European Society of Computational Methods in Sciences and Engineering (ESCMSE)

Journal of Numerical Analysis, Industrial and Applied Mathematics (JNAIAM)
vol. 3, no. 3-4, 2008, pp. 275-297
ISSN 1790–8140

BDF Compound-Fast Multirate Transient Analysis with Adaptive Stepsize Control[1]

A. Verhoeven[2], B. Tasić, T.G.J. Beelen, E.J.W. ter Maten, R.M.M. Mattheij

CASA, Department of Mathematics and Computer Science,
Technische Universiteit Eindhoven,
Den Dolech 2, 5600 MB, Eindhoven, The Netherlands

DMS, NXP Semiconductors B.V.
High Tech Campus 48, 5656 AE, Eindhoven, The Netherlands

Received January 30, 2007; accepted in revised form June 23, 2008

Abstract: Transient analysis is an important circuit simulation technique. The circuit model, which is a system of differential-algebraic equations, is solved for a given initial condition using numerical time integration techniques. Multirate methods are efficient if the dynamical behaviour of the circuit model is not uniform. This paper deals with the analysis and control of the discretization errors for multirate time integration methods.

Keywords:
Backward Difference Formula; Circuit simulation; Differential-algebraic equations; Discretization error; Multirate; Numerical time integration; Partitioning; Transient analysis

Mathematics Subject Classification: 65L; 65M55; 34E13; 47N70;

PACS: 85.40

1 Introduction

Analogue electrical circuits are usually modeled by differential-algebraic equations of the following type:

$$\frac{d}{dt}\left[\mathbf{q}(t,\mathbf{x})\right] + \mathbf{j}(t,\mathbf{x}) = \mathbf{0}. \tag{1}$$

The vector-valued functions $\mathbf{q}, \mathbf{j} \in \mathbb{R}^d$ are constructed by Modified Nodal Analysis [5] and represent the charges and currents in the network model. The state vector $\mathbf{x}(t) \in \mathbb{R}^d$ represents the nodal voltages and the currents through the voltage-defined elements like voltage sources and inductors and depends on the time variable t. A common analysis is the transient analysis, which computes the solution $\mathbf{x}(t)$ of this non-linear DAE along the time interval $[0, T]$ for a given initial state. In the classical circuit simulators this Initial Value Problem is solved by means of implicit integration

[1]Published electronically October 15, 2008
[2]Corresponding author. E-mail: averhoev@kliksafe.nl and Arie.Verhoeven@na-net.ornl.gov

methods, like the BDF-methods. All equations are discretized by means of the same stepsize. Often, parts of electrical circuits have latency or multirate behaviour. Latency means that parts of the circuit are constant or slowly time-varying during a certain time interval. Multirate behaviour means that some variables are slowly time-varying compared to other variables. In contrast to latency and activity, multirate behaviour is a relative concept and independent of the model. In both cases, it would be attractive to integrate these slow parts with a larger timestep than the other parts. This saves the computational workload while the accuracy is preserved. Multirate time-integration methods appear to be very efficient for this kind of circuit models.

1.1 Partition of the system

For a multirate method it is necessary to partition the variables and equations into an active (A) and a latent (L) part. This can be done by the user or automatically. Sometimes it is even useful to change the partition during the transient simulation. Although the construction of a good partition is an interesting research topic, it is skipped in this paper. Details can be found in [16, 18]. Let $\mathbf{B}_A \in \mathbb{R}^{d_A \times d}$ and $\mathbf{B}_L \in \mathbb{R}^{d_L \times d}$, with $d_A + d_L = d$, be selection matrices which satisfy $\mathbf{B}_A\mathbf{B}_A^T = \mathbf{I}_{d_A}, \mathbf{B}_L\mathbf{B}_L^T = \mathbf{I}_{d_L}, \mathbf{B}_A\mathbf{B}_L^T = \mathbf{O}, \mathbf{B}_L\mathbf{B}_A^T = \mathbf{O}$, and $\mathbf{B}_A^T\mathbf{B}_A + \mathbf{B}_L^T\mathbf{B}_L = \mathbf{I}_d$. Here $\mathbf{I}_{d_*} \in \mathbb{R}^{d_*^2}$ is the identity matrix and $\mathbf{O}$ a non square zero matrix. Then the variables and functions can be split in active (A) and latent (L) parts:

$$\begin{array}{rcl} \mathbf{x} & = & \mathbf{B}_A^T\mathbf{x}_A + \mathbf{B}_L^T\mathbf{x}_L, \\ \mathbf{q}(t,\mathbf{x}) & = & \mathbf{B}_A^T\mathbf{q}_A(t, \mathbf{B}_A\mathbf{x}, \mathbf{B}_L\mathbf{x}) + \mathbf{B}_L^T\mathbf{q}_L(t, \mathbf{B}_A\mathbf{x}, \mathbf{B}_L\mathbf{x}), \\ \mathbf{j}(t,\mathbf{x}) & = & \mathbf{B}_A^T\mathbf{j}_A(t, \mathbf{B}_A\mathbf{x}, \mathbf{B}_L\mathbf{x}) + \mathbf{B}_L^T\mathbf{j}_L(t, \mathbf{B}_A\mathbf{x}, \mathbf{B}_L\mathbf{x}). \end{array} \tag{2}$$

