Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

MAS

Modelling, Analysis and Simulation

Modelling, Analysis and Simulation

A multirate time stepping strategy for parabolic PDE

V. Savcenco, W.H. Hundsdorfer, J.G. Verwer

REPORT MAS-E0516 AUGUST 2005

CWI is the National Research Institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organization for Scientific Research (NWO).
CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-3703

A multirate time stepping strategy for parabolic PDE

ABSTRACT
To solve PDE problems with different time scales that are localized in space, multirate time stepping is examined. We introduce a self-adjusting multirate time stepping strategy, in which the step size at a particular grid point is determined by the local temporal variation of the solution, instead of using a minimal single step size for the whole spatial domain. The approach is based on the `method of lines', where first a spatial discretization is performed, together with local error estimates for the resulting semi-discret system. We will primarily consider implicit time stepping methods, suitable for parabolic problems. Our multirate strategy is tested on several parabolic problems in one spatial dimension (1D).

2000 Mathematics Subject Classification: 65L06, 65L50, 65M06, 65M20
Keywords and Phrases: Multirate time stepping, local time stepping, partial differential equations
Note: *Acknowledgement:* The work of V. Savcenco is supported by a Peterich Scholarship through the Netherlands Organisation for Scientific Research NWO.

A Multirate Time Stepping Strategy for Parabolic PDEs

V. Savcenco*, W. Hundsdorfer, J.G. Verwer
CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract

To solve PDE problems with different time scales that are localized in space, multirate time stepping is examined. We introduce a self-adjusting multirate time stepping strategy, in which the step size at a particular grid point is determined by the local temporal variation of the solution, instead of using a minimal single step size for the whole spatial domain. The approach is based on the 'method of lines', where first a spatial discretization is performed, together with local error estimates for the resulting semi-discret system. We will primarily consider implicit time stepping methods, suitable for parabolic problems. Our multirate strategy is tested on several parabolic problems in one spatial dimension (1D).

2000 Mathematics Subject Classification: 65L06, 65L50, 65M06, 65M20
Keywords and Phrases: Multirate time stepping, local time stepping, partial differential equations
Acknowledgement: The work of V. Savcenco is supported by a Peterich Scholarship through the Netherlands Organisation for Scientific Research NWO.

1 Introduction

There are usually two stages in the numerical solution of time-dependent partial differential equations (PDEs). The first stage is the spatial discretization in which the spatial derivatives of the PDE are discretized, for example with finite differences, finite volumes or finite element schemes. By discretizing the spatial operators, the PDE with its boundary conditions is converted into a system of ordinary differential equations (ODEs) in $\mathbb{R}^m$,

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

called the semi-discrete system. It will be assumed that the components of $w(t)$ are associated with the PDE solution at grid points or surrounding cells, say $w_i(t) \approx u(x_i, t)$, or with the weights of local trial functions in finite elements. This ODE system is still continuous in time and needs to be integrated. So, the second stage in the numerical solution is the numerical time integration of system (1.1).

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

The multirate approach introduced in this paper is based on local temporal error estimation. Given a global time step $\Delta t_n = t_n - t_{n-1}$, we compute an approximation at

*Corresponding author. Email address: savcenco@cwi.nl

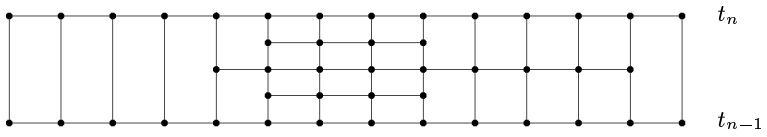

Figure 1: Multirate time stepping for a time slab $[t_{n-1}, t_n]$.

the new time level in all grid points. In those spatial regions where the error estimator indicates that smaller steps are needed, the computation is redone with $\frac{1}{2}\Delta t_n$. The result with the coarser time step will furnish boundary conditions for this refined step 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 certain tolerance for all grid points. Schematically, with space horizontally and time vertically, the multirate time stepping is displayed in Figure 1. Small time steps will be used in the more active space region and larger ones in the less active space region.

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 involving the imposed 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 amount of grid points at which the solution needs to be calculated, including the overhead due to repeated computations in refined regions.

In this report we use one-step Rosenbrock methods. Further we consider one-dimensional problems and fixed spatial grids. Variable grids and higher spatial dimensions is the focus of our present research. We note that due to the simple one-step nature of our approach, after each completed time slab a regridding in space could be performed without affecting the strategy described in this paper.

