
MOX-Report No. 08/2018

# A self adjusting multirate algorithm based on the TR-BDF2 method

Bonaventura, L.; Casella, F.; Delpopolo, L.; Ranade, A.;

MOX, Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

mox-dmat@polimi.it http://mox.polimi.it

# A self adjusting multirate algorithm based on the TR-BDF2 method

Luca Bonaventura[(1)], Francesco Casella[(2)]
Ludovica Delpopolo[(1)], Akshay Ranade[(3)]

January 27, 2018

[(1)] MOX – Modelling and Scientific Computing
Dipartimento di Matematica, Politecnico di Milano
Via Bonardi 9, 20133 Milano, Italy
luca.bonaventura@polimi.it, ludovica.delpopolo@polimi.it

[(2)] Dipartimento Elettronica, Informazione e Bioingegneria
Politecnico di Milano
Via Ponzio 34/35, 20133 Milano, Italy
francesco.casella@polimi.it

[(3)] Department of Computer Science
University College Cork
Western Road
Cork, Ireland
akshay.ranade@cs.ucc.ie

**Keywords**: Multirate methods, stiff ODE problems, Implicit Runge Kutta methods, SDIRK methods

**AMS Subject Classification**: 65L04, 65L05, 65L07, 65M12, 65M20

**Abstract**

We propose a self adjusting multirate method based on the TR-BDF2 solver. The potential advantages of using TR-BDF2 as the key component of a multirate framework are highlighted. A linear stability analysis of the resulting approach is presented and the stability features of the resulting algorithm are analysed. The analysis framework is completely general and allows to study along the same lines the stability of self adjusting multirate methods based on a generic one step solver. A number of numerical experiments demonstrate the efficiency and accuracy of the resulting approach also the time discretization of hyperbolic partial differential equations.

# 1 Introduction

The concept of multirate methods was first proposed in [17] and a great attention has been devoted to these methods, see e.g. [1], [8] and the more comprehensive review reported in [16]. The basic idea of multirate methods is to integrate each component using a time step that is the most appropriate to the timescale of that component. Slowly varying components are integrated with larger time steps, while smaller time steps are used for fast components only. Thus, multirate methods can avoid a significant amount of the computation that is necessary in the standard, single rate approach. In other words, in the multirate approach, the most appropriate time resolution is employed for each state variable of the system. In the context of multirate algorithms, the faster components are often referred to as active and the slow ones are called instead latent. It is also to be remarked that conceptually similar approaches have been introduced in the literature on numerical weather prediction, see e.g. [13], for the purpose of avoiding the time step stability restrictions related to fast propagating acoustic waves.

The specific way in which the partitioning is performed and the degree of coupling allowed between latent and active components determines the efficiency and the stability of the resulting methods. The stability of multirate methods has been investigated, among others, in [2] and [10], while the stability of split-explicit methods has been analyzed in [3], [20]. In general, it is seen that the stability region of the multirate methods decreases as the strength of the coupling between the partitions or the stiffness of the system increases. In spite of this, there is ample evidence in the literature that multirate algorithms can offer a promising path for the numerical integration of large systems with widely varying time scales or with highly localised periods of activity. In order to address the potential problems of multirate methods, the so-called compound step multirate methods have been proposed in [10]. A self-adjusting, recursive time stepping strategy has been proposed in [19]. In this kind of approach, a tentative global step is first taken for all components, using a robust, unconditionally stable method. The time step is then reduced only for those components for which some local error estimator is greater than the specified tolerance. In this way, automatic detection of the latent components is achieved. In [18], an algorithm based on this recursive strategy was shown to be more stable than the original compound step algorithm, while a stability analysis of a self-adjusting algorithm based on the $\theta-$method has been proposed in [12].

In this paper, we propose a self-adjusting multirate method based on the same concept as [19], that relies on the TR-BDF2 method

as fundamental single rate solver. The TR-BDF2 method has been originally introduced in [4] and more thoroughly analyzed in [11]. It is a second order, one step, L-stable implicit method endowed with a number of interesting properties, as discussed in [11]. For example, it allows for cheap error estimation via an embedded third order method, it can provide continuous output via a cubic Hermite interpolant and it has an explicit second order companion with which it can be combined to form a second order, implicit-explicit additive Runge Kutta method. Due to its favourable properties, it has been recently applied for efficient discretization of high order finite element methods for numerical weather forecasting in [9], [22]. Its appealing features also make it an interesting candidate for multirate approaches targeted at large scale stiff problems, resulting either from the simulation industrial systems or from the spatial discretization of partial differential equations. In particular, the cubic Hermite interpolant can be employed to maintain higher accuracy also for stiff problems in the framework of a multirate approach. The proposed self-adjusting multirate TR-BDF2 method is equipped here with a partitioning and time step selection criterion based on the technique proposed in [7].

The stability of the method has been analyzed in the framework of the classical linear model problem. Even though no complete theoretical results can be achieved, numerical computation of the stability function norm for a range of relevant model problems shows that the method has good stability properties. This is true not only for contractive problems with strictly dissipative behaviour, but also for problems with purely imaginary eigenvalues, which suggests that the method could be advantageous also for the application to hyperbolic PDEs and structural mechanics problems. The numerical results obtained on several benchmarks show that the application of the proposed method leads to significant efficiency gains, that are analogous to those achieved by other self-adjusting multirate approaches with respect to their single rate counterparts. In particular, the method appears to perform well when applied to the time discretization of nonlinear, hyperbolic partial differential equations, allowing to achieve automatic detection of complex localized phenomena such as shock waves and significant reductions in computational cost.

In section 2, the self-adjusting implicit multirate approach is described. The time step refinement criterion and the resulting partitioning principle between active and latent components are presented in section 3. The TR-BDF2 method is reviewed in section 4. The interpolation procedures we have considered are described in section 5. A linear stability analysis is then presented in section 6. The approach proposed for such analysis is rather general and allows to study

generic multirate methods based on any one step solver and interpolation procedure. Numerical simulations are presented in section 7, while some conclusions and perspectives for future work are presented in 8.

# 2 A self adjusting implicit multirate approach

We focus on multirate methods for the solution of the Cauchy initial value problem

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y}) \qquad \mathbf{y}(0) = \mathbf{y}_0 \in \mathbf{R}^m, \qquad t \in [0, T]. \tag{2.1}$$

We consider time discretizations associated to discrete time levels $t_n,\ n = 0, \ldots, N$ such that $h_n = t_{n+1} - t_n$ and we will denote by $\mathbf{u}_n$ the numerical approximation of $\mathbf{y}(t^n)$. We will also denote by $\mathbf{u}^{n+1} = \mathcal{S}(\mathbf{u}^n, h_n)$ the implicitly defined operator $\mathcal{S} : \mathbf{R}^m \to \mathbf{R}^m$ whose application is equivalent to the computation of one step of size $h_n$ of a given single step method. While here only implicit methods will be considered, the whole framework can be extended to explicit and IMEX methods. Notice that, if $\mathbf{P}$ is the projector onto a linear subspace $\mathcal{V} \subset \mathbf{R}^m$ with dimension $p < m$, the operator $\mathcal{S}^{\mathcal{V}} : \mathbf{R}^p \times \mathbf{R}^{m-p} \to \mathbf{R}^p$ that represents the solution of the subsystem obtained freezing the components of the unknown belonging to $\mathcal{V}^{\perp}$ to the value $\mathbf{z} \in \mathbf{R}^{m-p}$ can be defined by $\mathbf{y} = \mathcal{S}^{\mathcal{V}}(\mathbf{x}, \mathbf{z}, h_n) = \mathbf{P}\mathcal{S}(\mathbf{x} \oplus \mathbf{z}, h_n)$. Furthermore, we will denote by $\mathbf{Q}(\mathbf{u}^{n+1}, \mathbf{u}^n, \zeta)$ an interpolation operator, that provides an approximation of the numerical solution at intermediate time levels $t_n + \zeta$, where $\zeta \in [0, h_n]$. Linear interpolation is often employed, but, for multistage methods, knowledge of the intermediate stages also allows the application of more accurate interpolation procedures without substantially increasing the computational cost.

We propose a self-adjusting multirate algorithm that is a generalization of the method introduced in [19], in which the choice of the local time step is left completely to the time step selection criterion. The multirate algorithm can be described as follows.

1) Perform a tentative global (or macro) time step of size $h_n$ with the standard single rate method and compute $\hat{\mathbf{u}}^{n+1} = \mathcal{S}(\mathbf{u}^n, h_n)$. The specific way in which error control mechanisms are applied to the choice of the global and local time steps are crucial for the efficiency of the algorithm and will be discussed in detail in section 3.

2) Compute the error estimator and partition the state space into active and latent variables, based on the value of the error estimate. The projection onto the subspace $\mathcal{V}_0$ of the active variables is denoted by $\mathbf{P}_n = \mathbf{P}_n^{(0)}$, while the projection onto the complementary subspace will be denoted by $\bar{\mathbf{P}}_n^{(0)}$. Define $\bar{\mathbf{P}}_n^{(0)}\mathbf{u}^{n+1} = \bar{\mathbf{P}}_n^{(0)}\hat{\mathbf{u}}^{n+1}$ as well as $\mathbf{u}^{n,0} = \mathbf{u}^n$ and $t_{n,0} = t_n$.

3) For $k \geq 1$, choose a local (or micro) time step $h_n^{(k-1)}$ for the active variables, based on the value of the error estimator. Set $t_{n,k} = \min\{t_{n,k-1} + h_n^{(k-1)}, t_{n+1}\}$.

   3.1) Update the latent variables by interpolation

$$\bar{\mathbf{P}}_n^{(k-1)}\mathbf{u}^{n,k} = \mathbf{Q}(\bar{\mathbf{P}}_n^{(k-1)}\mathbf{u}^{n+1}, \bar{\mathbf{P}}_n^{(k-1)}\mathbf{u}^{n,k-1}, h_n^{(k-1)}).$$

   3.2) Update the active variables by computing

$$\mathbf{P}_n^{(k-1)}\mathbf{u}^{n,k} = \mathcal{S}^{\mathcal{V}_{k-1}}(\mathbf{P}_n^{(k-1)}\mathbf{u}^{n,k-1}, \bar{\mathbf{P}}_n^{(k-1)}\mathbf{u}^{n,k}, h_n^{(k-1)}).$$

   3.3) Compute the error estimator for the active variables only and partition again $\mathcal{V}_{k-1}$ into latent and active variables. Denote by $\mathcal{V}_k \subset \mathcal{V}_{k-1}$ the new subspace of active variables and by $\mathbf{P}_n^{(k)}$ the corresponding projection.

   3.4) Repeat 3.1) - 3.3) until $t_{n,k} = t_{n+1}$.

Clearly, the effectiveness of the above procedure depends in a crucial way on the accuracy and stability of the basic ODE solver $\mathcal{S}$, as well as on the time step refinement and partitioning criterion. Furthermore, as well known in multirate methods, unconditional stability of the solver $\mathcal{S}$ does not necessarily entail that the same property holds for the derived multirate solver. The refinement and partitioning criterion will be described in detail in section 3. Here, we only remark that, as it will be shown by the numerical experiments in section 7, the approach outlined above is able to reduce significantly the computational cost with respect to the equivalent single rate methods, without a major reduction in stability for most of the envisaged applications.

# 3 The time step refinement and partitioning criterion

We now describe in detail the time step refinement and partitioning strategy that we have used in the multirate algorithm described in section 2. Our approach is based on the strategy proposed for an explicit Runge Kutta multirate method in [7], where the time steps

for refinement are obtained from the error estimates of the global step. The user specified tolerance plays an important role in the partitioning of the system. In [7], a simple absolute error tolerance was considered. However, in most engineering system simulations, whenever the typical values of different components can vary greatly, the tolerance used is in general a combination of absolute and relative error tolerances, see e.g. [21]. We have thus extended the strategy proposed in [7] in order to employ such a combination of absolute and relative error tolerances.