Because of the properties of $\mathbf{B}_A, \mathbf{B}_L$ we have $\mathbf{x}_A = \mathbf{B}_A\mathbf{x}, \mathbf{x}_L = \mathbf{B}_L\mathbf{x}, \mathbf{q}_A = \mathbf{B}_A\mathbf{q}$, etc. Now equation (1) is equivalent to the following partitioned system:

$$\frac{d}{dt}\left[\mathbf{q}_A(t, \mathbf{x}_A, \mathbf{x}_L)\right] + \mathbf{j}_A(t, \mathbf{x}_A, \mathbf{x}_L) = \mathbf{0}, \tag{3}$$

$$\frac{d}{dt}\left[\mathbf{q}_L(t, \mathbf{x}_A, \mathbf{x}_L)\right] + \mathbf{j}_L(t, \mathbf{x}_A, \mathbf{x}_L) = \mathbf{0}. \tag{4}$$

Of course it is also possible to extend this partition in a further partition of k sub-systems, where the k sub-systems have an decreasing activity

$$\begin{array}{c} \frac{d}{dt}\left[\mathbf{q}_1(t, \mathbf{x}_1, \ldots \mathbf{x}_k)\right] + \mathbf{j}_1(t, \mathbf{x}_1, \ldots, \mathbf{x}_k) = \mathbf{0}, \\ \vdots \\ \frac{d}{dt}\left[\mathbf{q}_k(t, \mathbf{x}_1, \ldots \mathbf{x}_k)\right] + \mathbf{j}_k(t, \mathbf{x}_1, \ldots \mathbf{x}_k) = \mathbf{0}. \end{array} \tag{5}$$

Now we need the selection matrices $\mathbf{B}_i \in \mathbb{R}^{d_i \times d}$ for $i = 1, \ldots, k$ with the properties:

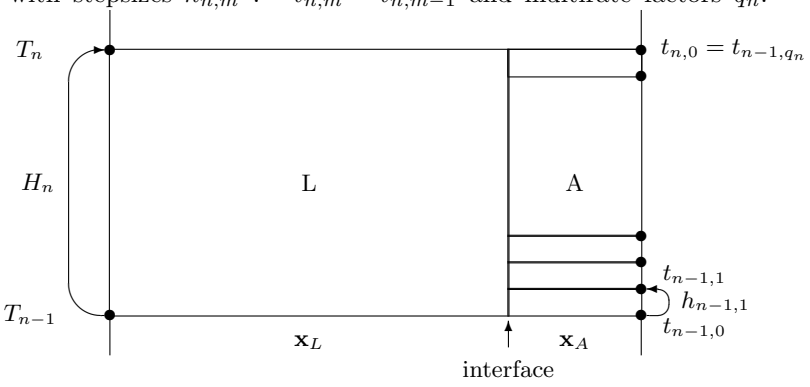
$$\mathbf{B}_i\mathbf{B}_j^T = \begin{cases} \mathbf{I}_{d_i} & \text{if } i = j, \\ \mathbf{O} & \text{if } i \neq j. \end{cases}$$

Then the variables and functions can be split in parts of different activity, where again $\mathbf{x}_i = \mathbf{B}_i\mathbf{x}$, etc.

$$\begin{array}{rcl} \mathbf{x} & = & \mathbf{B}_1^T\mathbf{x}_1 + \ldots + \mathbf{B}_k^T\mathbf{x}_k, \\ \mathbf{q}(t,\mathbf{x}) & = & \mathbf{B}_1^T\mathbf{q}_1(t, \mathbf{B}_1\mathbf{x}, \ldots, \mathbf{B}_k\mathbf{x}) + \ldots + \mathbf{B}_k^T\mathbf{q}_k(t, \mathbf{B}_1\mathbf{x}, \ldots, \mathbf{B}_k\mathbf{x}), \\ \mathbf{j}(t,\mathbf{x}) & = & \mathbf{B}_1^T\mathbf{j}_1(t, \mathbf{B}_1\mathbf{x}, \ldots, \mathbf{B}_k\mathbf{x}) + \ldots + \mathbf{B}_k^T\mathbf{j}_k(t, \mathbf{B}_1\mathbf{x}, \ldots, \mathbf{B}_k\mathbf{x}). \end{array} \tag{6}$$

1.2 Multirate time-integration

In contrast to classical integration methods, when $k = 2$, multirate time-integration methods integrate both parts (3),(4) with different stepsizes or even with different schemes. Besides the coarse time-grid $\{T_n, 0 \leq n \leq N\}$ with stepsizes $H_n := T_n - T_{n-1}$, also a refined time-grid $\{t_{n-1,m}, 1 \leq n \leq N, 0 \leq m \leq q_n\}$ is used with stepsizes $h_{n,m} := t_{n,m} - t_{n,m-1}$ and multirate factors q_n. For non-synchronized grids interpolation at the coarse time-points T_n is used, which can be more efficient for low multirate factors. Because this might also reduce the accuracy and stability, we will only consider synchronized time-grids. If the two time-grids are synchronized, $T_n = t_{n,0} = t_{n-1,q_n}$ holds for all n. The integration orders of the BDF metod for the coarse and refined time-grids are K_n and $k_{n,m}$ respectively. Although they can be variable, from now on they are assumed to be fixed with $K_n = K, k_{n,m} = k$.