This paper is organized as follows. In Section 2 we will briefly discuss related work on multirate schemes and introduce the two-stage Rosenbrock method that will be used as our basic numerical integration method. It should be noted that other one-step methods could also be used directly within our approach. In Section 3 the multirate time stepping is described in detail, together with the time slab strategies. In Section 4 we will discuss the performance of the schemes through several numerical experiments for nonlinear parabolic problems. Section 5 contains the conclusions and an outlook on further work.

2 Background material and preliminaries

2.1 Related work

Multirate algorithms based on multistep methods have been described by Gear and Wells [6]. Their work represents some first, tentative steps towards an automatic multirate method. With multistep methods, extrapolations of past values may be needed. Moreover, the use of multistep methods will make a coupling with grid refinements in space more difficult than with one-step methods. For these reasons we decided to use one-step methods.

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 partic-

ular, 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]. Kværnø[11] has presented some stability results for such schemes applied to systems of two linear equations with one fast and one slow component.

An algorithm based on finite elements in time 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

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

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 report 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 used in variable step size control we will use the embedded first-order Rosenbrock method

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

The test problems that will be considered in this paper lead to semi-discrete systems (1.1) of the type

$$w'(t) = Aw(t) + b(t) + G(w(t))\,, \tag{2.3}$$

where A is a discretized spatial operator, $G(w)$ a reaction term, and $b(t)$ is a vector containing discretized Dirichlet boundary values. We note that with our multirate approach, problems arise on parts of the spatial region with time dependent Dirichlet conditions, even if these are not present in the original semi-discrete system (1.1). The boundary term $b(t)$ may only be known in t_{n-1} and t_n; missing values will then be found by interpolation, and the F_t term in (2.1) will usually be approximated by

$$\widetilde{F}_t(t_{n-1},w_{n-1}) = \frac{1}{\tau}\big(b(t_n)-b(t_{n-1})\big)\,. \tag{2.4}$$

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,

$$E_n = \|w_n - \overline{w}_n\|_\infty \,. \tag{2.5}$$

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

Having the estimate E_n and a tolerance Tol specified by the user, two cases can occur: $E_n > Tol$ or $E_n \leq Tol$. 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

$$\tau_{new} \;=\; \vartheta\, \tau_n \, \sqrt[p]{Tol/E_n}\,, \tag{2.6}$$

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 [14]. We will use it in two ways in our multirate approach: to select the time slabs and to determine the local regions where 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

Let us consider a single time slab $[t_{n-1}, t_n]$, as illustrated in Figure 1. Suppose that the approximation w_{n-1} at time t_{n-1} is known, and 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 spatial region where the computation of the solution should be refined, that is, performed with a smaller time step. We refine in those spatial points in which the estimated local error is larger than a prescribed tolerance Tol. This set of points will be denoted as $\mathcal{J}_1$.

Refinement is done by doubling of the number of time steps on the local spatial region. So at all points in $\mathcal{J}_1$ we recalculate the solution using two steps of size $\frac{1}{2}\Delta t_n$.

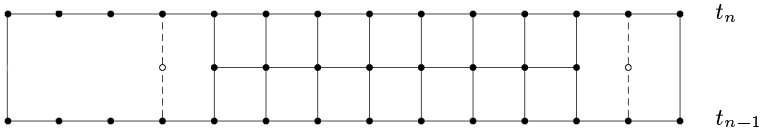

Figure 2: Refinement step in a time slab with interpolation points at the internal boundaries.

After this refinement phase we get the numerical solution in the spatial region $\mathcal{J}_1$ at time levels $t_{n-1/2}$ and t_n. We then define $\mathcal{J}_2$ as the subset of points from $\mathcal{J}_1$ in which the estimated local error is larger than the tolerance at either $t_{n-1/2}$ or t_n, and for all points 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. At the boundaries of a refinement domain we will need solution values in points adjacent to that domain, which will serve as Dirichlet boundary values for the refinement domain.

For example, as illustrated in Figure 2, in a first refinement step the solution is advanced on a part of the spatial domain using the halved time step $\frac{1}{2}\Delta t_n$. For the Rosenbrock method (2.1) this will require boundary values at the time levels t_{n-1}, t_n and $t_{n-1/2}$, indicated by the open circles in the figure. At time level t_{n-1} and t_n these are available from the solution that has been computed with 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 boundary points. 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 spatial points. 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 (2.6), 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} will be made. This is to cover a situation where shortly after the last accepted time level t_n a new source or reaction term suddenly becomes active. If, after the global time step of size Δt_{n+1} has been performed, it turns out that refinement is needed everywhere, 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 at each spatial point 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, 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.