More specifically, denote by $\tau_r$, $\tau_a$ the user defined error tolerances for relative and absolute errors, respectively. Furthermore, assume that the tentative global step $\hat{\mathbf{u}}^{n+1} = \mathcal{S}(\mathbf{u}^n, h_n^{(0)})$ has been computed and that an error estimator $\boldsymbol{\epsilon}^{n+1,0}$ is available. The first task of the time step selection criterion is to asses whether the global time step $h_n^{(0)}$ has been properly chosen. Denoting by $\epsilon_i^{n+1,0}, \hat{u}_i^{n+1}, i = 1, \ldots, m$ the $i-$th components of the error estimator and of the tentative global step numerical solution, respectively, we define a vector $\boldsymbol{\eta}^{n+1,0}$ of normalized errors with components

$$\eta_i^{n+1,0} = \frac{\epsilon_i^{n+1,0}}{\tau_r |\hat{u}_i^{n+1}| + \tau_a}.$$

Since clearly the condition $\max_i \eta_i^{n+1,0} \leq 1$ has to be enforced, before proceeding to the partitioning into active and latent variables, this condition is checked and the global step is repeated with a smaller value of $h_n^{(0)}$ whenever it is violated. Numerical experiments have shown that, while an increase of efficiency with respect to the single rate version of the method is always achieved, independently of the choice of the tentative step, the greatest improvements are only possible if the global time step does not have to be repeated too often.

Once the condition $\max_i \eta_i^{n+1,0} \leq 1$ is satisfied, the set of indices of the components flagged for the first level of time step refinement is identified by

$$\mathcal{A}^{n+1,0} = \{i : \eta_i^{n+1,0} > \delta \|\boldsymbol{\eta}^{n+1,0}\|_\infty\}, \tag{3.1}$$

where $\delta \in (0, 1)$ is a user defined coefficient. The smaller the value of $\delta$, the larger is the fraction of components marked for refinement. Notice that, if $\delta$ is set to unity, the algorithm effectively operates in single rate mode. For each iteration $k = 1, \ldots, k_{max}$ of the algorithm described in section 2, active variable sets $\mathcal{A}^{n+1,k}$ are then defined analogously. In each iteration $k = 0, \ldots, k_{max}$, the time step $h_n^{(k)}$ is chosen according to the standard criterion (see e.g. [14], [19])

$$h_n^{(k)} = \nu \min_{j \in \mathcal{A}^{n+1,k}} \left( \frac{\tau_r |\hat{u}^{n+1}|_j + \tau_a}{\epsilon_i^{n+1,k}} \right)^{\frac{1}{p+1}}, \tag{3.2}$$

where $p$ is the convergence order of the solver $\mathcal{S}$ and $\nu$ is an user defined safety parameter.

# 4 The single rate TR-BDF2 method

The TR-BDF2 method was originally proposed in [4]. It is a composite one step, two stage method, consisting of one stage of the trapezoidal method followed by another of the BDF2 method. The stages are so adjusted that both the trapezoidal and the BDF-2 stages use the same Jacobian matrix. This composite method has been reinterpreted in [11] as a one step Diagonally Implicit Runge Kutta (DIRK) method with two internal stages The TR-BDF2 method is also in some sense the optimal method among all 3-stage DIRK methods, owing to the following properties:

1) it is first same as last (FSAL), so that only two implicit stages need to be evaluated;
2) the Jacobian matrix for both the stages is the same;
3) it has an embedded third order companion that allows for a cheap error estimator;
4) the method is strongly S-Stable;
5) all the stages are evaluated within the time step;
6) it is endowed with a cubic Hermite interpolation algorithm for dense output that yields globally $C^1$ continuous trajectories.

Features 3), 5) and 6) will play an important role in the multirate method proposed here. Furthermore, as shown in [9], the method has an explicit second order companion with which it can be combined to form a second order implicit-explicit additive Runge Kutta method. Due to its favourable properties, it has been recently applied for efficient discretization of high order finite element methods for numerical weather forecasting in [9], [22].

The TR-BDF2 method, considered as a composite method consisting of a step with the trapezoidal method followed by a step of the BDF2 method, can be written as

$$
\begin{aligned}
\mathbf{u}^{n+\gamma} &= \mathbf{u}^n + \frac{\gamma h_n}{2}\left(\mathbf{f}(t,\mathbf{u}^n) + \mathbf{f}(t,\mathbf{u}^{n+\gamma})\right) \\
\mathbf{u}^{n+1} &= \frac{1}{\gamma(2-\gamma)}\mathbf{u}^{n+\gamma} - \frac{(1-\gamma)^2}{\gamma(2-\gamma)}\mathbf{u}^n + \frac{1-\gamma}{2-\gamma}h_n\mathbf{f}(t,\mathbf{u}^{n+\gamma})
\end{aligned}
\tag{4.1}
$$

The TR-BDF2 method viewed as a DIRK method has the following Butcher tableau:

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| $\gamma$ | $d$ | $d$ | 0 |
| 1 | $w$ | $w$ | $d$ |
| | $w$ | $w$ | $d$ |
| * | $\frac{(1-w)}{3}$ | $\frac{(3w+1)}{3}$ | $\frac{d}{3}$ |

where $\gamma = 2-\sqrt{2}$, $d = \frac{\gamma}{2}$ and $w = \frac{\sqrt{2}}{4}$ and the row * corresponds to the embedded third order method that can be used to build a convenient error estimator. Here, the value $\gamma = 2-\sqrt{2}$ is chosen for the two stages to have the same Jacobian matrix, as well as in order to achieve L-stability, as it will be shown shortly. It can be seen that the method has the First Same As Last (FSAL) property, i.e., the first stage of any step is the same as the last stage of the previous step. Thus, in any step, the first explicit stage need not be computed.

The implementation of the two implicit stages is done as suggested in [11]. The two stages according to the Butcher tableau are given by equation (4.1). Instead of iterating on the variable $\mathbf{u}^{n+1}$, we define another variable $\mathbf{z} = h\mathbf{f}(t,\mathbf{u})$ and solve for this variable by iteration. Thus, for the first implicit stage, we take $\mathbf{u}^{n+\gamma,k} = \mathbf{u}^n + d\mathbf{z}^n + d\mathbf{z}^{n+\gamma,k}$ and $\mathbf{z}^{n+\gamma,k}$ is computed by Newton iterations as

$$
\begin{aligned}
(\mathbf{I} - dh\mathbf{J}^n)\boldsymbol{\Delta}^k &= h_n\mathbf{f}(t_{n+\gamma}, \mathbf{y}^k_{n+\gamma}) - \mathbf{z}^{n+\gamma,k} \\
\mathbf{z}^{n+\gamma,k+1} &= \mathbf{z}^{n+\gamma} + \boldsymbol{\Delta}^k,
\end{aligned}
$$

where $\mathbf{J}^n = \mathbf{J}(t_n, \mathbf{u}_n)$ denotes the Jacobian of $\mathbf{f}$. Similarly, for the second implicit stage we take $\mathbf{u}^k_{n+1} = \mathbf{u}_n + w\mathbf{z}^n + w\mathbf{z}^{n+\gamma} + d\mathbf{z}^{n+1,k}$ and the Newton iteration is given by

$$
\begin{aligned}
(\mathbf{I} - dh\mathbf{J}^n)\boldsymbol{\Delta}^k &= h\mathbf{f}(t_{n+\gamma}, \mathbf{y}^k_{n+\gamma}) - \mathbf{z}^{n+\gamma,k} \\
\mathbf{z}^{n+1,k+1} &= \mathbf{z}^{n+1} + \boldsymbol{\Delta}^k.
\end{aligned}
$$

The reason for doing so is that if the problem is stiff, a function evaluation will amplify the numerical error in the stiff components. Iterating on $\mathbf{z}$, however, ensures that it is computed to within the specified tolerance. For a more detailed discussion the interested reader is referred to [11]. An important point to be noted is that, although the TR-BDF2 method is L-stable, its third order companion formula is not. Therefore, the error estimate at time level $n+1$, given by

$$\boldsymbol{\epsilon}^{n+1,*} = (b_1^* - b_1)\mathbf{z}^n + (b_2^* - b_2)\mathbf{z}_{n+\gamma} + (b_3^* - b_3)\mathbf{z}_{n+1}$$

cannot be expected to be accurate for stiff problems. To overcome this problem, it is suggested in [11] to modify the error estimate by considering as error estimator the quantity $\boldsymbol{\epsilon}_{n+1}$ defined as the solution of the linear system

$$(\mathbf{I} - dh\mathbf{J}^n)\boldsymbol{\epsilon}^{n+1} = \boldsymbol{\epsilon}^{n+1,*}. \tag{4.2}$$

This modification of the error estimate allows to improve it for stiff components, while preserving its accuracy in the limit of small time steps. Notice that the stability function of the TR-BDF2 method is given by

$$R(z) = \frac{[1 + (1-\gamma)^2]z + 2(2-\gamma)}{z^2(1-\gamma)\gamma + z(\gamma^2 - 2) + 2(2-\gamma)}, \tag{4.3}$$

whence it can be seen that the method is L-stable for $\gamma = 2 - \sqrt{2}$.

# 5 The interpolation procedures

An essential component of any multirate algorithm is the procedure employed to reconstruct the values of the latent variables at those intermediate time levels for which only the active variables are computed. Self-adjusting multirate procedures based on implicit methods, such as the one proposed in [19] and that presented in this paper, can avoid the use of extrapolation, thus increasing their overall stability. The simplest and most commonly used procedure is linear interpolation, that is defined for $\zeta \in [0, h_n]$ as

$$\mathbf{Q}(\mathbf{u}^{n+1}, \mathbf{u}^n, \zeta) = \frac{\zeta}{h_n}\mathbf{u}^{n+1} + \frac{(h_n - \zeta)}{h_n}\mathbf{u}^n. \tag{5.1}$$

In case a multi-stage solver is employed, higher order interpolation can also be employed at no extra cost. Assuming that a sufficiently accurate approximation of the solution $\mathbf{u}^{n+\lambda}$ is available at time $t_{n+\lambda} \in$

$(t_n, t_{n+1})$ and setting $h_\lambda = t_{n+\lambda} - t_n$, quadratic Lagrange interpolation can then be written as

$$\begin{aligned} \mathbf{Q}(\mathbf{u}^{n+1}, \mathbf{u}^n, \zeta) &= \frac{(\zeta - h_\lambda)\zeta}{h_n(h_n - h_\lambda)}\mathbf{u}^{n+1} + \frac{\zeta(\zeta - h_n)}{(h_\lambda - h_n)h_\lambda}\mathbf{u}^{n+\lambda} \\ &+ \frac{(h_\lambda - \zeta)(h_n - \zeta)}{h_\lambda h_n}\mathbf{u}^n. \end{aligned} \tag{5.2}$$

An interesting feature of the TR-BDF2 method in the multirate framework is that the method, as explained in [11], is endowed with a cubic Hermite, globally $C^1$ interpolation algorithm for dense output. Using the notation of section 4, the Hermite cubic interpolant can be defined as

$$\begin{aligned} \mathbf{Q}(\mathbf{u}^{n+1}, \mathbf{u}^n, \zeta) &= (\boldsymbol{\alpha}_3 - 2\boldsymbol{\alpha}_2)\beta^3(\zeta) \\ &+ (3\boldsymbol{\alpha}_2 - \boldsymbol{\alpha}_3)\beta^2(\zeta) + \boldsymbol{\alpha}_1\beta(\zeta) + \boldsymbol{\alpha}_0, \end{aligned} \tag{5.3}$$

where, for $0 \le \zeta \le \gamma h_n$ one has

$$\boldsymbol{\alpha}_0 = \mathbf{u}_n, \quad \boldsymbol{\alpha}_1 = \gamma \mathbf{z}_n, \quad \boldsymbol{\alpha}_2 = \mathbf{u}_{n+\gamma} - \mathbf{u}_n - \gamma \boldsymbol{z}_n,$$
$$\boldsymbol{\alpha}_3 = \gamma(\boldsymbol{z}_{n+\gamma} - \boldsymbol{z}_n), \qquad \beta(\zeta) = \frac{\zeta}{\gamma h_n}$$

and for $\gamma h_n \le \zeta \le h_n$ one has instead

$$\boldsymbol{\alpha}_0 = \mathbf{u}_{n+\gamma},\ \boldsymbol{\alpha}_1 = (1-\gamma)\mathbf{z}_{n+\gamma},\ \boldsymbol{\alpha}_2 = \mathbf{u}_{n+1} - \mathbf{u}_{n+\gamma} - \boldsymbol{\alpha}_1,$$
$$\boldsymbol{\alpha}_3 = (1-\gamma)(\mathbf{z}_{n+1} - \mathbf{z}_{n+\gamma}) \qquad \beta(\zeta) = \frac{\zeta - \gamma h_n}{(1-\gamma)h_n}.$$

This interpolant allows to have accurate approximations of the latent variables without extra computational cost or memory storage requirements, since it employs the intermediate stages of the TR-BDF2 method in order to achieve higher order accuracy. This contributes to making the TR-BDF2 method an attractive choice as the basis for a multirate approach.