There are several multirate approaches [1, 2, 7, 12, 14, 15, 17, 20] for the partitioned system (3,4). For specific aspects related to DAEs see: [3, 15]. We will consider the "Compound-Fast" [20] version of the BDF methods because of stability reasons. The stability of this multirate method has been studied in [20]. The Compound-Fast method (Alg.1) first integrates (3) and (4) together with one large stepsize H, which results in $\mathbf{x}_A^n$ and $\mathbf{x}_L^n$. Afterwards, only equation (3) is re-integrated with a small stepsize h, while $\mathbf{x}_L$ is approximated by means of interpolation of order K. The implicit correctors $\mathbf{x}_L^n, \mathbf{x}_A^n$ and $\mathbf{x}_A^{n-1,m}$ satisfy the nonlinear equations (7),(8),(10) that has to be solved by e.g. the Newton method. As starting guess one uses the explicit predictors $\mathbf{y}_L^n, \mathbf{y}_A^n$ and $\mathbf{y}_A^{n-1,m}$ using extrapolation of order K and k, respectively.

ALGORITHM 1 *The BDF Compound-Fast multirate method (for $H_n = H, h_{n-1,m} = h$)*

Compound phase *Solve for $\boldsymbol{x}_L^n$ and $\boldsymbol{x}_A^n$:*

$$\rho_0 \boldsymbol{q}_A(T_n, \boldsymbol{x}_A^n, \boldsymbol{x}_L^n) + \ldots + \rho_K \boldsymbol{q}_A(T_{n-K}, \boldsymbol{x}_A^{n-K}, \boldsymbol{x}_L^{n-K}) + H\boldsymbol{j}_A(T_n, \boldsymbol{x}_A^n, \boldsymbol{x}_L^n) = \boldsymbol{0} \quad (7)$$

$$\rho_0 \boldsymbol{q}_L(T_n, \boldsymbol{x}_A^n, \boldsymbol{x}_L^n) + \ldots + \rho_K \boldsymbol{q}_L(T_{n-K}, \boldsymbol{x}_A^{n-K}, \boldsymbol{x}_L^{n-K}) + H\boldsymbol{j}_L(T_n, \boldsymbol{x}_A^n, \boldsymbol{x}_L^n) = \boldsymbol{0} \quad (8)$$

Refinement phase *Solve for $\boldsymbol{x}_A^{n-1,m}$ $(m = 1, \ldots, q)$:*

$$\bar{\rho}_0 \boldsymbol{q}_A(t_{n-1,m}, \boldsymbol{x}_A^{n-1,m}, \hat{\boldsymbol{x}}_L^{n-1,m}) + \ldots + \bar{\rho}_k \boldsymbol{q}_A(t_{n-1,m-k}, \boldsymbol{x}_A^{n-1,m-k}, \hat{\boldsymbol{x}}_L^{n-1,m-k}) + \quad (9)$$

$$h\boldsymbol{j}_A(t_{n-1,m}, \boldsymbol{x}_A^{n-1,m}, \hat{\boldsymbol{x}}_L^{n-1,m}) = \boldsymbol{0} \quad (10)$$

$$\hat{\boldsymbol{x}}_L^{n-1,m} - (\mu_0 \boldsymbol{x}_L^n + \ldots + \mu_K \boldsymbol{x}_L^{n-K}) = \boldsymbol{0} \quad (11)$$

Remark that $d_A \ll d_L$. If $d_A = \epsilon d_L, h = \nu H$ we can expect an efficiency gain $S \approx \frac{1}{\nu+\epsilon}$. It clearly depends on the dynamics and geometry (partition sizes).

1.3 Efficiency analysis of multirate methods

Although we introduced the Compound-Fast BDF algorithm, the next analysis is valid for a much larger family of multirate methods. Let W_C, W_R be the computational work per timestep for the compound phase and the refinement phase and define the workload ratio by $E := \frac{W_R}{W_C}$. Let W_S be the computational work per step for the standard singlerate BDF method of order K, which satisfies $\frac{W_S}{W_C} =: F \approx 1$. If H, h are the average compound and refinement steps and $q := \frac{H}{h}$ is the average multirate factor, then a multirate method on $[0, T]$ will need the following computational workload:

$$W_{\text{mult}} := W_R \frac{T}{h} + W_C \frac{T}{H} = W_C \; T \; (E\frac{1}{h} + \frac{1}{H}) = W_C \frac{T}{h}(E + \frac{1}{q}), \tag{12}$$

while a singlerate method, with step h_s, would need $W_{\text{sing}} := W_S \frac{T}{h_s}$. Although $h \approx h_s$, they are kept different because h has to be slightly smaller than h_s to obtain the same accuracy. Thus we have the following speed-up factor for the multirate method

$$S := \frac{W_{\text{sing}}}{W_{\text{mult}}} = \frac{W_S \frac{1}{h_s}}{W_C \frac{1}{h}(E + \frac{1}{q})} = F \frac{h}{h_s} \frac{1}{\frac{1}{q} + E} \approx \frac{h}{h_s} \frac{1}{\frac{1}{q} + E}. \tag{13}$$

Here q is the multirate factor which is large if the dynamics of the refined part are more active than the other slow part. The ratio E is determined by the geometric partition and describes the relative costs of a refinement step which depends on the size d_A of the refinement part. It follows that $S \to \frac{F}{E} \frac{h}{h_s}$ for $q \to \infty$, and $S \to F \; q \frac{h}{h_s}$ for $E \to 0$. Clearly, we get a large speed-up factor if q is large and E is small. Only if $S > 1$ it could be attractive to use for instance the multirate version of a certain integration scheme. This model (13) can be used for automatic or dynamical partitioning [18].