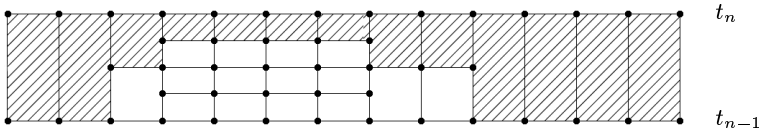

Figure 3: Time steps used for the estimation of τ^*_{n+1}.

After each level of refinement we know the points in space we already have refined (recall that for the k-th level of refinement this set of points is denoted by $\mathcal{J}_k$) and which points in space we ought to refine in the next level of refinement. Therefore, after the k-th level of refinement, for all points 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 in all spatial points 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 in the whole spatial domain.

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 space-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 spatial points is proportional to m^r with $r \geq 1$. In this report one-dimensional problems are considered in combination with the two-stage Rosenbrock 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 report 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 points in space were refined, where $m_0 = m$. Since 2^k time steps are 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

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

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 that has been done. 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 spatial points 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 points 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} spatial points 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

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

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 $\rho = (\frac{1}{2})^{1/r}$. 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 $\Delta t'_n = 2^{-l}\Delta t_n$, as C'_l. Then it is easily seen that

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

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 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 $\Delta t''_n$ on the whole spatial domain ($m_0 = m$ points) would have been performed, followed by refinement steps. Suppose that the first level of refinement would have involved m_* grid points. 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 spatial points. Therefore, it can be assumed that for the second level of refinement in the virtual step, refinement would also take place on m_1 grid points. Similarly, the k-th level of refinement can be assumed to involve m_{k-1} spatial points. 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,

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

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 points $|\mathcal{I}_1|$ 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 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 locally in space, 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 spatial region where we should refine in time. In that spatial region 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 4.

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

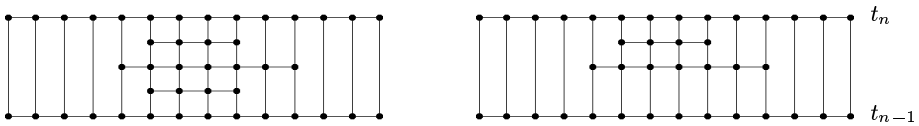

Figure 4: Example of a time slab created by the original strategy I (left) and the modified strategy II (right).

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 spatial points $m_0, m_1, \ldots, m_{s_n}$ in the levels of refinement for the n-th time slab. For the modified strategy these number of points 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

For three parabolic one-dimensional test problems we will present the numerical results of both 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 will take the number of grid points in space and time, where

the fact that with our multirate approach solutions at certain points will be computed several times is taken into account.

The results have been obtained on uniform grids in space with standard second-order differences for discretization of the spatial derivatives. Fourth-order central differences were also tried and the resulting temporal errors were very similar. For the results reported here we used quadratic interpolations at grid interfaces. Linear and cubic interpolations were also tried and the results were nearly identical, which simply indicates that the interpolation errors were 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 used.

The errors in the tables below are the maximum errors over the components at selected output times T, compared to a time-accurate ODE reference solution. Such reference solutions were computed by using very small tolerance values.

4.1 A nonlinear reaction-diffusion equation with traveling wave solution

As a first test problem we consider the reaction-diffusion equation

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

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

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

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 $L = 5$, $T = 3$. An illustration of the solution is given in Figure 5.

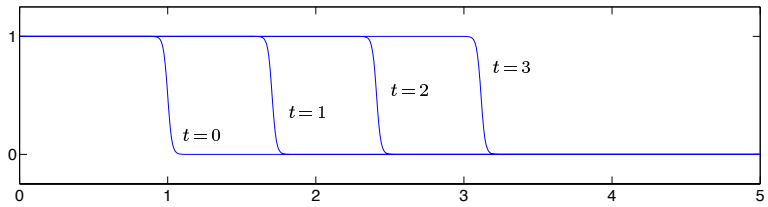

Figure 5: Traveling wave solution for problem (4.1)–(4.2) at various times.