# 6 Linear stability analysis

In this section, we will study the linear stability of the multirate algorithm outlined in the previous sections in the case of a generic one step solver. Following the approach employed in [12] for the $\theta-$method, we will consider the special case in which the macro time step is constant and denoted by $h$ and a single refinement is carried out, with the refined time step used for the active components given by $h/2$. As a result, a single subspace of active variables is introduced,

$\mathcal{V} \subset \mathbf{R}^m$, with dimension $p < m$. As in section 2, $\mathbf{P}$ will denote the projector onto this subspace, while $\mathbf{P}^\perp$ will denote the corresponding projector onto the complementary subspace $\mathcal{V}^\perp$ of the latent variables. We will also denote by $\mathbf{E}, \mathbf{E}^\perp$ the corresponding natural embedding operators of $\mathcal{V}, \mathcal{V}^\perp$ into $\mathbf{R}^m$. As a result, the operators $\mathbf{\Pi} = \mathbf{EP}$ and $\mathbf{\Pi}^\perp = \mathbf{E}^\perp\mathbf{P}^\perp$ represent the zeroing of the components of a given vector belonging to $\mathcal{V}, \mathcal{V}^\perp$ respectively. It is to be remarked that this special setting is a major simplification with respect to the generality of the algorithm described in section 2, so that the results of the analysis may not fully explain the behaviour of the algorithm in practical applications. In particular, large values of the stability function norm in this context do not necessarily imply that the method is unable to achieve the desired accuracy when applied in its most general form.

We now consider the linear problem $\mathbf{y}' = \mathbf{f}(t, \mathbf{y}) = \mathbf{A}\mathbf{y}$ and we set $h\mathbf{A} = \mathbf{Z}$. We assume that the amplification matrix of the basic one step solver can be written as

$$\mathbf{R} = \mathbf{R}(\mathbf{Z}) = \mathbf{D}(\mathbf{Z})^{-1}\mathbf{N}(\mathbf{Z}) = \mathbf{D}^{-1}\mathbf{N},$$

where $\mathbf{D}, \mathbf{N}$ are polynomials in $\mathbf{Z}$. Notice that this includes most standard one step solvers, either explicit or implicit. We will also use the notations

$$\mathbf{D}_{1/2} = \mathbf{D}(\mathbf{Z}/2) \qquad \mathbf{N}_{1/2} = \mathbf{N}(\mathbf{Z}/2).$$

Notice that, in the special case considered here, the interpolation operator can be represented by the application to the data of a matrix $\mathbf{Q}_{1/2}$ whose entries only depend on $h$ and on the amplification matrix of the single rate method employed. For example, the linear interpolant (5.1) is represented in this case by the matrix

$$\mathbf{Q}_{1/2} = \frac{1}{2}\left(\mathbf{I} + \mathbf{R}\right),$$

while for the cubic Hermite interpolant (5.4) one has

$$\mathbf{Q}_{1/2} = \mathbf{I} + \beta\gamma\mathbf{Z} + \beta^2\mathbf{F} + \beta^3\mathbf{G},$$

where $\beta = 1/(2\gamma)$ and

$$\mathbf{F} = 3(\mathbf{R}_\gamma - \mathbf{I} - \gamma\mathbf{Z}) - \gamma\mathbf{Z}(\mathbf{R}_\gamma - \mathbf{I}) \qquad \mathbf{G} = \gamma\mathbf{Z}(\mathbf{R}_\gamma - \mathbf{I}) - 2(\mathbf{R}_\gamma - \mathbf{I} - \gamma\mathbf{Z}),$$

$$\mathbf{R}_\gamma = \left(\mathbf{I} - \frac{\gamma}{2}\mathbf{Z}\right)^{-1}\left(\mathbf{I} + \frac{\gamma}{2}\mathbf{Z}\right).$$

Given these definitions, one step of the self-adjusting multirate algorithm can be rewritten in this special setting as

$$
\begin{aligned}
&\mathbf{P}^{\perp}\mathbf{u}^{n+1} = \mathbf{P}^{\perp}\mathbf{R}\mathbf{u}^{n} = \mathbf{P}^{\perp}\mathbf{u}^{n,2} \qquad \mathbf{P}^{\perp}\mathbf{u}^{n,1} = \mathbf{P}^{\perp}\mathbf{Q}_{1/2}\mathbf{u}^{n} \\
&\mathbf{P}\mathbf{D}_{1/2}\left(\mathbf{\Pi}\mathbf{u}^{n,1} + \mathbf{\Pi}^{\perp}\mathbf{Q}_{1/2}\mathbf{u}^{n}\right) = \mathbf{P}\mathbf{N}_{1/2}\mathbf{u}^{n} \qquad (6.1) \\
&\mathbf{P}\mathbf{D}_{1/2}\left(\mathbf{\Pi}\mathbf{u}^{n,2} + \mathbf{\Pi}^{\perp}\mathbf{R}\mathbf{u}^{n}\right) = \mathbf{P}\mathbf{N}_{1/2}(\mathbf{\Pi}\mathbf{u}^{n,1} + \mathbf{\Pi}^{\perp}\mathbf{Q}_{1/2}\mathbf{u}^{n}).
\end{aligned}
$$

Recalling the previous definitions, the equation for $\mathbf{P}\mathbf{u}^{n,1}$ can be rewritten as

$$
\mathbf{P}\mathbf{D}_{1/2}\mathbf{E}\mathbf{P}\mathbf{u}^{n,1} = \mathbf{P}\mathbf{N}_{1/2}\mathbf{u}^{n} - \mathbf{P}\mathbf{D}_{1/2}\mathbf{\Pi}^{\perp}\mathbf{Q}_{1/2}\mathbf{u}^{n},
$$

which implies

$$
\mathbf{P}\mathbf{u}^{n,1} = (\mathbf{P}\mathbf{D}_{1/2}\mathbf{E})^{-1}\left[\mathbf{P}\mathbf{N}_{1/2} - \mathbf{P}\mathbf{D}_{1/2}\mathbf{\Pi}^{\perp}\mathbf{Q}_{1/2}\right]\mathbf{u}^{n}.
$$

Substituting this expression in the last equation of (6.1) one obtains

$$
\begin{aligned}
&\mathbf{P}\mathbf{D}_{1/2}\mathbf{E}\mathbf{P}\mathbf{u}^{n,2} = -\mathbf{P}\mathbf{D}_{1/2}\mathbf{\Pi}^{\perp}\mathbf{R}\mathbf{u}^{n} \\
&+\mathbf{P}\mathbf{N}_{1/2}\mathbf{E}(\mathbf{P}\mathbf{D}_{1/2}\mathbf{E})^{-1}\left[\mathbf{P}\mathbf{N}_{1/2} - \mathbf{P}\mathbf{D}_{1/2}\mathbf{\Pi}^{\perp}\mathbf{Q}_{1/2}\right]\mathbf{u}^{n} \qquad (6.2) \\
&+\mathbf{P}\mathbf{N}_{1/2}\mathbf{\Pi}^{\perp}\mathbf{Q}_{1/2}\mathbf{u}^{n}.
\end{aligned}
$$

The stability matrix of the complete multirate method can then be obtained deriving from (6.2) an explicit expression for $\mathbf{P}\mathbf{u}^{n,2}$ and substituting in

$$
\mathbf{u}^{n+1} = \mathbf{\Pi}\mathbf{u}^{n,2} + \mathbf{\Pi}^{\perp}\mathbf{R}\mathbf{u}^{n}.
$$

Introducing for compactness the notation

$$
\mathbf{A}^{\parallel,\parallel} = \mathbf{P}\mathbf{A}\mathbf{E}, \quad \mathbf{A}^{\parallel,\perp} = \mathbf{P}\mathbf{A}\mathbf{E}^{\perp}
$$

one can express the stability matrix as follows

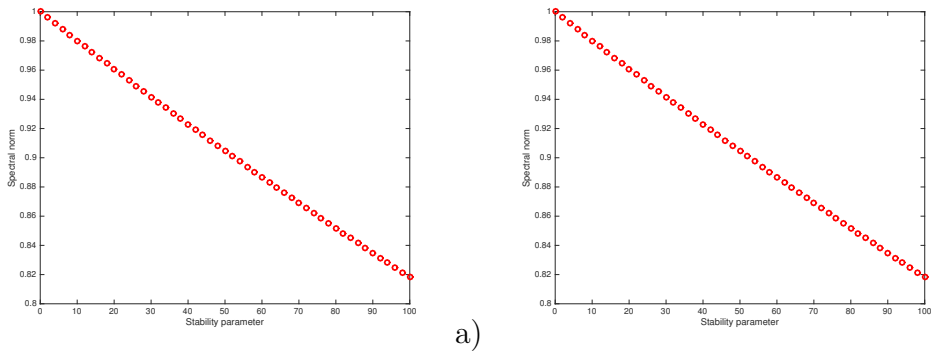
$$
\begin{aligned}
\mathbf{R}_{mr} \;=\; & \mathbf{E}\left\{\left(\mathbf{D}_{1/2}^{\parallel,\parallel}\right)^{-1}\mathbf{N}_{1/2}^{\parallel,\parallel}\left(\mathbf{D}_{1/2}^{\parallel,\parallel}\right)^{-1}\left[\mathbf{P}\mathbf{N}_{1/2} - \mathbf{D}_{1/2}^{\parallel,\perp}\mathbf{P}^{\perp}\mathbf{Q}_{1/2}\right]\right. \\
+ \;& \left.\left(\mathbf{D}_{1/2}^{\parallel,\parallel}\right)^{-1}\mathbf{N}_{1/2}^{\parallel,\perp}\mathbf{P}^{\perp}\mathbf{Q}_{1/2} - \left(\mathbf{D}_{1/2}^{\parallel,\parallel}\right)^{-1}\mathbf{D}_{1/2}^{\parallel,\perp}\mathbf{P}^{\perp}\mathbf{R}\right\} + \mathbf{\Pi}^{\perp}\mathbf{R}.
\end{aligned}
$$

We will not attempt a complete theoretical analysis based on the previously derived expression of $\mathbf{R}_{mr}$. However, the same expression can be employed to compute numerically the stability matrix norm and spectral radius for relevant model problems. We have done so for three classes of linear ODE systems $\mathbf{y}' = \mathbf{A}\mathbf{y}$. Notice that, for simplicity, all the systems have been rearranged so that the latent variables are the first of the vector of unknowns.

Firstly, we consider a $2 \times 2$ system with the same characteristics as those employed in [12], defined by

$$\mathbf{A} = \begin{bmatrix} -1 & 1 \\ -1000 & -1000 \end{bmatrix}. \tag{6.3}$$

In particular, in the notation of [12], these coefficient values correspond to the case of $\kappa = 1000$, $\beta = -1$. The amplification matrix and its spectral and $l_1$, $l_2$, $l_\infty$ matrix norms have been computed for different values of the time step, up to a maximum value $h_{max}$ such that $h_{max}/\max|\lambda(\mathbf{A})| = 100$. Therefore, the maximum time step considered is approximately 100 times larger than that of the customary stability restriction for explicit schemes. The values of the matrix norms are reported in figures 1, 2, for the spectral norm and the other matrix norms, respectively. It can be observed that no stability problems arise and that the multirate algorithm appears to be insensitive to the choice of the interpolation method.