The multirate factor q can be approximated by the ratio between the proposed steps for the next step of a multirate and singlerate method $\hat{q} := \frac{H_{\text{new}}}{h_{\text{new}}}$. Here $H_{\text{new}}, h_{\text{new}}$ are the proposed stepsizes for the next compound step or singlerate step, respectively. Any integration method has an algorithm for $H_{\text{new}}, h_{\text{new}}$, which always depend on the integration orders K, k. In fact $\hat{q}$ depends on the estimated local error vector $\hat{\mathbf{e}}$ and, if $K = k$, approximately behaves like

$$\hat{q} \approx \left(\frac{\|\mathbf{B}_A \hat{\mathbf{e}}\|}{\|\mathbf{B}_L \hat{\mathbf{e}}\|} \right)^{\frac{1}{K+1}}. \tag{14}$$

The workload ratio E is approximated by

$$\hat{E} := \left(\frac{d_A}{d} \right)^{\alpha}, \tag{15}$$

where $\alpha \in (1, 3)$ depends on the application. By default we use $\alpha = 2$. Note that it is also possible to model E by a parameterized rational function of d_A and d, where the parameters can be identified by using experimental data. The value of α is also important for automatic partitioning algorithms [16].

1.4 Conditions for the partition

Because of the hierarchical structure of the functions $\mathbf{q}, \mathbf{j}$ it can be proved that the circuit model (1) has an unique solution. For a proper implementation of the previous multirate schemes, it is required that the solvability is also preserved for the active part. Furthermore, it is also very useful if the active part of a stable circuit model is also stable and has the same differential index as the original DAE. This implies that not all partitions of a DAE are allowed.

Consider the linear time-invariant system

$$\Sigma : \quad \mathbf{C}\dot{\mathbf{x}} + \mathbf{G}\mathbf{x} = \mathbf{u}. \tag{16}$$

It is well-known that

$$\begin{array}{rcl} \text{the system (16) is solvable} & \Leftrightarrow & \sigma(\Sigma) \text{ is a finite set,} \\ \text{the system (16) is stable} & \Leftrightarrow & \forall_{\lambda\in\sigma(\Sigma)}\mathrm{Re}[\lambda] < 0, \end{array}$$

where $\sigma(\Sigma) = \{\lambda \in \mathbb{C} : \det(\lambda\mathbf{C} + \mathbf{G}) = 0\}$. For a general partition these properties are not preserved for the active part of a DAE. For example, consider the linear 2-dimensional problem $\Sigma : \mathbf{C}\dot{\mathbf{x}} + \mathbf{G}\mathbf{x} = \mathbf{s}$, where

$$\mathbf{C} = \mathbf{G} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

This DAE is solvable because $\det(\lambda\mathbf{C} + \mathbf{G}) = -(\lambda + 1)^2$ which is only equal to zero for $\lambda = -1$, so $\sigma(\Sigma) = \{-1\}$ is a finite set. If we take the partition with

$$\mathbf{B}_A = \begin{pmatrix} 1 & 0 \end{pmatrix}, \quad \mathbf{B}_L = \begin{pmatrix} 0 & 1 \end{pmatrix},$$

we get for the refinement the unsolvable problem

$$0\dot{x}_A + 0x_A = s_1.$$

Notice that the active part of an ODE is always solvable, because then $\mathbf{C} = \mathbf{I}$ is an invertible matrix. However, the stability is not automatically preserved for both ODEs and DAEs. If we take

$$\mathbf{C} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \mathbf{G} = \begin{pmatrix} -1 & -2 \\ 2 & 2 \end{pmatrix},$$

we have a stable ODE with eigenvalues $-\frac{1}{2} \pm \frac{1}{2}\sqrt{7}i$, but for the refinement we get the following unstable differential equation

$$\dot{x}_A = x_A + s_1.$$

Finally it can be shown that also the index of the active part is not always preserved.

1.5 Overview of this paper

This paper deals with the implementation and the error control of the Compound-Fast multirate version of the BDF scheme. The Compound-Fast multirate version appears to be a numerically stable method compared to other semi-explicit multirate methods [17, 20]. We use BDF integration methods because they use less function evaluations and they are very well suited for interpolation. For linear multistep methods the solution can always be represented by a piecewise polynomial, which can be used to interpolate the latent interface variables without loss of accuracy.
The paper is organized as follows. Section 2 describes in detail how the BDF Compound-Fast method can be efficiently implemented by means of the Nordsieck data representation. Section 3 contains an analysis of the local discretization error and shows furthermore how this error can be controlled by the multirate stepsizes H, h. Then section 4 shows some numerical test examples, which show how the presented multirate method and error control work in practice. Finally, section 5 closes this paper with some concluding remarks.

2 BDF Compound-Fast multirate algorithm

This section describes how the BDF Compound-Fast multirate algorithm can be efficiently implemented by means of the Nordsieck data representation. The BDF method stores the previous numerical values in polynomial form. There are several manners to store and process polynomials, such as the Lagrange, scaled divided difference or Nordsieck representation. This topic has been studied very thoroughly in [11]. Because the Nordsieck representation is frequently used, we will investigate some properties of them. More details can be read in [6].