In space we used a uniform grid of 1000 points and standard second-order differences. In order to compute the internal boundary values, at the interfaces of regions with different time steps, quadratic interpolation was performed using solutions already computed at the coarser level.

In Table 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.

Table 1: Errors and work amount for 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

For the single-rate scheme, the number of space-time points where the solution is computed is almost seven times larger than for the multirate schemes. Moreover, the error behaviour of the multirate schemes is very good. We have roughly a proportionality of the errors and tolerances, and the errors of the multirate schemes are approximately the same as for the single-rate scheme.

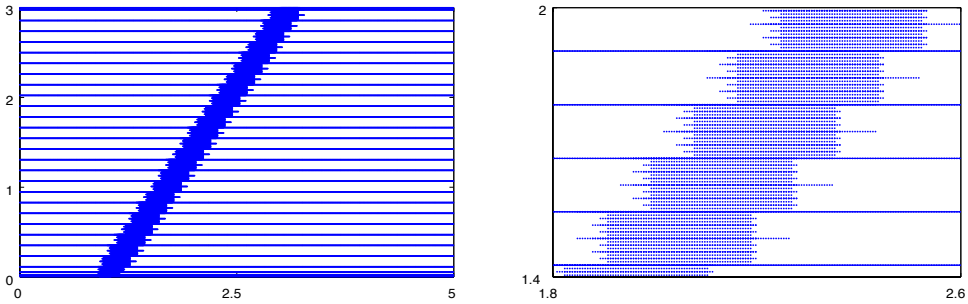

Figure 6: Space-time grid for problem (4.1)–(4.2) with strategy I. The right picture gives an enlargement for a part of the domain.

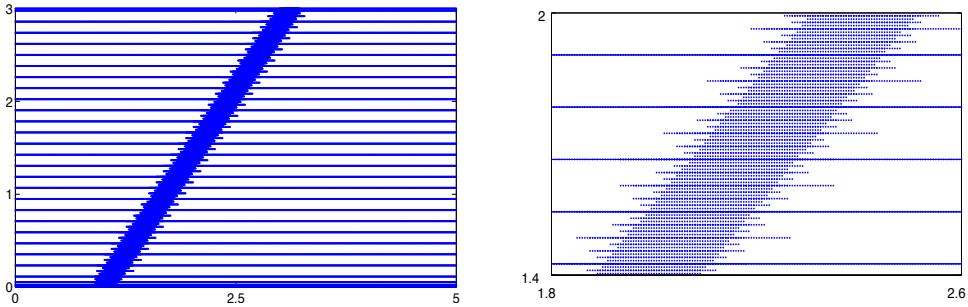

Figure 7: Space-time grid for problem (4.1)–(4.2) with strategy II. The right picture gives an enlargement for a part of the domain.

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 6 the space-time grid is shown on which the solution was calculated for strategy I with tolerance value $Tol = 5 \cdot 10^{-3}$. (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 7 shows the space-time grid for strategy II, again with $Tol = 5 \cdot 10^{-3}$.

4.2 A nonlinear problem from combustion theory

As a second test for the time slab estimation strategies we consider a one-dimensional nonlinear problem from combustion theory, taken from [1, 15],

$$u_t \;=\; d u_{xx} + f(u)\,, \qquad f(u) \;=\; \frac{R}{\alpha\delta}(1+\alpha-u)\,\mathrm{e}^{\delta(1-1/u)}\,, \tag{4.3}$$

for $t > 0$ and $0 < x < 1$. The initial- and boundary conditions are

$$u_x(0,t) = 0\,, \qquad u(1,t) = 1 \qquad \text{and} \qquad u(x,0) = 1\,. \tag{4.4}$$

For small times the solution gradually increases from unity with a 'hot spot' forming at $x = 0$. At a finite time, around $t = 0.26$ ignition occurs and the solution at $x = 0$ jumps to a value $1 + \alpha$. A flame front then forms and propagates towards $x = 1$. Around time $t = 0.29$ the front has reached the boundary of the domain and a steady state settles in. An illustration of the solution is given in Figure 8.

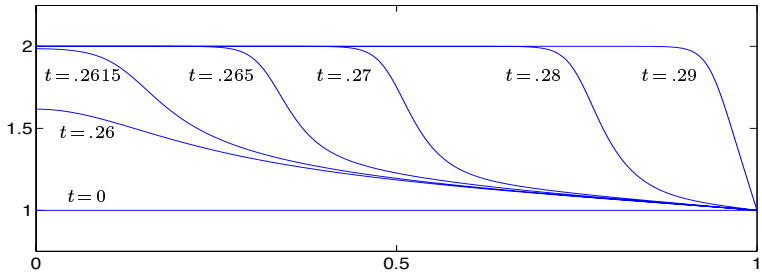

Figure 8: Solution of the combustion problem (4.3)–(4.4) at various times.