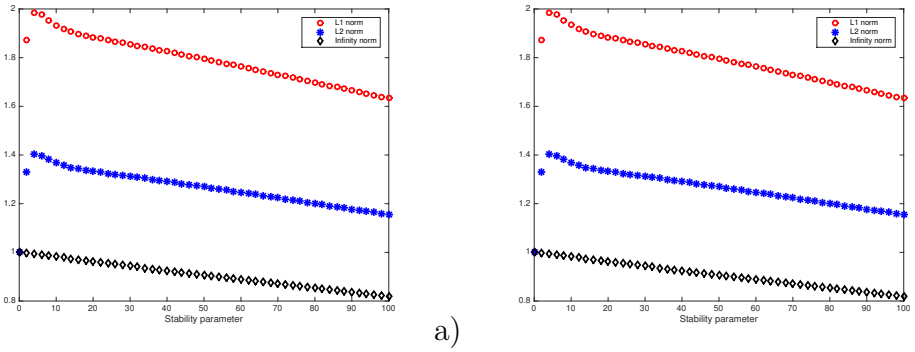

a) b)

Figure 1: Spectral norm of the TR-BDF2 multirate method amplification matrix for system (6.3), a) linear interpolation, b) cubic Hermite interpolation, for increasing values of the rescaled time step.

We have then considered the $4 \times 4$ dynamical system defined

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_1}{m_1} & -\gamma_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{k_2}{m_2} & 0 & -\frac{k_2}{m_2} & -\gamma_2 \end{bmatrix}. \tag{6.4}$$

The system represents two masses $m_1, m_2$ such that the first is tied to a wall by a spring of elastic constant $k_1$ and the second is tied to the first by a spring of elastic constant $k_2$, while $\gamma_1, \gamma_2$ represent friction parameters. Similar systems have been often used to analyze empirically the stability of numerical methods for structural mechanics, see




a) b)

Figure 2: Matrix norms of the TR-BDF2 multirate method amplification matrix for system (6.3), a) linear interpolation, b) cubic Hermite interpolation, for increasing values of the rescaled time step.

e.g. [6]. We have considered as an example the case $m_1 = m_2 = 1$, $k_1 = 1$, $k_2 = 10^6$, $\gamma_1 = 0$, $\gamma_2 = 100$. The values of the matrix norms are reported in figures 3, 4, for the spectral norm and the other matrix norms, respectively. We observe again that, even though the stability function norm can achieve large values, the method is still stable in the spectral norm sense. Furthermore, it does not appear to be sensitive to the choice of the interpolation operator. In figure 5, the same quantities are plotted for the same system in the case $\gamma_2 = 0$. The good stability behaviour displayed in these tests seems to justify further work on the application of this multirate approach to structural mechanics problems with multiple time scales, such as those considered e.g. in [6]. As a comparison, we also plot in figure 6 the corresponding norms for the single rate TR-BDF2 algorithm, that display an entirely analogous behaviour.

Finally, we have considered a $40 \times 40$ linear ODE system resulting from the discretization of a linear heat equation by second finite differences, with a diffusivity parameter $10^6$ times larger for the last 20 variables than for the first 20. All matrix norms are essentially identical to one over the whole range of time step values. Analogous results are obtained considering the corresponding system for linear advection-diffusion equation, taking in this case centered finite differences for the advection discretization and assuming the velocity and diffusivity parameters to be $10^4$ times larger for the last 20 variables than for the first 20. The values of the matrix norms are reported in figure 7 for the cubic interpolation case. In figure 8, the analogous quantities are plotted instead in the case of a pure advection system with the same velocity values as in the previous case. The good stability properties that result in this case seem to justify further work

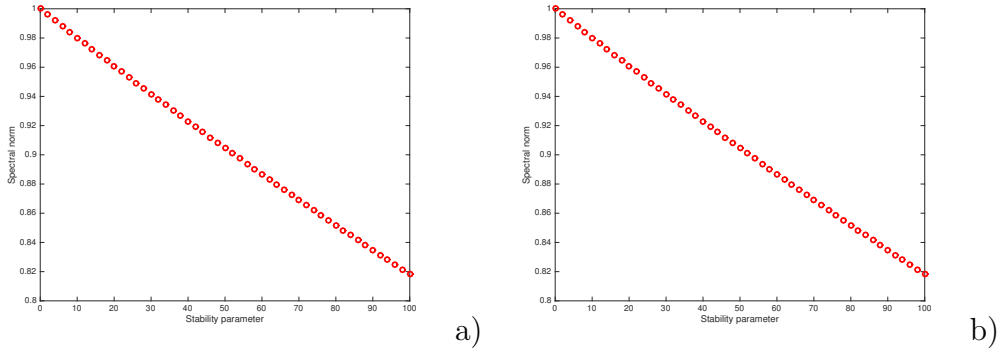



Figure 3: Spectral norm of the TR-BDF2 multirate method amplification matrix for system (6.4), a) linear interpolation, b) cubic Hermite interpolation, for increasing values of the rescaled time step.

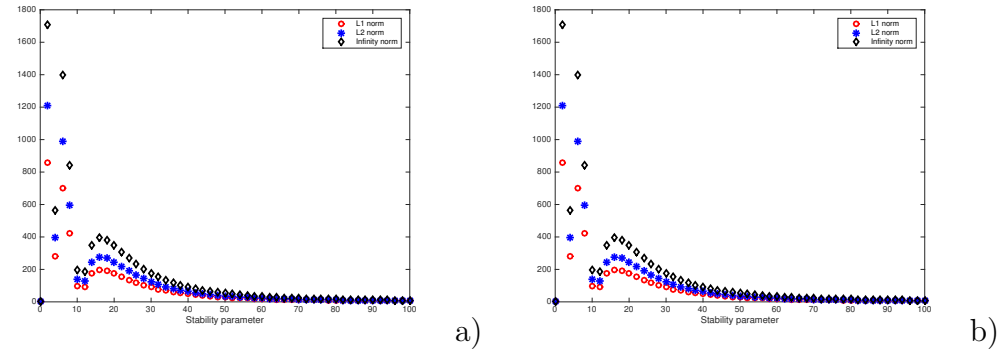
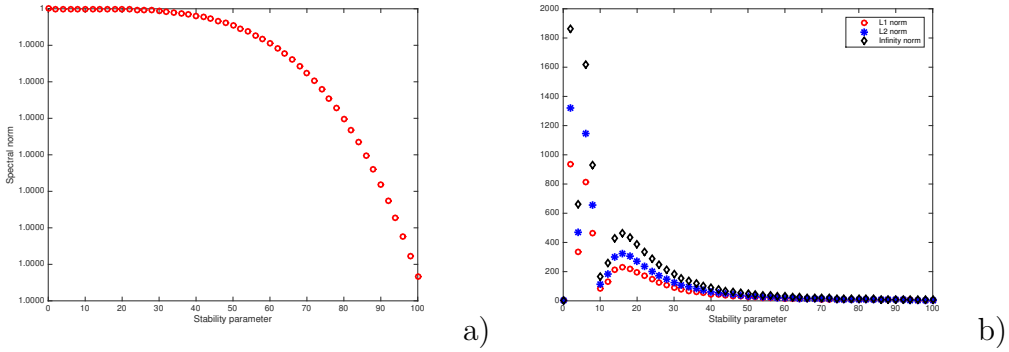
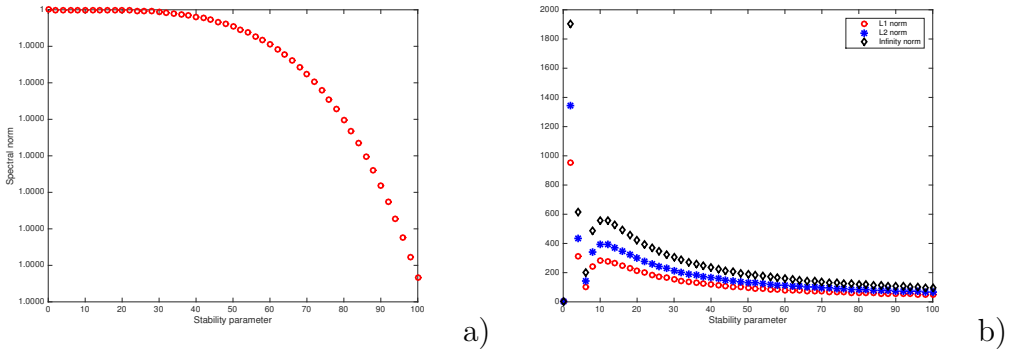



Figure 4: Matrix norms of the TR-BDF2 multirate method amplification matrix for system (6.4), a) linear interpolation, b) cubic Hermite interpolation, for increasing values of the rescaled time step.




a) b)

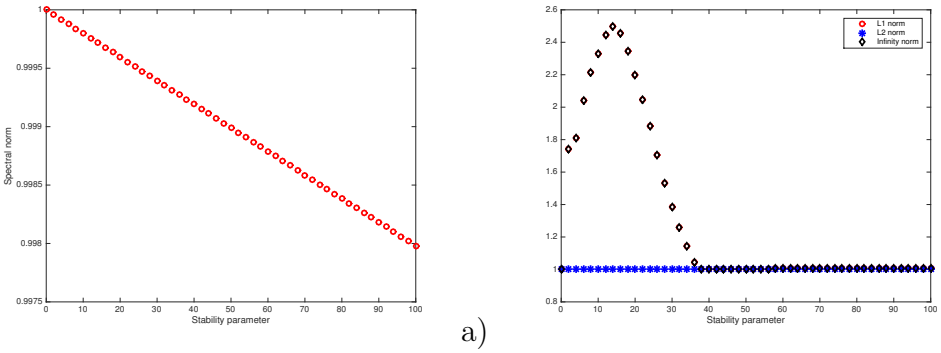
Figure 5: Norms of the TR-BDF2 multirate method amplification matrix for system (6.4) without friction, cubic Hermite interpolation a) spectral norm, b) other norms for increasing values of the rescaled time step.




a) b)

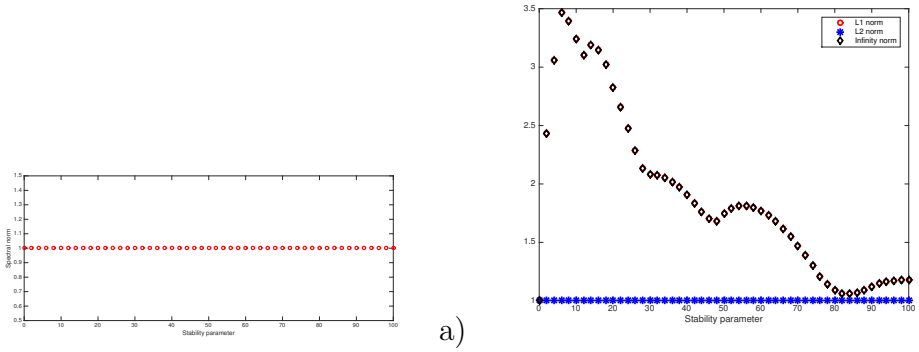
Figure 6: Norms of the TR-BDF2 single rate method amplification matrix for system (6.4) without friction, a) spectral norm, b) other norms for increasing values of the rescaled time step.

on the application of this multirate approach to hyperbolic equations with multiple time scales, such as those considered e.g. in [22].




a) b)

Figure 7: Matrix norms of the TR-BDF2 multirate method amplification matrix for an advection diffusion system with strongly varying coefficients and cubic interpolation, a) spectral norm, b) other matrix norms.




a) b)

Figure 8: Matrix norms of the TR-BDF2 multirate method amplification matrix for an advection system with strongly varying coefficients and cubic interpolation, a) spectral norm, b) other matrix norms.

# 7 Numerical experiments

Several benchmark problems already considered in the literature have been employed to assess the performance of the multirate algorithm outlined in section 2. The self adjusting multirate approach based on the TR-BDF2 solver has been compared to its single rate counterpart, considering several performance indicators, such as the CPU time required by the codes, the number of scalar function calls

and the number of components being computed at any given time step, as an estimate of the computational workload for each time step. In all the numerical experiments, the TR-BDF2 multirate algorithm employed the cubic Hermite interpolant and a nonlinear solver based on the Newton method with approximate computation of the Jacobian to solve the nonlinear systems involved.