2.1 Introduction to the Nordsieck data representation

Let $\mathbf{p}(t) : \mathbb{R} \to \mathbb{R}^d$ be a vector-valued polynomial of degree k. This polynomial can be written like a truncated Taylor series around t_1:

$$\mathbf{p}(t) = \sum_{i=0}^{k} \frac{1}{i!} \frac{d^i}{dt^i} \mathbf{p}(t_1)(t - t_1)^i.$$

It is very common to expand $\mathbf{p}(t)$ in the following scaled form (for some $h > 0$):

$$\mathbf{p}(t) = \sum_{i=0}^{k} \left(\frac{h^i \frac{d^i}{dt^i} \mathbf{p}(t_1)}{i!} \right) \left(\frac{t - t_1}{h} \right)^i.$$

This polynomial could describe the time-behaviour of $\mathbf{x}(t)$ or $\mathbf{q}(t, \mathbf{x}(t))$ at a certain time-interval $[t_{n-1}, t_n]$, where $t_1 = t_n$ and $h = t_n - t_{n-1}$. The Nordsieck matrix $\bar{\mathbf{P}}(t_1, h) \in \mathbb{R}^{d \times (k+1)}$ of this polynomial contains all coefficients of this polynomial, i.e.

$$\bar{\mathbf{P}} = \left(\mathbf{p}(t_1), h \frac{d}{dt} \mathbf{p}(t_1), \frac{h^2}{2} \frac{d^2}{dt^2} \mathbf{p}(t_1), \ldots, \frac{h^k}{k!} \frac{d^k}{dt^k} \mathbf{p}(t_1) \right).$$

This matrix $\bar{\mathbf{P}}$ is called the transposed Nordsieck vector if the state dimension d is equal to one. Now, the vector-valued polynomial and its Nordsieck matrix are related by the next equation:

$$\mathbf{p}(t) = \sum_{i=0}^{k} \bar{\mathbf{p}}_{i+1} \left(\frac{t - t_1}{h} \right)^i, \tag{17}$$

where $\bar{\mathbf{p}}_i$ is the i-th column of $\bar{\mathbf{P}}$.

Definition 1 *For $1 \leq i \leq k+1$ the vector $\boldsymbol{e}(k, \omega) \in \mathbb{R}^{k+1}$ is defined by*

$$\boldsymbol{e}(k, \omega) := \left[1, \omega, \ldots, \omega^k \right]^T.$$

For $1 \leq i \leq k+1$ and $1 \leq j \leq k+1$ the matrix $\boldsymbol{A}(k, \omega, \mu) \in \mathbb{R}^{(k+1) \times (k+1)}$ is defined by

$$a_{ij} := \begin{cases} 0 & i < y, \\ \binom{i-1}{j-1} \omega^{i-1} \mu^{i-j} & i \geq j. \end{cases} \tag{18}$$

For $1 \leq i \leq k+1$ and $1 \leq j \leq l+1$ the non-square Vandermonde matrix $\boldsymbol{V}(k, l) \in \mathbb{R}^{(k+1) \times (l+1)}$ is defined by

$$v_{ij} := \begin{cases} 1 & i = j = 1, \\ (1 - j)^{i-1} & \textit{otherwise.} \end{cases}$$

Clearly, the Nordsieck matrix depends on the time-point t_1 and the stepsize h. This means that changing t_1 or h will also change the Nordsieck vector. If the Nordsieck matrix $\bar{\mathbf{P}}(t_1, h_1)$ is available, $\mathbf{p}(t)$ can be constructed as follows:

$$\mathbf{p}(t) = \sum_{i=0}^{k} \bar{\mathbf{p}}_{i+1} \left(\frac{t - t_1}{h_1} \right)^i = \bar{\mathbf{P}} \cdot \mathbf{e}(k, \frac{t - t_1}{h_1}). \tag{19}$$

Two Nordsieck matrices $\bar{\mathbf{P}}(t_1, h_1)$ and $\bar{\mathbf{P}}(t_2, h_2)$ are called equivalent if they represent the same polynomial. If one Nordsieck matrix $\bar{\mathbf{P}} := \bar{\mathbf{P}}(t_1, h_1)$ is known, all other Nordsieck matrices $\bar{\mathbf{Q}} := \bar{\mathbf{P}}(t_2, h_2)$ can also be computed. Theorem 1 shows that $\bar{\mathbf{Q}}$ can be computed by just a matrix multiplication.

Theorem 1 *Assume that $\bar{\boldsymbol{P}} := \bar{\boldsymbol{P}}(t_1, h_1)$ and $\bar{\boldsymbol{Q}} := \bar{\boldsymbol{P}}(t_2, h_2)$ are two equivalent Nordsieck matrices, which represent the same polynomial $\boldsymbol{p}(t)$. They use time-points t_1, t_2 and stepsizes h_1, h_2 respectively. Then the Nordsieck matrices are related by*

$$\bar{\boldsymbol{Q}} = \bar{\boldsymbol{P}} \cdot \boldsymbol{A}(k, \frac{h_2}{h_1}, \frac{t_2 - t_1}{h_2}), \tag{20}$$

where $\boldsymbol{A} \in \mathbb{R}^{(k+1)\times(k+1)}$ is defined in (18).

This Theorem can be used to transform the Nordsieck matrices if $\bar{\mathbf{P}}$ and $\bar{\mathbf{Q}}$ have the same number of columns k. In practice $\bar{\mathbf{P}}$ and $\bar{\mathbf{Q}}$ may also have different numbers of columns, k_1 and k_2. If the polynomial $\mathbf{p}(t)$ of degree k_1 is represented by $\bar{\mathbf{P}}$, it is only possible to describe it also with $\bar{\mathbf{Q}}$ if $k_2 \geq k_1$. Otherwise, $\mathbf{p}(t)$ can only be approximated by a lower degree polynomial $\mathbf{q}(t)$, which is represented by $\bar{\mathbf{Q}}$, such that $\mathbf{q}(t)$ looks like $\mathbf{p}(t)$ in the neighbourhood of t_2. Before we state the next theorem, we need the following Definition.