Problem (4.3) is considered with parameter values $\alpha = 1$, $\delta = 20$, $d = 1$ and $R = 5$. Discretization in space is again done with standard second-order central differences, using a uniform grid with 100 points.

This problem is locally unstable causing considerable growth of local errors over time. This growth manifests itself by producing global errors that are substantially larger than the imposed tolerance for the local error estimate. As output time for the errors we have taken $T = 0.27$. At larger times the solution rapidly approaches a steady state, and then all errors would become small, no matter what tolerances would be used. For the time interval $[0, T]$ with several tolerance values , we present in Table 2 the errors (again measured in the maximum norm with respect to the reference ODE solution) and the amount of work (number of space-time points).

For this problem we do not get a significant improvement in work with the multirate schemes. This can be explained by the fact that a large part of the work is done before ignition occurs and in that time interval the temporal variations are similar in the whole space domain. After the ignition occurs there is an improvement in work for the multirate approach, but that period is quite short for this problem. Moreover, even

Table 2: Results for problem (4.3)–(4.4).

	single-rate		multirate I		multirate II	
Tol	error	work	error	work	error	work
10^{-3}	$2.2 \cdot 10^{-1}$	17200	$3.5 \cdot 10^{-1}$	11940	$3.8 \cdot 10^{-1}$	13445
$5 \cdot 10^{-4}$	$1.3 \cdot 10^{-1}$	22300	$2.3 \cdot 10^{-1}$	15458	$2.3 \cdot 10^{-1}$	17732
10^{-4}	$3.7 \cdot 10^{-2}$	37600	$4.6 \cdot 10^{-2}$	29015	$4.8 \cdot 10^{-2}$	32870
$5 \cdot 10^{-5}$	$1.9 \cdot 10^{-2}$	52400	$1.8 \cdot 10^{-2}$	39804	$1.8 \cdot 10^{-2}$	45052
10^{-5}	$3.8 \cdot 10^{-3}$	115400	$3.0 \cdot 10^{-3}$	86015	$3.8 \cdot 10^{-3}$	98758
$5 \cdot 10^{-6}$	$1.9 \cdot 10^{-3}$	162700	$1.5 \cdot 10^{-3}$	121167	$1.9 \cdot 10^{-3}$	139016

after ignition, the front is not very steep. Refinement will therefore be performed on relatively large spatial regions.

It is also seen from the table that for this problem the multirate strategy I is slightly better than strategy II. Apparently, the extra points where the solution is computed in the first strategy are useful when the ignition occurs. It should be emphasized that the error behavior of both strategies is good: the multirate errors are close to the single-rate ones for the smaller tolerances.

Figure 9 shows the grid of space-time points in which the solution was calculated for strategy I with $Tol = 0.05$. In the enlarged picture, for time approaching $T = 0.27$, it is seen that the the placement of the space-time grid is correct when the flame is propagating, but this phase is too short to give a substantial reduction in work compared to the single-rate approach.

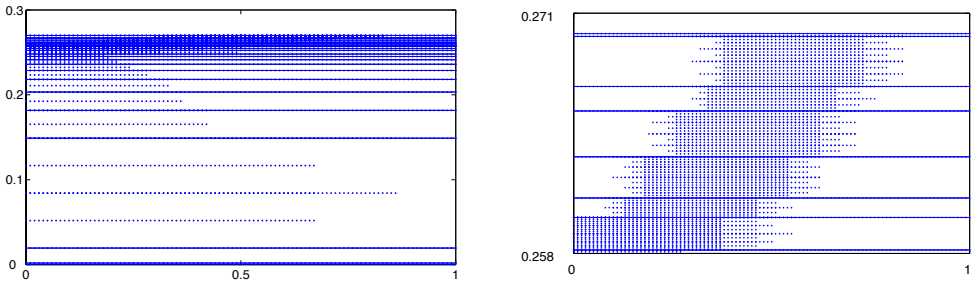

Figure 9: Space-time grid for problem (4.3)–(4.4) with strategy I. The right picture gives an enlargement for a part of the domain.