## 7.1 The Inverter Chain Problem

For the first test we consider the inverter chain problem which is an important test problem from the field of electrical circuits. It has also been considered e.g. in [19], [23]. The system of equations is given by

$$\begin{aligned} y_1'(t) &= U_{op} - y_1(t) - \Gamma g(u(t), y_1(t)), \\ y_j'(t) &= U_{op} - y_j(t) - \Gamma g(y_{j-1}(t), y_j(t)), \qquad j = 2, 3, \ldots, m, \end{aligned} \tag{7.1}$$

where $y_j, j = 1, \ldots, m$ represent a chain of $m$ inverters, $U_{op}$ is the operating voltage corresponding to the logical value 1 and $\Gamma$ serves as a stiffness parameter. The function $g$ is defined by

$$g(y, z) = \left(\max\left(y - U_\tau, 0\right)\right)^2 - \left(\max\left(y - z - U_\tau, 0\right)\right)^2. \tag{7.2}$$

The parameter values used were $m = 500$, $\Gamma = 100$, $U_\tau = 1$ and $U_{op} = 5$. The initial condition was defined as

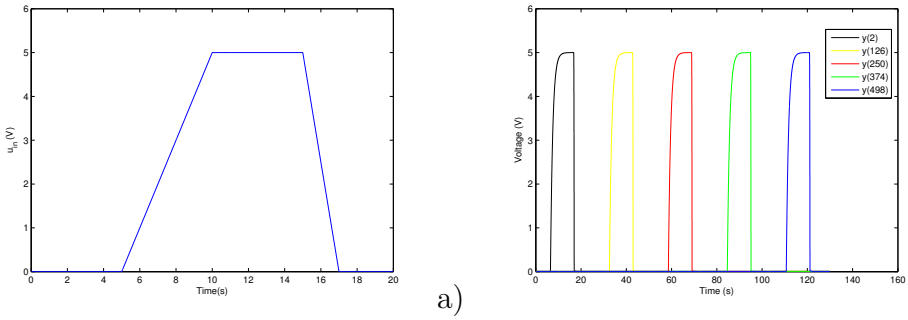
$$y_j(0) = \begin{cases} 6.247 \times 10^{-3} & \text{for } j \text{ even} \\ 5 & \text{for } j \text{ odd}, \end{cases} \tag{7.3}$$

while the input signal was given by

$$u(t) = \begin{cases} t - 5 & \text{for } 5 \le t \le 10 \\ 5 & \text{for } 10 \le t \le 15 \\ \dfrac{5}{2}(17 - t) & \text{for } 15 \le t \le 17 \\ 0 & \text{otherwise} \end{cases} \tag{7.4}$$

In this test case, the values of all the components vary in the range $(0, 5)$, so that only the absolute tolerance was used.

Figure 9 shows the solution of the inverter chain problem described above. The input signal can be seen to be propagating across the chain of inverters. It is seen that the input signal is also smoothed by the inverter chain. The reference solution was computed by the `ode15s` MATLAB solver with stringent tolerance and small maximum time




a) b)

Figure 9: Solution of the inverter chain problem: a) input signal applied to the first inverter as a function of time, b) variation of voltage in 5 inverters with respect to time

step. The same reference solution was employed to assess the accuracy of the multirate approach with respect to its single rate counterpart. The maximum norm errors at the final time of the simulation are shown in figure 10. Even though the multirate approach introduces a larger error than that of the single rate counterpart, the errors committed are in general close to the specified tolerances.

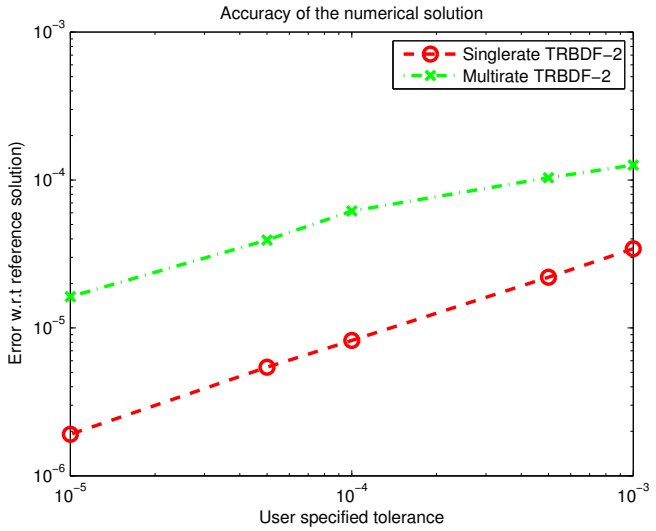



Figure 10: Errors of the multirate and single rate TR-BDF2 algorithms for the inverter chain problem, with respect to a reference solution obtained using the MATLAB `ode15s`.

The performance of the algorithms with respect to the different metrics was analyzed for different values of the tolerance. Figure 11

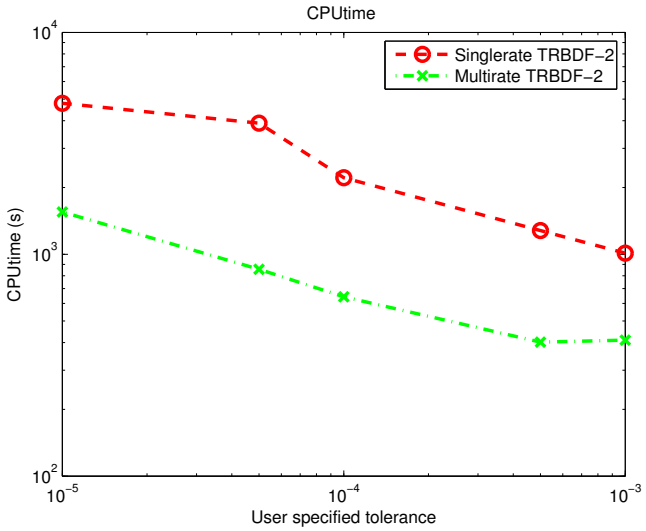



Figure 11: CPU time taken by the multirate and single rate TR-BDF2 algorithms for the inverter chain problem.

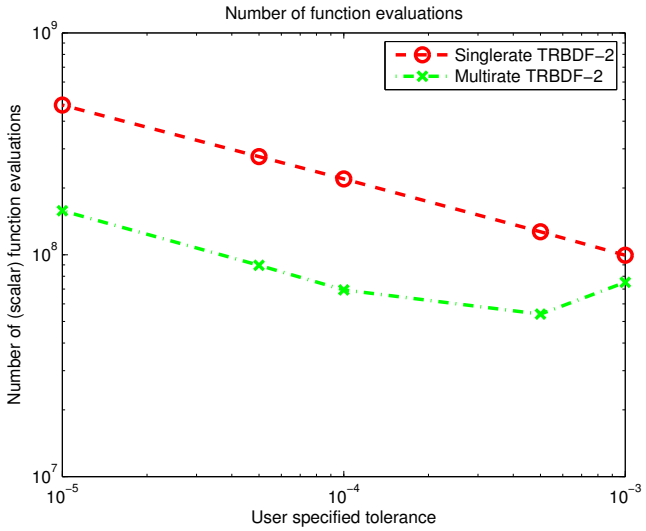



Figure 12: Number of scalar function evaluations for the multirate and singlerate TR-BDF2 algorithms for the inverter chain problem.

shows the plot of CPU time taken by the multirate and single rate TR-BDF2 algorithms. At a tolerance of $10^{-5}$, a speed-up of a factor 3 was achieved. Figure 12 shows the total number of scalar function evaluations at different tolerances. Again it can be seen that the performance of the multirate algorithm is better than that of the cor-

responding single rate algorithm. At a tolerance of $10^{-5}$ the number of function evaluations decreased by a factor of 3 for the TR-BDF2. For a tolerance of $10^{-3}$, the multirate TR-BDF2 made a larger number of function evaluations than it did at tolerance $10^{-4}$. Indeed, if the tolerance is not too stringent, the time step adaptation mechanism causes the algorithm to try larger steps, which makes it harder for the Newton solver to converge.

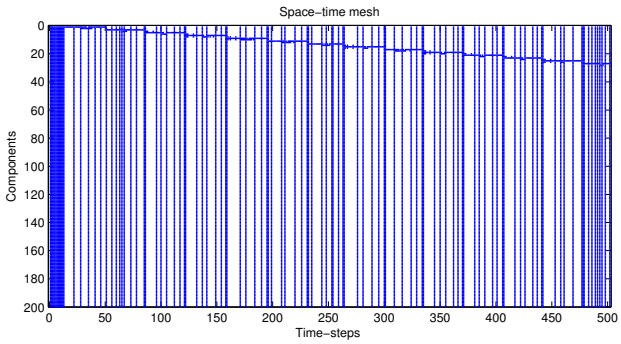



Figure 13: Space time diagram of the multirate TR-BDF2 algorithm for the inverter chain problem. Only 200 components for the first 500 time steps are shown.

In figure 13, a portion of the so-called space time diagram of the multirate algorithm is shown, denoting all the active components at each time step. For clarity, only a zoomed in view of 200 inverters for the first 500 time steps is shown. SInce the simulation is started with a very conservative value of the time step, the first few time steps involve all the degrees of freedom. The time step is then gradually increased by the adaptation mechanism, so that at a later stage there are only a few components being refined at each time step. The set of refined components moves forward across the inverters as the simulation goes on. This is as expected, because the signal is propagating across the inverter chain and the active components are being refined, while the others are being integrated with larger time-steps. The number of space-time points in the figure 13 can be regarded as a measure of the computational workload of the algorithm. Obviously, for a single rate algorithm this would simply be equal to the number of components times the number of time-steps. Figure 14 shows this estimate of computational workload for the algorithms for the inverter

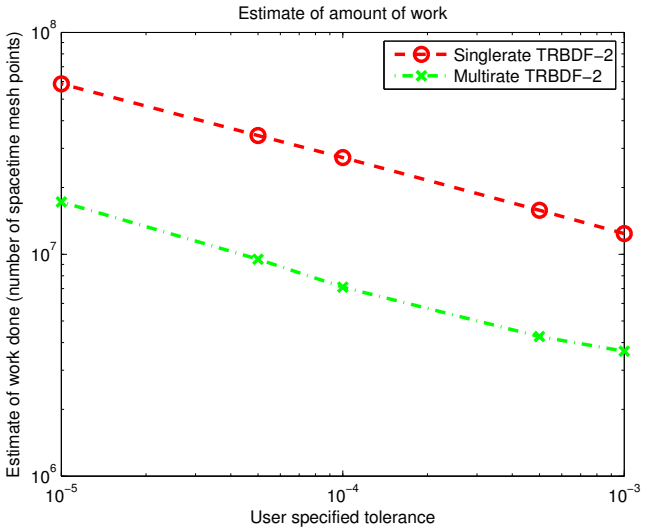



Figure 14: Workload estimate for the multirate and singlerate TR-BDF2 algorithms for the inverter chain problem.

chain problem. With regard to this metric, using a tolerance of $10^{-5}$ the multirate algorithm leads to a gain in efficiency of 3.4 times with respect to the single rate algorithm.

## 7.2 A reaction diffusion problem

We then consider a test in which the ODE problem to be solved results from the space discretization of a partial differential equation. In particular, we consider the reaction diffusion problem also discussed in [19], whose associated PDE can be written as

$$\frac{\partial y}{\partial t} = \epsilon \frac{\partial y^2}{\partial x^2} + \gamma y^2(1-y), \tag{7.5}$$

in the domain $0 < x < L$, $0 < t < T$. The initial and boundary conditions are given by

$$y(0,x) = \frac{1}{1+\exp\left(\lambda(x-1)\right)} \qquad \frac{\partial y(t,0)}{\partial x} = \frac{\partial u(t,L)}{\partial x} = 0.$$

The values used for parameters were $\lambda = \frac{1}{2}\sqrt{\frac{2\gamma}{\epsilon}}$, $\gamma = \frac{1}{\epsilon} = 100$, $L = 5$ and $T = 3$. The equation was discretized in space by simple second order finite differences on a mesh of constant spacing $\Delta x = L/N$, with different values of $N$. Even though no analytic solution is available, the solution is well known to consist in a wave moving in the positive

direction and connecting the two stable states of the nonlinear reaction potential. Figure 15 shows the solution as a function of space at different time instants.