Definition 2 *For $1 \leq i \leq k+1$ and $1 \leq j \leq l+1$ the matrices $\boldsymbol{T}^{(1)}(k, l, \omega, \mu), \boldsymbol{T}^{(2)}(k, l, \omega, \mu) \in \mathbb{R}^{(k+1)\times(l+1)}$ are defined by*

$$\boldsymbol{T}^{(1)} := \boldsymbol{A}(k, \omega, \mu) \cdot \boldsymbol{V}(k, l) \cdot \boldsymbol{V}(l, l)^{-1} \tag{21}$$

$$\boldsymbol{T}^{(2)} := \boldsymbol{A}(k, \omega, \mu) \cdot \left[\boldsymbol{V}(k, l-1), \boldsymbol{e}_2^k \right] \cdot \left[\boldsymbol{V}(l, l-1), \boldsymbol{e}_2^l \right]^{-1} \tag{22}$$

where $\boldsymbol{e}_2^k, \boldsymbol{e}_2^l$ are the unit vectors $[0, 1, 0, \ldots]^T$ of length $k+1$ and $l+1$, respectively.

Theorem 2 *Consider the polynomial $\boldsymbol{p}(t)$ of degree k_1 which is represented by $\bar{\boldsymbol{P}} \in \mathbb{R}^{d\times(k_1+1)}$ using time-point t_1 and stepsize h_1. Let $\bar{\boldsymbol{Q}} \in \mathbb{R}^{d\times(k_2+1)}$ be the Nordsieck matrix of $\boldsymbol{q}(t)$ of degree k_2 with time-point t_2 and stepsize h_2. Then it holds that*

$$\boldsymbol{q}(t_2 - jh_2) = \boldsymbol{p}(t_2 - jh_2), \quad 0 \leq j \leq k_2 \quad \Leftrightarrow \quad \bar{\boldsymbol{Q}} = \bar{\boldsymbol{P}} \cdot \boldsymbol{T}^{(1)}(k_1, k_2, \frac{h_2}{h_1}, \frac{t_2 - t_1}{h_2}), \tag{23}$$

$$\begin{cases} \boldsymbol{q}(t_2 - jh_2) = \boldsymbol{p}(t_2 - jh_2), & 0 \leq j \leq k_2 - 1 \\ \frac{d}{dt}\boldsymbol{q}(t_2) = \frac{d}{dt}\boldsymbol{p}(t_2) \end{cases} \quad \Leftrightarrow \quad \bar{\boldsymbol{Q}} = \bar{\boldsymbol{P}} \cdot \boldsymbol{T}^{(2)}(k_1, k_2, \frac{h_2}{h_1}, \frac{t_2 - t_1}{h_2}). \tag{24}$$

Using the transformation matrix $\mathbf{T}^{(1)}$ is just similar to interpolation, while $\mathbf{T}^{(2)}$ guarantees a smooth transition between $\mathbf{p}$ and $\mathbf{q}$ around t_2.

2.2 Implementation of the BDF Compound-Fast multirate algorithm

Before we describe the proposed multirate algorithm in detail for both the compound step and the refinement phase, we have to define the required polynomials for the BDF algorithm. Firstly, the BDF integration of order K on the coarse time-grid needs the Lagrange basis polynomial $l^n(t)$ on $\{T_{n-K},\dots,T_n\}$, with

$$l^n(T_{n-j}) = \begin{cases} 1 & \text{if } j=0, \\ 0 & \text{if } j\neq 0. \end{cases}$$

Let $\bar{\mathbf{l}}^n \in \mathbb{R}^{(K+1)}$ be the corresponding Nordsieck vector [6] of $l^n(t)$ which can be expressed as

$$l^n(t) = \sum_{i=0}^{K} \bar{l}^n_{i+1} \left(\frac{t-T_n}{H_n}\right)^i.$$

Here $\bar{l}^n_{i+1}$ is the $i+1$-th element of the Nordsieck vector

$$\bar{\mathbf{l}}^n = \left(l^n(T_n), H_n\frac{d}{dt}l^n(T_n),\dots,\frac{H_n^K}{K!}\frac{d^K}{dt^K}l^n(T_n)\right)^T.$$

Furthermore, also the Nordsieck matrices $\bar{\mathbf{Y}}^n, \bar{\mathbf{X}}^n, \bar{\mathbf{P}}^n, \bar{\mathbf{Q}}^n \in \mathbb{R}^{d\times(K+1)}$ are needed, which represent the local predictor and corrector polynomials for $\mathbf{x}(t)$ and $\mathbf{q}(t,\mathbf{x}(t))$, respectively [6]. For instance, the predictor polynomial $\mathbf{y}^n(t)$ for $\mathbf{x}(t)$ on $[T_{n-1},T_n]$ satisfies

$$\mathbf{y}^n(t) = \sum_{i=0}^{K} \bar{\mathbf{y}}^n_{i+1} \left(\frac{t-T_n}{H_n}\right)^i.$$

Here $\bar{\mathbf{y}}^n_{i+1}$ is the $i+1$-th column of the Nordsieck matrix

$$\bar{\mathbf{Y}}^n = \left(\mathbf{y}^n(T_n), H_n\frac{d}{dt}\mathbf{y}^n(T_n),\dots,\frac{H_n^K}{K!}\frac{d^K}{dt^K}\mathbf{y}^n(T_n)\right).$$

The multirate BDF method also integrates the active part independently on a refined time-grid with order k. There it needs a different Lagrange basis polynomial $l^{n-1,m}(t)$ on $\{t_{n-1,m-k},\dots,t_{n-1,m}\}$, with

$$l^{n-1,m}(t_{n-1,m-j}) = \begin{cases} 1 & \text{if } j=0, \\ 0 & \text{if } j\neq 0. \end{cases}$$