4.3 The Allen-Cahn equation

As a third test problem we consider the Allen-Cahn equation

$$u_t = \epsilon u_{xx} + u(1 - u^2)\,, \tag{4.5}$$

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

$$u_x(-1, t) = 0\,, \qquad u_x(2, t) = 0\,, \qquad u(x, 0) = u_0(x)\,, \tag{4.6}$$

where the initial profile is given by

$$u_0(x) = \begin{cases} \tanh((x+0.8)/(2\sqrt{\epsilon})) & \text{for } -1 < x < -0.8\,, \\ \tanh((0.2-x)/(2\sqrt{\epsilon})) & \text{for } -0.8 \le x < 0.28\,, \\ \tanh((x-0.36)/(2\sqrt{\epsilon})) & \text{for } 0.28 \le x < 0.4865\,, \\ \tanh((0.613-x)/(2\sqrt{\epsilon})) & \text{for } 0.4865 \le x < 0.7065\,, \\ \tanh((x-0.8)/(2\sqrt{\epsilon})) & \text{for } 0.7065 \le x < 2\,. \end{cases} \tag{4.7}$$

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

The nonlinear reaction term in (4.5) has $u = 1$ and $u = -1$ as stable equilibrium states, whereas the zero solution is an unstable equilibrium. The solution of (4.5)–(4.7) starts with three 'wells', see Figure 10. 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 $t \approx 141$. For this problem we considered $\epsilon = 9 \cdot 10^{-4}$ and we used a space grid of 400 points with second-order central differences. A time-accurate numerical solution is shown in Figure 10.

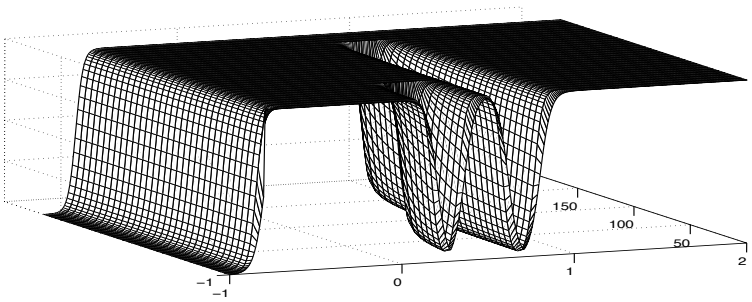

Figure 10: Evolution of the solution for problem (4.5)–(4.7).

To test the performance of the schemes, the time interval $[0, T]$ was considered with $T = 142$. At this output point, the solution is still changing in the second well; for larger times the solution becomes steady-state. In Table 3 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.

Table 3: Errors and work amount for problem (4.5)–(4.7).

	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

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

5 Conclusions

In this paper we presented self-adjusting multirate time stepping strategies for parabolic PDEs. The step size at a particular spatial grid point is determined by the local temporal variation of the solution, in contrast to the use of a single step size for the whole spatial domain 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 at inactive spatial grid points, 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 [7] and [12, 13], our multirate approach is more simple: it 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 regions 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 spatial regions, existing linear algebra solvers can still be used.

As basic time stepping methods, we used in this paper a second-order Rosenbrock method with an embedded first-order method. The multirate approach could be applied without adjustments to higher-order methods. Preliminary experiments with fourth-order Rosenbrock schemes showed again good results.

The multirate strategy discussed in this report can be extended to variable grids in space, with regridding after each completed time slab. This will also enable the use of explicit methods for hyperbolic problems. Along with this topic, we are currently investigating the use of multirate time-stepping for multi-dimensional problems. It is for 2D and 3D problems that a gain in efficiency is needed to be able to solve many problems of practical importance within a reasonable execution time.

References

[1] S. Adjerid, J.E. Flaherty, *A moving finite element method with error estimation and refinement for one-dimensional time-dependent partial differential equations.* SIAM J. Numer. Anal.23 (1986), 778–796.

[2] A. Bartel, M. Günther, *A multirate W-method for electrical networks in state space formulation.* J. Comp. Appl. Math. 147 (2002), 411–425.

[3] C. Engstler, C. Lubich, *Multirate extrapolation methods for differential equations with different time scales.* Computing 58 (1997), 173–185.

[4] C. Engstler, C. Lubich, *MUR8: A multirate extension of the eight-order Dormand-Prince method.* Appl. Numer. Math. 25 (1997), 185–192.

