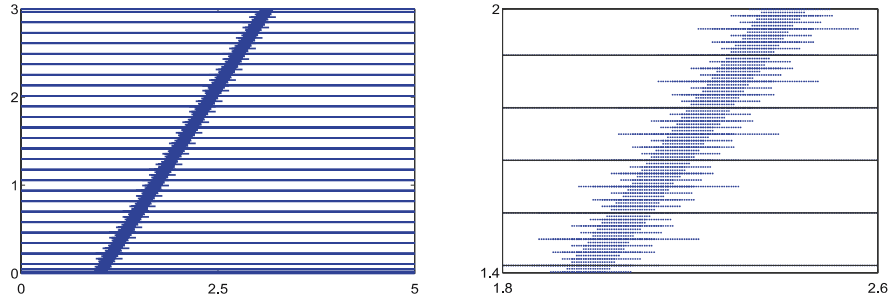

Figure 4.3: Space-time grid for problem (4.1)–(4.2) with strategy II. The right picture gives an enlargement for a part of the domain.

4.2 An ODE system obtained from semi-discretization: the Allen–Cahn equation.

The second test consists of a semi-discrete version of the Allen–Cahn equation

$$u_t = \epsilon u_{xx} + u(1 - u^2)\,, \tag{4.3}$$

for $t > 0$, $-1 < x < 2$, with initial- and boundary conditions

$$u_x(-1,t) = 0\,, \qquad u_x(2,t) = 0\,, \qquad u(x,0) = u_0(x)\,. \tag{4.4}$$

We take $\epsilon = 9 \cdot 10^{-4}$ and initial profile

$$u_0(x) = \begin{cases} \tanh((x+0.9)/(2\sqrt{\epsilon})) & \text{for } -1 < x < -0.7\,, \\ \tanh((0.2-x)/(2\sqrt{\epsilon})) & \text{for } -0.7 \le x < 0.28\,, \\ \tanh((x-0.36)/(2\sqrt{\epsilon})) & \text{for } 0.28 \le x < 0.4865\,, \\ \tanh((0.613-x)/(2\sqrt{\epsilon})) & \text{for } 0.4865 \le x < 0.7065\,, \\ \tanh((x-0.8)/(2\sqrt{\epsilon})) & \text{for } 0.7065 \le x < 2\,. \end{cases} \tag{4.5}$$

This problem is an extended version version of the bistable problem considered in [5].

For this problem we used a uniform space grid of 400 points with second-order central differences. Figure 4.4 shows a time-accurate numerical solution. The nonlinear reaction term in (4.3) has $u = 1$ and $u = -1$ as stable equilibrium states, whereas the zero solution is an unstable equilibrium. The solution of (4.3)–(4.5) starts with three 'wells', see Figure 4.4. The first well, on the left, persists during the integration interval. The second well is somewhat thinner than the others and it collapses at time $t \approx 41$, whereas the third well collapses at $t \approx 141$.

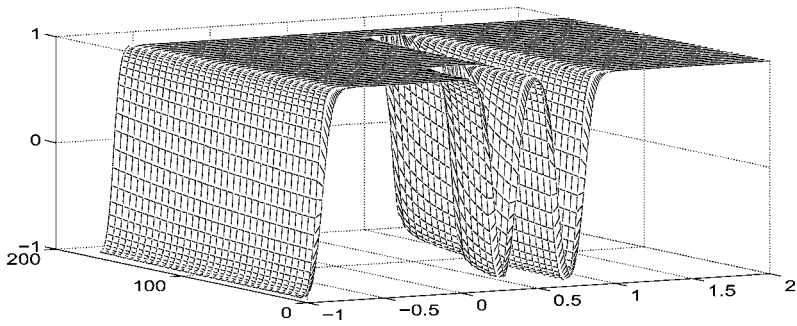

Figure 4.4: Evolution of the solution for problem (4.3)–(4.5).

To test the performance of the schemes, the output was considered for $T = 142$. At this output point, the solution is still changing in the second well; for larger times the solution becomes steady-state and then all errors vanish. In Table 4.2 the errors (measured in the maximum norm with respect to the reference ODE solution) and the amount of work (number of space-time points) for different tolerances are presented. For this problem there is again a significant improvement in work with the multirate schemes compared to the single-rate scheme.