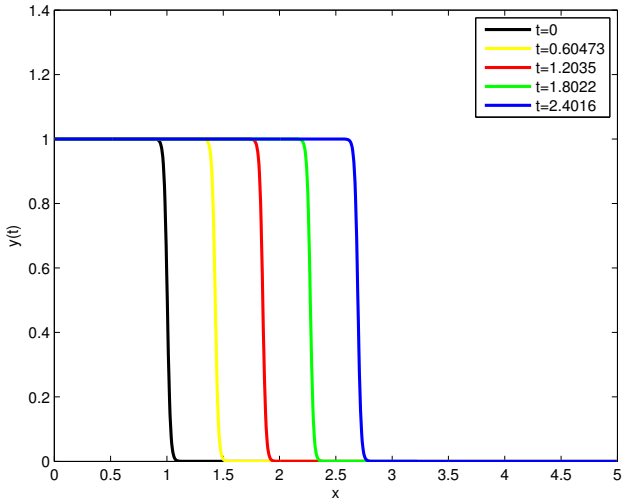



Figure 15: Reference solution of the reaction-diffusion problem obtained by the MATLAB `ode15s` solver, shown at five evenly spaced time instants in the simulation interval.

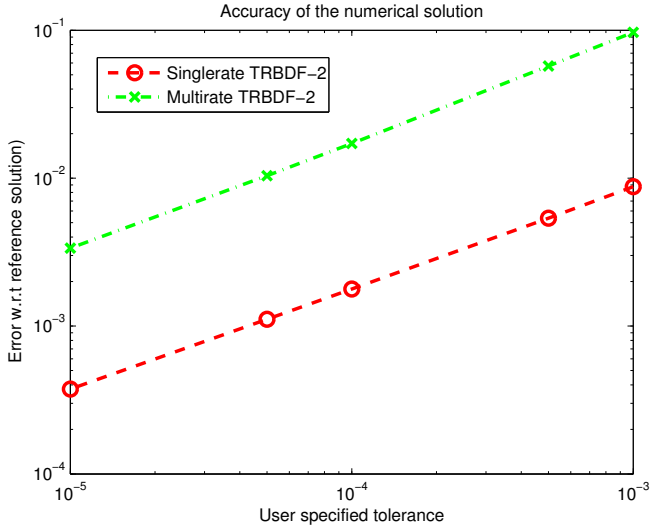



Figure 16: Errors committed by the multirate and single rate TR-BDF2 algorithms for different tolerances for the reaction diffusion problem described in section 7.2

As in the previous section, we have compared the accuracy of the proposed algorithm with respect to the reference solution computed by the MATLAB `ode15s` solver with tight tolerance and small maximum time step values. Figure 16 shows the plot of the errors at the final simulation time $T = 3$s for different values of user specified tolerance. The performance of the algorithms with respect to the different metrics was analyzed for different values of the tolerance. Figure 17 shows the plot of CPU time and the number of function evaluations required by the multirate and single rate TR-BDF2 algorithms, respectively. At a tolerance of $10^{-5}$, the speed-up of the multirate algorithm with respect to the single rate algorithm was 2.7.

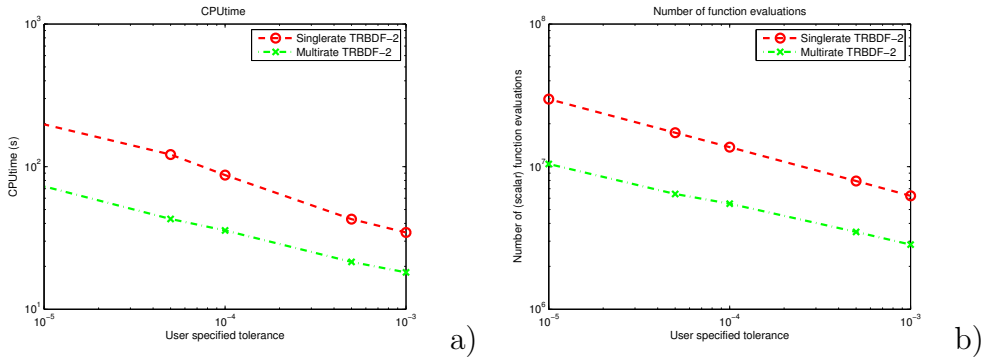



Figure 17: a) CPU time and b) number of scalar function evaluations for the multirate and single rate TR-BDF2 algorithms for different tolerances for the reaction diffusion problem.

## 7.3 Linear Advection Equation

As first example of hyperbolic equations we consider the linear problem

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0, \qquad x \in [-20, 20] \tag{7.6}$$

with a Gaussian initial datum. To discretize in space we used a first order upwind method with periodic boundary conditions. The interval time is $[0, 3]$ with a number of cells equal to 400. The relative error tolerance was set to $10^{-6}$, while the absolute error tolerance was set to $10^{-8}$. The initial size of the time step was taken to be $10^{-2}$. The upwind discretization entails a relevant amount of numerical diffusion, which is responsible for the spreading of the initial profile. This effect also entails that the number of computed components increases as time progresses, see Figure 18.

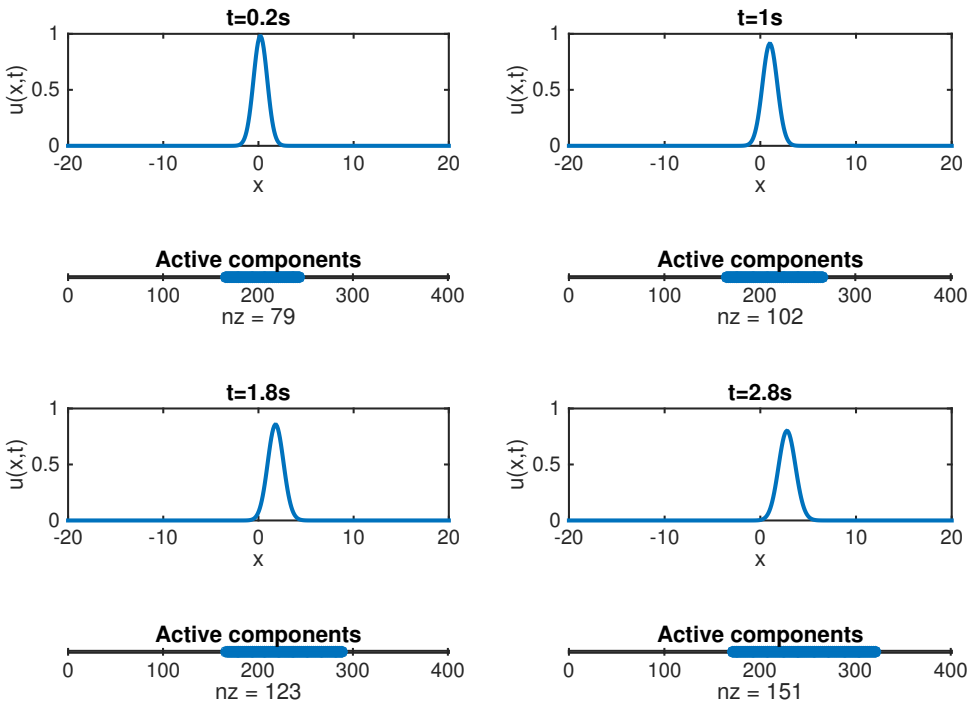



Figure 18: Multirate TR-BDF2 method with cubic interpolation for the linear advection problem.

| Method | CPU time | Number of time steps |
|---|---|---|
| Single rate TR-BDF2 | $21.41s$ | 323 |
| Multirate TR-BDF2 | $11.59s$ | 433 |

Table 1: CPU time and number of time steps for the linear advection test.

| Time [s] | Cubic Interp. | Linear Interp. | Single rate |
|---|---|---|---|
| 0.2 | $5.7 \times 10^{-2}$ | $5.71 \times 10^{-2}$ | $5.88 \times 10^{-2}$ |
| 1 | $2.73 \times 10^{-1}$ | $2.74 \times 10^{-1}$ | $2.63 \times 10^{-1}$ |
| 1.8 | $4.52 \times 10^{-1}$ | $4.53 \times 10^{-1}$ | $4.46 \times 10^{-1}$ |
| 2.8 | $6.41 \times 10^{-1}$ | $6.43 \times 10^{-1}$ | $6.41 \times 10^{-1}$ |

Table 2: Relative error in $l^\infty$ norm at different times for the linear advection problem. The exact solution was used as a reference and the solution was computed with a) the multirate method with cubic Hermite interpolator b) the multirate method with linear interpolator and c) the single-rate TR-BDF2 method.

| Time [s] | Cubic Interp. | Linear Interp. | Single rate |
|---|---|---|---|
| 0.2 | $1.41 \times 10^{-6}$ | $3.07 \times 10^{-5}$ | $1.38 \times 10^{-6}$ |
| 1 | $5.45 \times 10^{-6}$ | $1.54 \times 10^{-5}$ | $4.76 \times 10^{-6}$ |
| 1.8 | $8.78 \times 10^{-6}$ | $1.49 \times 10^{-5}$ | $8.80 \times 10^{-6}$ |
| 2.8 | $1.24 \times 10^{-5}$ | $1.72 \times 10^{-5}$ | $1.02 \times 10^{-5}$ |

Table 3: Relative errors in $l^\infty$ norm at different times for the linear advection problem. The `ode45` solver solution was used as a reference and the solution was computed with a) the multirate method with cubic Hermite interpolator b) the multirate method with linear interpolator and c) the single-rate TR-BDF2 method.

The CPU times and total number of computed time steps are reported in Table 1 for the multirate and single rate TR-BDF2 methods, respectively. For the multirate method, cubic interpolation was found to be necessary in order to achieve an improvement over the performance of the single rate method, while linear interpolation lead to an excessive number of rejections of macro steps. In order to assess also the accuracy of the multirate approach, the relative errors obtained in this test case are reported in Table 2. More specifically, errors obtained by the multirate TR-BDF2 method with cubic Hermite interpolator, the multirate TR-BDF2 method with linear interpolator

and the single-rate TR-BDF2 method are compared. It can be seen that all three approaches yield comparable errors. Since the first order upwind space discretization is responsible for the largest part of the error, we also compute the error with respect to the reference solution obtained by discretizing in time with the MATLAB `ode45` solver set to a suitably stringent tolerance. The results, reported in Table 3, show that also in this case the errors obtained with the three variants are entirely analogous, albeit smaller by several orders of magnitude due the factoring out of the spatial discretization error.

## 7.4 Burgers equation

Finally, as a nonlinear test for hyperbolic equations, we consider the Riemann problem for the inviscid Burgers equation

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right) = 0, \qquad x \in [-1, 3], \quad t > 0, \tag{7.7}$$

which amounts to assigning a piecewise constant initial datum $u_0(x) = u_l \qquad x < 0$, $u_0(x) = u_r \qquad x > 0$. We first consider the case $u_l > u_r$, for which a shock wave solution is obtained. The space discretization is given by finite volume method with Rusanov numerical flux, see e.g. [15]. The interval time is $[0, 1]$ with a number of cells equal to 400. The relative error tolerance was set to $10^{-4}$, while the absolute error tolerance was set to $10^{-6}$ and the tolerance for the Newton solver required by the implicit TR-BDF2 method was taken to be $10^{-8}$. We took the initial size of the time step equal to $10^{-2}$. For the Riemann problem we used in this first case $u_l = 1$, while $u_r = 0$.

| Method | CPU time | Number of time steps |
|---|---|---|
| Single rate TR-BDF2 | $78.23s$ | 1008 |
| Multirate TR-BDF2 | $24.25s$ | 1378 |

Table 4: CPU time and number of time steps for the Burgers equation Riemann problem with shock wave solution.