[5] D. Estep, M.G. Larson, R.D. Williams, *Estimating the Error of Numerical Solutions of Systems of Reaction-Diffusion Equations.* Mems. Amer. Math. Soc. 146 (2000), no. 696.

[6] C. Gear, D. Wells, *Multirate linear multistep methods.* BIT 24 (1984), 484–502.

[7] M. Günther, A. Kværnø, P. Rentrop, *Multirate partitioned Runge-Kutta methods.* BIT 41 (2001), 504–514.

[8] E. Hairer, G. Wanner, *Solving Ordinary Differential Equations II – Stiff and Differential-Algebraic Problems.* Second edition, Springer Series in Comp. Math. 14, Springer, 1996.

[9] W. Hundsdorfer, J.G. Verwer, *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations.* Springer Series in Comp. Math. 33, Springer, 2003.

[10] J. Jansson, 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), 97–105.

[12] A. Logg, *Multi-adaptive Galerkin methods for ODEs I.* SIAM J. Sci. Comput. 24 (2003), 1879–1902.

[13] A. Logg, *Multi-adaptive Galerkin methods for ODEs II. Implementation and applications.* SIAM J. Sci. Comput. 25 (2003), 1119–1141.

[14] L.F. Shampine, *Numerical Solution of Ordinary Differential Equations.* Chapman & Hall, 1994.

[15] J.G. Verwer, *Explicit Runge-Kutta methods for parabolic partial differential equations.* Appl. Numer. Math. 22 (1996), 359–379.

A Appendix: Results for fourth-order Rosenbrock methods

The same tests were performed with some fourth-order Rosenbrock methods. A number of popular methods in this class are listed in [8, Tab. IV.7.2]. Here we consider the well-known methods GRK4A and GRK4T, due to Kaps and Rentrop [Numer. Math. 33 (1979)]. The interpolation at grid interfaces was performed with cubic Hermite interpolation.

The safety factors ϑ were originally taken to be 0.9, as for the second-order methods before, but it turned out that for the combustion problem and Allen-Cahn problem a better performance was obtained with $\vartheta = 0.8$.

The results are given in the following tables. The conclusions from the experiments with ROS2 remain unchanged. In general, the results with the GRK4T method were somewhat better than for the GRK4A method. Both these fourth-order methods were more efficient than the second-order ROS2 method, even when taking into account that these methods have four stages. The 'work' in the tables is again the number of space-time points, without counting the internal computations.

These results are preliminary. The spatial discretization was still done with second-order finite differences, whereas fourth-order would be more natural to get a balance between spatial and temporal errors. The cubic Hermite interpolation could lower the accuracy of the schemes. An interpolation were the internal stage values are used could (possibly) give a better accuracy. Moreover, order reduction and boundary corrections were not investigated, and these might play a role for the fourth-order methods.

Table 4: Errors and work for problem (4.1)–(4.2) with GRK4A ($\vartheta = 0.9$).

	single-rate		multirate I		multirate II	
tol	error	work	error	work	error	work
10^{-2}	0.089	127127	0.068	49460	0.055	37261
$5 \cdot 10^{-3}$	0.05	154154	0.0827	54817	0.0804	43871
10^{-3}	0.0099	249249	0.0079	74012	0.0095	65457
$5 \cdot 10^{-4}$	0.0046	308308	0.0051	82536	0.0044	65205
10^{-4}	$7.86 \cdot 10^{-4}$	497497	$7.84 \cdot 10^{-4}$	114242	$7.67 \cdot 10^{-4}$	96879
$5 \cdot 10^{-5}$	$3.69 \cdot 10^{-4}$	606606	$3.67 \cdot 10^{-4}$	126019	$3.16 \cdot 10^{-4}$	114333
10^{-5}	$6.53 \cdot 10^{-5}$	951951	$5.75 \cdot 10^{-5}$	177306	$5.16 \cdot 10^{-5}$	163467

Table 5: Errors and work for problem (4.1)–(4.2) with GRK4T ($\vartheta = 0.9$).