Strategy II again behaves slightly better for this problem than strategy I. The error behavior of both multirate schemes is excellent: the errors are close to – or

Table 4.2: Errors and work amount for (semi-discrete) problem (4.3)–(4.5).

	Single-rate		Multirate I		Multirate II	
Tol	error	work	error	work	error	work
$5 \cdot 10^{-4}$	$3.8 \cdot 10^{-3}$	102255	$3.0 \cdot 10^{-3}$	48342	$3.6 \cdot 10^{-3}$	36811
10^{-4}	$2.2 \cdot 10^{-3}$	217743	$1.5 \cdot 10^{-3}$	85241	$1.1 \cdot 10^{-3}$	66360
$5 \cdot 10^{-5}$	$1.2 \cdot 10^{-3}$	303958	$1.0 \cdot 10^{-3}$	107920	$1.3 \cdot 10^{-3}$	75653
10^{-5}	$2.8 \cdot 10^{-4}$	664858	$2.5 \cdot 10^{-4}$	257473	$2.6 \cdot 10^{-4}$	227554
$5 \cdot 10^{-6}$	$1.3 \cdot 10^{-4}$	935533	$1.1 \cdot 10^{-4}$	355627	$1.2 \cdot 10^{-4}$	324501

even smaller than – the errors of the single-rate scheme. As in the other tests, this shows that our multirate strategies behave very robustly.

In CPU times the factor gained with the multirate schemes, compared to the single-rate scheme, was a factor two approximately. As for the previous problem this is somewhat less than the factors for the number of space-time points due to overhead.

4.3 An inverter chain problem.

An inverter is an electrical sub-circuit which transforms a logical input signal to its negation. The inverter chain is a concatenation of several inverters, where the output of an inverter serves as input for the succeeding one. An inverter chain with an even number of inverters will delay a given input signal and will also provide some smoothing of the signal.

A detailed description of a mathematical model for an inverter chain is given in [1]. The model for m inverters consists of the equations

$$\text{(4.6)} \qquad \begin{cases} w_1'(t) = U_{\rm op} - w_1(t) - \Upsilon g\big(u_{\rm in}(t), w_1(t)\big)\,, \\ w_j'(t) = U_{\rm op} - w_j(t) - \Upsilon g\big(w_{j-1}(t), w_j(t)\big)\,, \quad j = 2, \ldots, m\,, \end{cases}$$

where

$$\text{(4.7)} \qquad g(u,v) = \big(\max(u - U_{\rm thres}, 0)\big)^2 - \big(\max(u - v - U_{\rm thres}, 0)\big)^2.$$

The coefficient Υ serves as stiffness parameter. Following [1, 2], we solve the problem for a chain of $m = 500$ inverters with $\Upsilon = 100$, $U_{\rm thres} = 1$ and $U_{\rm op} = 5$. The initial condition is

$$\text{(4.8)} \qquad w_j(0) = 6.247 \cdot 10^{-3} \ \text{ for } j \text{ even}, \quad w_j(0) = 5 \ \text{ for } j \text{ odd}.$$

The input signal is given by

$$\text{(4.9)} \qquad u_{\rm in}(t) = \begin{cases} t - 5 & \text{for} \quad 5 \le t \le 10\,, \\ 5 & \text{for} \quad 10 \le t \le 15\,, \\ \frac{5}{2}(17 - t) & \text{for} \quad 15 \le t \le 17\,, \\ 0 & \text{otherwise.} \end{cases}$$

An illustration of the solution for some of the even components is given in Figure 4.5.