Again, it also needs the refined Nordsieck vectors $\bar{\mathbf{Y}}_A^{n-1,m}, \bar{\mathbf{X}}_A^{n-1,m}, \bar{\mathbf{P}}_A^{n-1,m}, \bar{\mathbf{Q}}_A^{n-1,m} \in \mathbb{R}^{d_A\times(k+1)}$, which represent the local refined predictor and corrector polynomials for $\mathbf{x}(t)$ and $\mathbf{q}(t,\mathbf{x}(t))$, respectively. Now, the refined predictor polynomial $\mathbf{y}_A^{n-1,m}(t)$ for $\mathbf{x}_A(t)$ on $[t_{n-1,m-1},t_{n-1,m}]$ satisfies

$$\mathbf{y}_A^{n-1,m}(t) = \sum_{i=0}^{k} \bar{\mathbf{y}}^{n-1,m}_{A,i+1} \left(\frac{t-t_{n-1,m}}{h_{n-1,m}}\right)^i,$$

where

$$\bar{\mathbf{Y}}_A^{n-1,m} = \left(\mathbf{y}_A^{n-1,m}(t_{n-1,m}),\dots,\frac{h^k_{n-1,m}}{k!}\frac{d^k}{dt^k}\mathbf{y}_A^{n-1,m}(t_{n-1,m})\right).$$

Figure 1 shows the typical form of the predictor and corrector polynomials at the coarse and refined grids. The polynomials are just of degree one, which implies the use of linear extrapolation for the prediction. Clearly, the solution becomes smoother for higher degree polynomials.

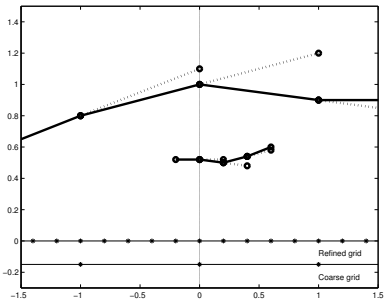


Figure 1: Typical form of the predictor (dashed) and corrector (solid) polynomials on the coarse and refined time-grids.

The compound step

During the compound step in fact a normal BDF step is done for the complete DAE. This means that the next algebraic system is solved:

$$\rho_0^n \mathbf{q}(T_n, \mathbf{x}_n) + H_n \mathbf{j}(T_n, \mathbf{x}_n) + \mathbf{b}_n = \mathbf{0}, \tag{25}$$

where ρ_0^n is an order-dependent parameter and $\mathbf{b}_n$ is a vector which represents the history of the numerical integration. We can easily express ρ_0^n and $\mathbf{b}_n$ in terms of $\bar{\mathbf{l}}^n$ and $\bar{\mathbf{P}}^n$ by $\rho_0^n = \bar{l}_2^n$ and $\mathbf{b}_n = \bar{\mathbf{p}}_2^n - \rho_0^n \bar{\mathbf{p}}_1^n$ [6]. Here $\bar{\mathbf{p}}_1^n$ and $\bar{\mathbf{p}}_2^n$ are just the first two columns of $\bar{\mathbf{P}}^n$, the Nordsieck matrix of the predictor polynomial $\mathbf{p}^n(t)$, which is set to be equal to $\mathbf{q}^{n-1}(t)$ if the integration order is the same as in the previous timestep. For the Nordsieck vectors this means that

$$\bar{\mathbf{P}}^n = \bar{\mathbf{Q}}^{n-1}\mathbf{T}_n, \quad \bar{\mathbf{Y}}^n = \bar{\mathbf{X}}^{n-1}\mathbf{T}_n, \tag{26}$$

where $\mathbf{T}_n = \mathbf{T}^{(*)}(K_n, K_{n-1}, \frac{H_n}{H_{n-1}}, 1)$ has been defined before in Definition 2. For a smooth transition at T_n one should use the second transformation matrix $\mathbf{T}^{(2)}$. Usually the initial guess for the solution of (25) is equal to the predictor value $\bar{\mathbf{y}}_1^n$, that is based on extrapolation of order K. Because the compound step will be much larger than the time-constant of the active part, we use a modified Newton scheme which relaxes the active part of the residual by using a small positive weighting factor in front of the active part. Thus only the latent part of $\mathbf{x}_n$ really has to converge, while the active part only has to be bounded. For a compound step, the active part of $\mathbf{x}_n$ still must be improved by the refinement phase. After the refinement, we use the updated $\mathbf{x}_n$ and $\mathbf{q}_n$ to correct the complete predictor polynomials $\mathbf{p}^n(t)$ and $\mathbf{y}^n(t)$:

$$\mathbf{q}^n(t) = \mathbf{p}^n(t) + (\mathbf{q}(T_n, \mathbf{x}_n) - \mathbf{p}^n(T_n))l^n(t), \tag{27}$$

$$\mathbf{x}^n(t) = \mathbf{y}^n(t) + (\mathbf{x}_n - \mathbf{y}^n(T_n))l^n(t). \tag{28}$$

For the Nordsieck vectors this means: $\bar{\mathbf{Q}}^n = \bar{\mathbf{P}}^n + (\mathbf{q}_n - \hat{\mathbf{q}}_n)\left(\bar{\mathbf{l}}^n\right)^T$, $\bar{\mathbf{X}}^n = \bar{\mathbf{Y}}^n + (\mathbf{x}_n - \hat{\mathbf{x}}_n)\left(\bar{\mathbf{l}}^n\right)^T$.