Figure 19 shows how the shock wave solution evolves in space as time progresses. It can be seen how the automatically selected set of active components moves along with the shock wave and does not increase in size. The same result is apparent from the space time diagram in Figure 20. The CPU times and total number of computed time steps are reported in Table 4 for the multirate and single rate TR-BDF2 methods, respectively. For the multirate method, cubic interpolation was found to be necessary in order to achieve an improvement over the performance of the single rate method, while linear

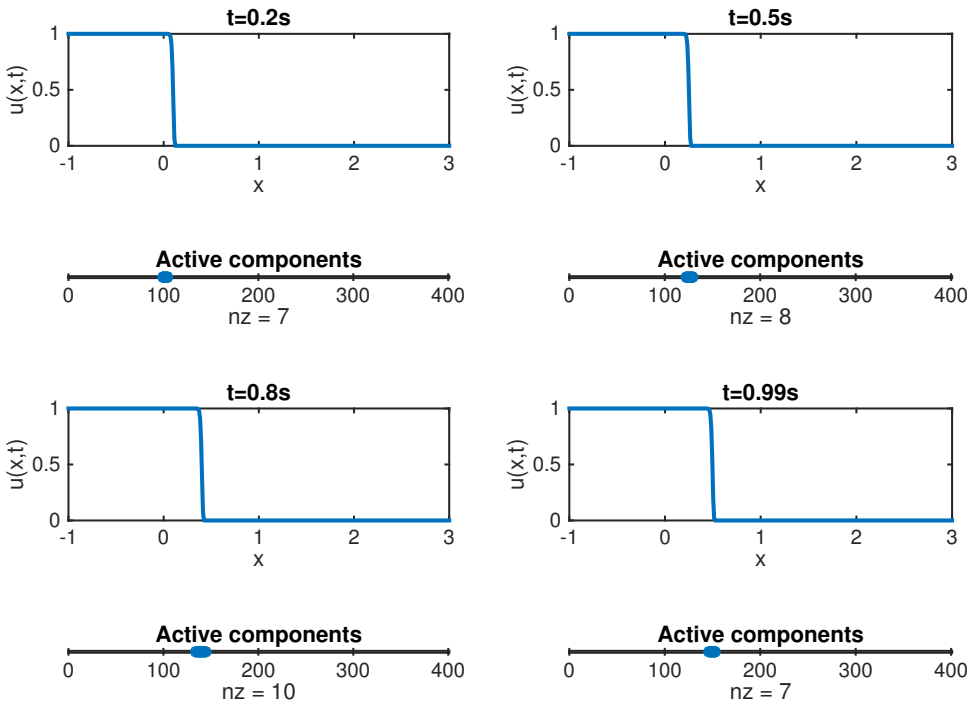



Figure 19: Multirate TR-BDF2 integration with cubic interpolation for the Burgers equation Riemann problem with shock wave solution.

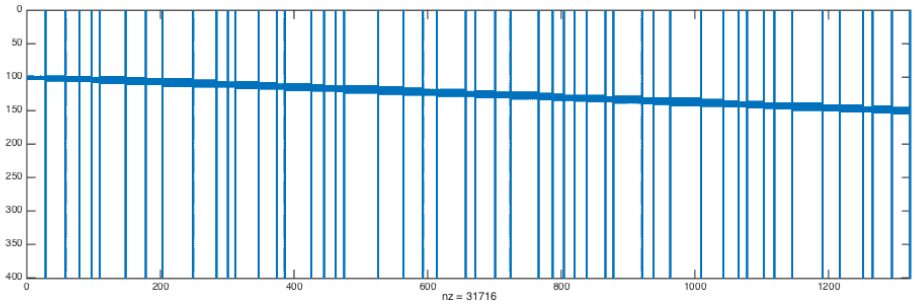



Figure 20: Space time diagram for the multirate TR-BDF2 algorithm for the Burgers equation Riemann problem with shock wave solution.

interpolation lead to an excessive number of rejections of macro steps. In order to assess also the accuracy of the multirate approach, the

| Time [s] | Cubic Interp. | Linear Interp. | Single rate |
|---|---|---|---|
| 0.2 | $4.85 \times 10^{-1}$ | $4.83 \times 10^{-1}$ | $4.81 \times 10^{-1}$ |
| 0.5 | $4.86 \times 10^{-1}$ | $4.86 \times 10^{-1}$ | $4.83 \times 10^{-1}$ |
| 0.8 | $4.81 \times 10^{-1}$ | $4.82 \times 10^{-1}$ | $4.80 \times 10^{-1}$ |
| 0.99 | $5.27 \times 10^{-1}$ | $5.31 \times 10^{-1}$ | $5.24 \times 10^{-1}$ |

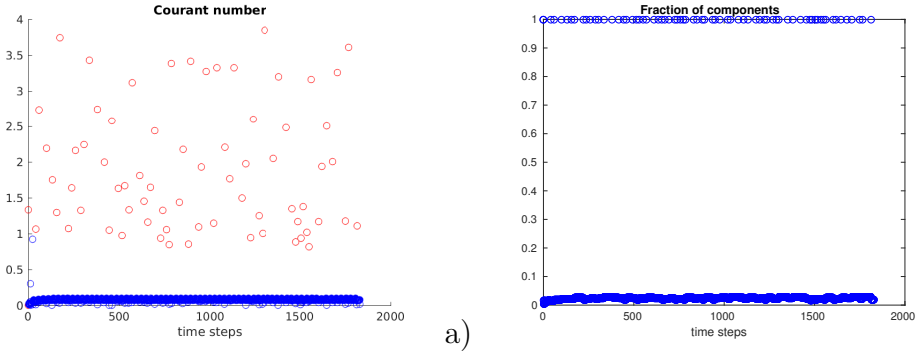
Table 5: Relative errors in $l^\infty$ norm at different times for the Burgers equation with shock wave solution. The exact solution was used as a reference and the solution was computed with a) the multirate method with cubic Hermite interpolator b) the multirate method with linear interpolator and c) the single-rate TR-BDF2 method.

| Time [s] | Cubic interp. | Linear interp. | Single rate |
|---|---|---|---|
| 0.2 | $4.37 \times 10^{-5}$ | $1.48 \times 10^{-4}$ | $1.34 \times 10^{-5}$ |
| 0.5 | $2.76 \times 10^{-4}$ | $6.08 \times 10^{-4}$ | $3.5 \times 10^{-5}$ |
| 0.8 | $6.65 \times 10^{-4}$ | $1.08 \times 10^{-3}$ | $3.63 \times 10^{-5}$ |
| 0.99 | $9.18 \times 10^{-4}$ | $1.2 \times 10^{-3}$ | $3.59 \times 10^{-5}$ |

Table 6: Relative errors in $l^\infty$ norm at different times for the Burgers equation Riemann problem with shock wave solution. The `ode45` solution was used as a reference and the solution was computed with a) the multirate method with cubic Hermite interpolator b) the multirate method with linear interpolator and c) the single-rate TR-BDF2 method.

relative errors obtained in this test case are reported in Table 5. More specifically, errors obtained by the multirate TR-BDF2 method with cubic Hermite interpolator, the multirate TR-BDF2 method with linear interpolator and the single-rate TR-BDF2 method are compared. It can be seen that all three approaches yield comparable errors. Since the first order upwind space discretization is responsible for the largest part of the error, we also compute the error with respect to the reference solution obtained by discretizing in time with the MATLAB `ode45` solver set to a suitably stringent tolerance. The results, reported in Table 6, show that also in this case the errors obtained with the three variants are entirely analogous, albeit smaller by several orders of magnitude due the factoring out of the spatial discretization error.

In order to highlight the capability of the adaptive strategy to select automatically a time step that is compatible with the underlying dynamics, we have also reported the maximum Courant numbers at




a) b)

Figure 21: a) Courant numbers for each time step and b) fraction of refined components for the Burgers equation with shock wave solution (red circles in a) denote Courant numbers associated to the global time steps).

each time step, computed as

$$\max_{i=1,\dots,N_x} |f'(c_i^n)| \frac{h_n}{\Delta x} \tag{7.8}$$

where $N_x$ is the total number of cells. In Figure 21 a), the Courant number values associated to the global time steps are represented by red circles, while those associated to local, refined time steps are represented by blue symbols. It can be seen that, while large Courant numbers are used for the latent components, much smaller values are employed for the active components that correspond to the locations actually crossed by the shock wave. In Figure 21 b) the corresponding fractions of active components are displayed, highlighting the significant reduction in computational cost in this case.

We then consider the case $u_l > u_r$, for which a rarefaction wave solution is obtained. More specifically, we consider $u_l = 0$ and $u_r = 1$. The other parameters are the same as in the previous case.

| Method | CPU time | Numb. time steps |
|---|---|---|
| Single rate TR-BDF2 | $7.13s$ | 79 |
| Multirate TR-BDF2 | $2.99s$ | 169 |

Table 7: CPU time and number of time steps for the rarefaction wave.

Figure 22 shows how the rarefaction wave solution evolves in space as time progresses. It can be seen how the automatically selected set of active components moves along with the wave. The same result is apparent from the space time diagram in Figure 23. Since in this case the solution undergoes significant changes in a larger number of cells,

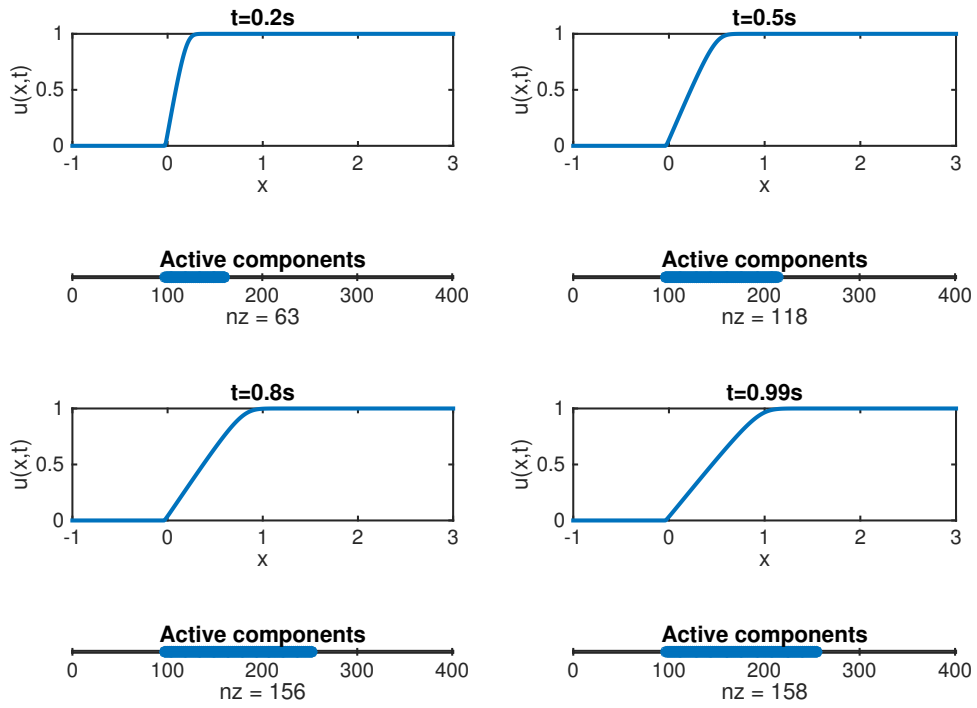



Figure 22: Multirate TR-BDF2 integration with cubic interpolation for the Burgers equation Riemann problem with rarefaction wave solution.

| Time [s] | Cubic interp. | Linear interp. | Single rate |
|---|---|---|---|
| 0.2 | $3.59 \times 10^{-1}$ | $3.59 \times 10^{-1}$ | $3.59 \times 10^{-1}$ |
| 0.5 | $3.16 \times 10^{-1}$ | $3.16 \times 10^{-1}$ | $3.14 \times 10^{-1}$ |
| 0.8 | $2.93 \times 10^{-1}$ | $2.95 \times 10^{-1}$ | $2.85 \times 10^{-1}$ |
| 0.99 | $2.84 \times 10^{-1}$ | $2.92 \times 10^{-1}$ | $2.83 \times 10^{-1}$ |

Table 8: Relative errors in $l^\infty$ norm at different times for the Burgers equation Riemann problem with rarefaction wave solution. The exact solution was used as a reference and the solution was computed with a) the multirate method with cubic Hermite interpolator b) the multirate method with linear interpolator and c) the single-rate TR-BDF2 method.

the number of active components is correspondingly larger than in the shock wave case.