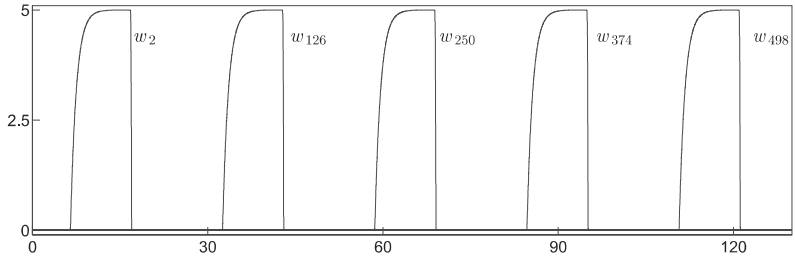

Figure 4.5: Solution components $w_j(t)$, $j = 2, 126, 250, 374, 498$, for problem (4.6)–(4.9).

In Table 4.3 maximal errors over $t_n \in [0, T]$, $T = 130$, and over all components (measured with respect to an accurate reference solution) together with the amount of work and CPU times (in seconds) are presented for several tolerances for the single-rate scheme and the multirate strategy II. Similar as for the previous examples, the amount of work is taken as the number of components at which solutions are computed over the integration interval $[0, T]$; this is proportional to the number of scalar function evaluations (4.7).

It is seen from the table that for the prescribed tolerances we get roughly a factor 10 of improvement in work and a factor 6 improvement in CPU time with the multirate scheme, whereas for each given tolerance the errors of the multirate scheme are somewhat smaller than with the single-rate scheme.

In Figure 4.6 the component-time grid is shown on which the solution was calculated with tolerance value $Tol = 5 \cdot 10^{-2}$. For this large tolerance the structure of the grid is better visible than for smaller tolerances, but still only every tenth global step is displayed in the left picture to make it more clear. Again it is seen that the refinements are properly adjusted to the steep gradients in the various components of the solution.

The same problem served as a numerical test for a multirate W-method in [2], where for each time slab a partitioning of the components in two classes, slow (latent) and fast (active), was used; the partitioning was based on a monitor function suited for this particular problem. Inside a time slab, all fast components were then solved with the same small step sizes (micro-steps). In this way a factor 3.7 of improvement in work was obtained compared to the single-rate

Table 4.3: Errors and work amount for problem (4.6)–(4.8).

	Single-rate			Multirate II		
Tol	error	work	CPU	error	work	CPU
$5 \cdot 10^{-4}$	$1.74 \cdot 10^{-1}$	28938500	19.75	$1.12 \cdot 10^{-1}$	3314690	3.74
$1 \cdot 10^{-4}$	$3.91 \cdot 10^{-2}$	62379000	42.64	$2.41 \cdot 10^{-2}$	4795878	6.36
$5 \cdot 10^{-5}$	$2.10 \cdot 10^{-2}$	87384000	59.72	$1.88 \cdot 10^{-2}$	6456558	8.81
$1 \cdot 10^{-5}$	$6.07 \cdot 10^{-3}$	193494000	132.32	$3.84 \cdot 10^{-3}$	17358472	21.65

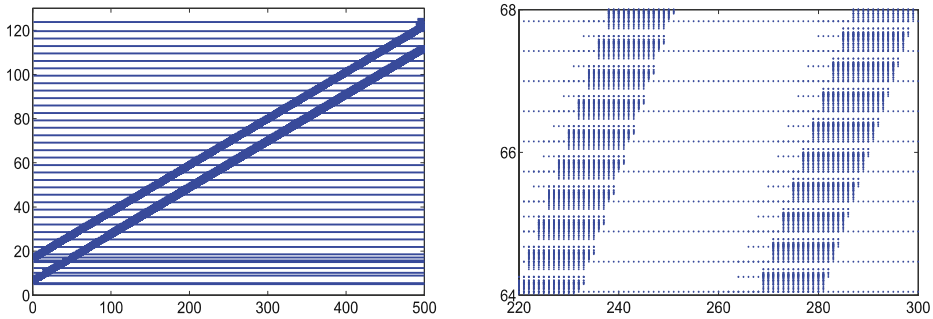

Figure 4.6: Component-time grid for problem (4.1)–(4.2) with strategy II. The right picture gives an enlargement for a part of the components and time interval.

scheme. With our strategy this factor is much higher. This seems mainly due to the dynamic partitioning into several classes, together with the choices for the size of the time slabs and local time steps found by estimating the total amount of work.