	single-rate		multirate I		multirate II	
tol	error	work	error	work	error	work
10^{-2}	0.022	147147	0.046	45623	0.030	34827
$5 \cdot 10^{-3}$	0.012	174174	0.029	49676	0.028	36279
10^{-3}	0.0027	261261	0.0031	64600	0.0034	57292
$5 \cdot 10^{-4}$	0.0014	311311	0.0018	72489	0.0017	66105
10^{-4}	$3.07 \cdot 10^{-4}$	470470	$3.76 \cdot 10^{-4}$	105769	$3.64 \cdot 10^{-4}$	94843
$5 \cdot 10^{-5}$	$1.55 \cdot 10^{-4}$	561561	$1.86 \cdot 10^{-4}$	118588	$1.80 \cdot 10^{-4}$	108611
10^{-5}	$3.18 \cdot 10^{-5}$	846846	$3.46 \cdot 10^{-5}$	164325	$3.10 \cdot 10^{-5}$	148812

Table 6: Errors and work for problem (4.3)–(4.4) with GRK4A ($\vartheta = 0.8$).

	single-rate		multirate I		multirate II	
tol	error	work	error	work	error	work
10^{-3}	0.0647	6300	0.101	5218	0.101	4914
$5 \cdot 10^{-4}$	0.0554	7500	0.0605	5576	0.0632	5042
10^{-4}	0.0225	11200	0.0173	9086	0.0141	8144
$5 \cdot 10^{-5}$	0.0113	13000	0.0121	10558	0.00799	9928
10^{-5}	$2.73 \cdot 10^{-3}$	18400	$2.71 \cdot 10^{-3}$	16598	$1.55 \cdot 10^{-4}$	14090
$5 \cdot 10^{-6}$	$1.73 \cdot 10^{-3}$	20600	$1.38 \cdot 10^{-3}$	16660	$1.39 \cdot 10^{-3}$	16420

Table 7: Errors and work for (4.3)–(4.4) with GRK4T ($\vartheta = 0.8$).

	single-rate		multirate I		multirate II	
tol	error	work	error	work	error	work
10^{-3}	0.00914	6900	0.0125	6398	0.00964	5482
$5 \cdot 10^{-4}$	0.00612	8300	0.00786	8320	0.00803	7472
10^{-4}	0.00153	11600	0.00112	11886	0.00209	11758
$5 \cdot 10^{-5}$	$8.80 \cdot 10^{-4}$	13100	$8.27 \cdot 10^{-4}$	14942	$8.95 \cdot 10^{-4}$	14298
10^{-5}	$2.77 \cdot 10^{-4}$	16700	$2.76 \cdot 10^{-4}$	20998	$2.80 \cdot 10^{-4}$	18444
$5 \cdot 10^{-6}$	$1.22 \cdot 10^{-4}$	19800	$1.37 \cdot 10^{-4}$	22417	$1.47 \cdot 10^{-4}$	23573

Table 8: Errors and work for problem (4.5)–(4.7) with GRK4A ($\vartheta = 0.8$).

	single-rate		multirate I		multirate II	
tol	error	work	error	work	error	work
10^{-3}	0.172	30476	0.109	15145	0.154	10699
$5 \cdot 10^{-4}$	0.0569	33283	0.0359	17431	0.0472	15041
10^{-4}	0.00469	51328	0.00583	26019	0.00588	23863
$5 \cdot 10^{-5}$	0.00177	64160	0.00194	36357	0.00192	30186
10^{-5}	$2.11 \cdot 10^{-4}$	103458	$2.59 \cdot 10^{-4}$	60504	$2.85 \cdot 10^{-4}$	49796
$5 \cdot 10^{-6}$	$8.83 \cdot 10^{-5}$	127518	$8.90 \cdot 10^{-5}$	69615	$8.65 \cdot 10^{-5}$	66138

Table 9: Errors and work for problem (4.5)–(4.7) with GRK4T ($\vartheta = 0.8$).

	single-rate		multirate I		multirate II	
tol	error	work	error	work	error	work
10^{-3}	0.0321	30476	0.0211	18790	0.0147	17715
$5 \cdot 10^{-4}$	0.0127	34887	0.016	25784	0.0115	14106
10^{-4}	0.00132	48120	0.0014	27202	0.00127	21184
$5 \cdot 10^{-5}$	$5.58 \cdot 10^{-4}$	58145	$5.075 \cdot 10^{-4}$	38475	$5.95 \cdot 10^{-4}$	29075
10^{-5}	$8.39 \cdot 10^{-5}$	88621	$9.31 \cdot 10^{-5}$	52189	$9.02 \cdot 10^{-5}$	47636
$5 \cdot 10^{-6}$	$3.81 \cdot 10^{-5}$	107067	$4.17 \cdot 10^{-5}$	72206	$4.33 \cdot 10^{-5}$	59357