The CPU times and total number of computed time steps are reported in Table 7 for the multirate and single rate TR-BDF2 methods, respectively. For the multirate method, cubic interpolation was found to be necessary in order to achieve an improvement over the perfor-

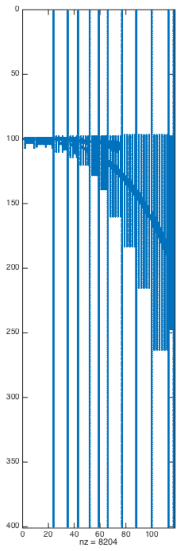


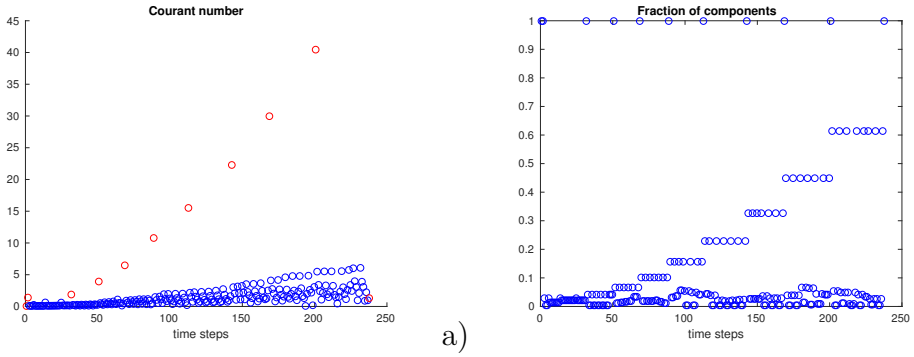

Figure 23: Space time diagram for the multirate TR-BDF2 algorithm for the Burgers equation Riemann problem with rarefaction wave solution.

| Time [s] | Cubic interp. | Linear interp. | Single rate |
|---|---|---|---|
| 0.2 | $3.13 \times 10^{-4}$ | $4.42 \times 10^{-4}$ | $1.92 \times 10^{-4}$ |
| 0.5 | $5.53 \times 10^{-4}$ | $9.91 \times 10^{-4}$ | $6.99 \times 10^{-4}$ |
| 0.8 | $7.57 \times 10^{-4}$ | $1.75 \times 10^{-3}$ | $7.25 \times 10^{-4}$ |
| 0.99 | $7.48 \times 10^{-4}$ | $1.25 \times 10^{-3}$ | $6.64 \times 10^{-4}$ |

Table 9: Relative errors in $l^\infty$ norm at different times for the Burgers equation Riemann problem with rarefaction wave solution. The `ode45` solution was used as a reference and the solution was computed with a) the multirate method with cubic Hermite interpolator b) the multirate method with linear interpolator and c) the single-rate TR-BDF2 method.

mance of the single rate method, while linear interpolation lead to an excessive number of rejections of macro steps. In order to assess also the accuracy of the multirate approach, the relative errors obtained in this test case are reported in Table 8. More specifically, errors obtained by the multirate TR-BDF2 method with cubic Hermite interpolator, the multirate TR-BDF2 method with linear interpolator and the single-rate TR-BDF2 method are compared. It can be seen that all three approaches yield comparable errors. Since the first order upwind space discretization is responsible for the largest part of the error, we also compute the error with respect to the reference solution

obtained by discretizing in time with the MATLAB `ode45` solver set to a suitably stringent tolerance. The results, reported in Table 9, show that also in this case the errors obtained with the three variants are entirely analogous, albeit smaller by several orders of magnitude due the factoring out of the spatial discretization error.




a) b)

Figure 24: a) Courant numbers for each time step and b) fraction of refined components for the Burgers equation with rarefaction wave solution (red circles in a) denote Courant numbers associated to the global time steps).

In Figure 24 a) we have reported the Courant number values associated to each time step. Comparing to the shock wave case, we can see that larger Courant numbers are allowed by the error estimator also in the case of refined time steps, as a consequence of the different dynamics of this kind of solution. In Figure 24 b) the corresponding fractions of active components are displayed, highlighting the lesser reduction in computational cost in this case with respect to the shock wave solution.

# 8 Conclusions

We have proposed a self-adjusting multirate method that relies on the TR-BDF2 method as fundamental single rate solver. The TR-BDF2 method has a number of interesting properties, such as cheap error estimation via an embedded third order method and continuous output via a cubic Hermite interpolant, that can be exploited to increase efficiency and accuracy of a multirate approach.

Our self-adjusting multirate TR-BDF2 method has been coupled to a partitioning and time step selection criterion based on the technique proposed in [7]. The stability of the method has been analyzed in the framework of the classical linear model problem. Even though the results achieved were only based on the numerical computation

of the stability function norm, we show that for a range of relevant stiff model problems the method has remarkable stability properties. This is true not only for contractive problems with strictly dissipative behaviour, but also for problems with purely imaginary eigenvalues, which suggests that the method could be advantageous also for the application to hyperbolic PDEs and structural mechanics problems. The numerical results obtained on several standard benchmarks show that application of the proposed method leads to significant efficiency gains, that are analogous to those achieved by other self-adjusting approaches when compared to their corresponding single rate variants. In particular, the has been applied to the time discretization of nonlinear, hyperbolic partial differential equations, allowing to achieve automatic detection of complex localized phenomena such as shock waves and significant reductions in computational cost.

In a companion work, we will focus on the extensive application of the method described in this paper to nonlinear hyperbolic equations and on the derivation of mass conservative versions of this multirate approach.

# Acknowledgements

This paper contains developments of the results presented in the PhD thesis by A.R. [16] and in the Master thesis by L.D. [5], both at Politecnico di Milano. A.R.'s PhD grant was supported by the Erasmus Mundus Heritage project, while he is presently supported by the Science Foundation Ireland grant 13/RC/2094. L.B. was partially supported by the INDAM - GNCS 2015 project *Metodi numerici semi-impliciti e semi-Lagrangiani per sistemi iperbolici di leggi di bilancio.* Useful discussions with Luca Formaggia are gratefully acknowledged, as well the comments by Nicola Guglielmi and Claus Führer on the first draft of the thesis by Akshay Ranade.

# References

[1] J.F. Andrus. Numerical solution of systems of ordinary differential equations separated into subsystems. *SIAM Journal of Numerical Analysis*, 16:605–611, 1979.

[2] J.F. Andrus. Stability of a multi-rate method for numerical integration of ODEs. *Computers and Mathematics with Applications*, 25:3–14, 1993.

[3] M. Baldauf. Linear stability analysis of Runge-Kutta-based partial time-splitting schemes for the Euler equations. *Monthly Weather Review*, 138:4475–4496, 2010.

[4] R.E. Bank, W.M. Coughran, W. Fichtner, E.H. Grosse, D.J. Rose, and R.K. Smith. Transient simulation of silicon devices and circuits. *IEEE Transactions on Electron Devices*, 32:1992–2007, 1985.

[5] L. Delpopolo Carciopolo. Application of the multirate TR-BDF2 method to the time discretization of nonlinear conservation laws. Master's thesis, Degree in Mathematical Engineering, Politecnico di Milano, 2015.

[6] S. Erlicher, L. Bonaventura, and O. S. Bursi. The analysis of the generalized-$\alpha$ method for non-linear dynamic problems. *Computational Mechanics*, 28:83–104, 2002.

[7] P.K. Fok. A linearly fourth order multirate Runge–Kutta method with error control. *Journal of Scientific Computing*, pages 1–19, 2015.

[8] C.W. Gear and D.R. Wells. Multirate linear multistep methods. *BIT Numerical Mathematics*, 24:484–502, 1984.

[9] F.X. Giraldo, J.F. Kelly, and E.M. Constantinescu. Implicit-explicit formulations of a three-dimensional nonhydrostatic unified model of the atmosphere (NUMA). *SIAM Journal of Scientific Computing*, 35(5):1162–1194, 2013.

[10] M. Günther, Anne Kvaernø, and P. Rentrop. Multirate partitioned Runge-Kutta methods. *BIT Numerical Mathematics*, 41:1504–514, 2001.

[11] M.E. Hosea and L.F. Shampine. Analysis and implementation of TR-BDF2. *Applied Numerical Mathematics*, 20:21–37, 1996.

[12] W. Hundsdorfer and V. Savcenco. Analysis of a multirate theta-method for stiff ODEs. *Applied Numerical Mathematics*, 59:693–706, 2009.

[13] J.B. Klemp and R.B. Wilhelmson. The simulation of three-dimensional convective storm dynamics. *Journal of the Atmospheric Sciences*, 35:1070–1096, 1978.

[14] J.D. Lambert. *Numerical methods for ordinary differential systems: the initial value problem.* Wiley, 1991.

[15] R. J. Leveque. *Finite Volume Methods for Hyperbolic Problems.* Cambridge University Press, Cambridge, UK, 2002.

[16] A. Ranade. *Multirate Algorithms Based On DIRK Methods For Large Scale System Simulation.* PhD thesis, Politecnico di Milano, 2016.

[17] J.R. Rice. Split Runge-Kutta methods for simultaneous equations. *Journal of Research of the National Institute of Standards and Technology*, 60, 1960.

[18] V. Savcenco. Comparison of the asymptotic stability properties for two multirate strategies. *Journal of Computational and Applied Mathematics*, 220:508–524, 2008.

[19] V. Savcenco, W. Hundsdorfer, and J.G. Verwer. A multirate time stepping strategy for stiff ordinary differential equations. *BIT Numerical Mathematics*, 47:137–155, 2007.

[20] W.C. Skamarock and J.B. Klemp. The stability of time-split numerical methods for the hydrostatic and the nonhydrostatic elastic equations. *Monthly Weather Review*, 120:2109–2127, 1992.

[21] G. Söderlind and L. Wang. Evaluating numerical ODE/DAE methods, algorithms and software. *Journal of Computational and Applied Mathematics*, 185:244–260, 2006.

[22] G. Tumolo and L. Bonaventura. A semi-implicit, semi-Lagrangian, DG framework for adaptive numerical weather prediction. *Quarterly Journal of the Royal Meteorological Society*, 141:2582–2601, 2015.

[23] A. Verhoeven, B. Tasić, T.G.J. Beelen, E.J.W. ter Maten, and R.M.M. Mattheij. Automatic partitioning for multirate methods. In *Scientific Computing in Electrical Engineering*, pages 229–236. Springer, 2007.

# MOX Technical Reports, last issues

Dipartimento di Matematica
Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

**07/2018** Ieva, F.; Bitonti, D.
*Network Analysis of Comorbidity Patterns in Heart Failure Patients using Administrative Data*

**06/2018** Antonietti, P.F.; Mazzieri, I.
*High-order Discontinuous Galerkin methods for the elastodynamics equation on polygonal and polyhedral meshes*

**05/2018** Pagani, S.; Manzoni, A.; Quarteroni, A.
*Numerical approximation of parametrized problems in cardiac electrophysiology by a local reduced basis method*

**03/2018** Antonietti, P. F.; Houston, P.; Pennesi, G.
*Fast numerical integration on polytopic meshes with applications to discontinuous Galerkin finite element methods*

**04/2018** Ekin, T.; Ieva, F.; Ruggeri, F.; Soyer, R.
*Statistical Medical Fraud Assessment: Exposition to an Emerging Field*

**02/2018** Canuto, C.; Nochetto, R. H.; Stevenson, R.; Verani, M.
*A saturation property for the spectral-Galerkin approximation of a Dirichlet problem in a square*

**01/2018** Berrone, S.; Bonito, A.; Stevenson, R.; Verani, M.
*An optimal adaptive Fictitious Domain Method*

**67/2017** Esterhazy, S.; Schneider, F.; Mazzieri, I; Bokelmann, G.
*Insights into the modeling of seismic waves for the detection of underground cavities*

**68/2017** Paolucci, R.; Infantino, M.; Mazzieri, I.; Özcebe, A.G.; Smerzini, C.; Stupazzini, M.
*3D physics-based numerical simulations: advantages and current limitations of a new frontier to earthquake ground motion prediction. The Istanbul case study.*

**66/2017** Tamellini, M.; Parolini, N.; Verani, M.
*An optimal control problem of two-phase compressible-incompressible flows*