Apart from the partitioning, the most important difference between [2] and our approach is the use of a 'compound step' in [2], whereby the slow components and the first (micro-) step for the fast components are solved simultaneously. Here extrapolation (from fast to slow components) and interpolation (from slow to fast components) is incorporated. In this way some of the overhead in our approach is avoided, because there are no coarse step values that are later overwritten, but such compound steps will become very complicated if the components are partitioned into more than two classes.

5 Conclusions.

In this paper we presented self-adjusting multirate time stepping strategies for stiff ODEs. The step size for a particular system component is determined by the local temporal variation of the solution, in contrast to the use of a single step size for the whole set of components as in the traditional (single-rate) methods. Numerical experiments confirmed that the efficiency of time integration methods can be significantly improved by using large time steps for inactive components, without sacrificing accuracy.

Although our two strategies produced results not too far apart, we do have a slight preference for the recursive two-level approach (strategy II) over the uniform treatment within time slabs (strategy I). Cases can be constructed with very large time slabs where strategy II will be much more efficient than strategy I.

Compared to the approaches in [2, 7] and [12, 13], our multirate approach avoids the use of compound steps or very large implicit systems. On the other hand, there is some overhead with our approach, because in the refined component sets the solution is computed repeatedly. We do think, however, that for many problems simplicity will be preferable. Since the structure of the problems

with the refined steps is the same as for the original problems, only on smaller component sets, linear algebra solvers suitable for the single-rate scheme can still be used.

As basic time stepping method, we used in this paper a second-order Rosenbrock method with an embedded first-order method. Except for the interpolation, the multirate approach can be applied without adjustments to higher-order methods. Preliminary experiments with fourth-order Rosenbrock schemes are promising.

Acknowledgments.

We would like to thank A. Verhoeven and E. J. W. ter Maten from Philips Research for suggesting the inverter chain test problem.

REFERENCES

1. A. Bartel, *Generalised Multirate. Two ROW-type versions for circuit simulation*, Unclass. Natlab Report No. 2000/84, Philips Electronics, 2000.
2. A. Bartel and M. Günther, *A multirate W-method for electrical networks in state space formulation*, J. Comput. Appl. Math., 147 (2002), pp. 411–425.
3. C. Engstler and C. Lubich, *Multirate extrapolation methods for differential equations with different time scales*, Computing, 58 (1997), pp. 173–185.
4. C. Engstler and C. Lubich, *MUR8: A multirate extension of the eight-order Dormand–Prince method*, Appl. Numer. Math., 25 (1997), pp. 185–192.
5. D. Estep, M. G. Larson, and R. D. Williams, *Estimating the Error of Numerical Solutions of Systems of Reaction-Diffusion Equations*, Mem. Am. Math. Soc., 146 (2000).
6. C. Gear and D. Wells, *Multirate linear multistep methods*, BIT, 24 (1984), pp. 484–502.
7. M. Günther, A. Kværnø, and P. Rentrop, *Multirate partitioned Runge–Kutta methods*, BIT, 41 (2001), pp. 504–514.
8. E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II – Stiff and Differential-Algebraic Problems*, 2nd edn., Springer Series in Comput. Math. 14, Springer, Berlin, 1996.
9. W. Hundsdorfer and J. G. Verwer, *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*, Springer Series in Comput. Math. 33, Springer, Berlin, 2003.
10. J. Jansson and A. Logg, *Algorithms for multi-adaptive time stepping*, Chalmers Finite Element Center, Preprint 2004-13, 2004.
11. A. Kværnø, *Stability of multirate Runge–Kutta schemes*, Int. J. Differ. Equ. Appl., 1A (2000), pp. 97–105.
12. A. Logg, *Multi-adaptive Galerkin methods for ODEs I*, SIAM J. Sci. Comput., 24 (2003), pp. 1879–1902.
13. A. Logg, *Multi-adaptive Galerkin methods for ODEs II. Implementation and applications*, SIAM J. Sci. Comput., 25 (2003), pp. 1119–1141.
14. L. F. Shampine, *Numerical Solution of Ordinary Differential Equations*, Chapman & Hall, New York, 1994.