The refinement phase

In fact, the refinement solves a new initial value problem for a much smaller perturbed DAE. It solves for each time-point $t_{n-1,m}$ the nonlinear equation:

$$\bar{\rho}_0^{n-1,m}\mathbf{q}_A(t_{n-1,m}, \mathbf{x}_A^{n-1,m}, \hat{\mathbf{x}}_L^{n-1,m}) + h_{n-1,m}\mathbf{j}_A(t_{n-1,m}, \mathbf{x}_A^{n-1,m}, \hat{\mathbf{x}}_L^{n-1,m}) + \mathbf{b}_A^{n-1,m} = \mathbf{0}, \tag{29}$$

where $\hat{\mathbf{x}}_L^{n-1,m}$ is the interpolated latent part. We can compute $\hat{\mathbf{x}}_L^{n-1,m}$ from interpolation-based functions which are constructed between the compound step and the refinement phase. Another possibility is to compute it from the Nordsieck matrix $\bar{\mathbf{X}}^n$ if it is already corrected for the latent part. Then we can compute directly

$$\hat{\mathbf{x}}_L^{n-1,m} = \mathbf{B}_L \bar{\mathbf{X}}^n \mathbf{e}(K, \frac{t_{n-1,m} - T_n}{H_n}), \tag{30}$$

where we used the formula in (19). Furthermore $\bar{\rho}_0^{n-1,m}$ and $\mathbf{b}_A^{n-1,m}$ can be computed in a similar way as for the compound step: $\bar{\rho}_0^{n-1,m} = \bar{l}_2^{n-1,m}, \mathbf{b}_A^{n-1,m} = \bar{\mathbf{p}}_{A,2}^{n-1,m} - \bar{\rho}_0^{n-1,m}\bar{\mathbf{p}}_{A,1}^{n-1,m}$. The predictor Nordsieck matrices $\bar{\mathbf{P}}_A^{n-1,m}, \bar{\mathbf{Y}}_A^{n-1,m}$ are similarly computed by

$$\bar{\mathbf{P}}_A^{n-1,m} = \bar{\mathbf{Q}}_A^{n-1,m-1}\mathbf{T}_{n-1,m}, \quad \bar{\mathbf{Y}}_A^{n-1,m} = \bar{\mathbf{X}}_A^{n-1,m-1}\mathbf{T}_{n-1,m}, \tag{31}$$

where $\mathbf{T}_{n-1,m} = \mathbf{T}^{(*)}(k_{n-1,m}, k_{n-1,m-1}, \frac{h_{n-1,m}}{h_{n-1,m-1}}, 1)$ for $m > 0$. For $m = 0$ we need the matrices $\bar{\mathbf{Q}}_A^{n-2,q-1}, \bar{\mathbf{X}}_A^{n-2,q-1}$ which are only available for a static partition. Otherwise we need to transform a part of the coarse Nordsieck matrices $\bar{\mathbf{Q}}^{n-1}, \bar{\mathbf{X}}^{n-1}$.

In general, it is not very difficult to solve (29) because we can trust the predictor polynomial giving an accurate initial guess $\bar{\mathbf{y}}_{A,1}^{n-1,m}$. The solution is used to correct the predictor polynomials. For the Nordsieck representation we get: $\bar{\mathbf{Q}}_A^{n-1,m} = \bar{\mathbf{P}}_A^{n-1,m} + (\mathbf{q}_A(t_{n-1,m}, \mathbf{x}_A^{n-1,m}, \hat{\mathbf{x}}_L^{n-1,m}) - \hat{\mathbf{q}}_A) \left(\bar{\mathbf{l}}^{n-1,m}\right)^T$, $\bar{\mathbf{X}}_A^{n-1,m} = \bar{\mathbf{Y}}_A^{n-1,m} + (\mathbf{x}_A^{n-1,m} - \hat{\mathbf{x}}_A^{n-1,m}) \left(\bar{\mathbf{l}}^{n-1,m}\right)^T$. Finally, if $t_{n-1,m} = T_n$ the refined corrector polynomials are evaluated at T_n:

$$\mathbf{B}_A \mathbf{x}_n = \bar{\mathbf{X}}_A^{n-1,m} \mathbf{e}(k, \frac{T_n - t_{n-1,m}}{h_{n-1,m}}), \quad \mathbf{B}_A \mathbf{q}_n = \bar{\mathbf{Q}}_A^{n-1,m} \mathbf{e}(k, \frac{T_n - t_{n-1,m}}{h_{n-1,m}}), \tag{32}$$

where again we used the formula in (19).

3 Error control of BDF Compound-Fast multirate algorithm

The accuracy of a multirate method can be controlled by the stepsizes of the compound step and the refinement phase. This section will show how H_n and $h_{n-1,m}$ can be controlled such that the local error is smaller than a given tolerance level. We will analyze how the local discretization errors at the coarse and refined grid asymptotically behave. Afterwards an error model is constructed which is used to develop stepsize controllers for the coarse and refined time-grids.

3.1 Analysis of the local discretization error

In this section we generalize some techniques in [11, 13] to our multirate case. The local discretization error $\mathbf{d}^n \in \mathbb{R}^d$ is defined as the residual of the scheme at the coarse time-grid after inserting the exact solution. It still has the familiar behaviour $\mathbf{d}^n = O(H_n^{K+1})$. In practice the local discretization errors are not known and should be estimated up to a sufficient accuracy. The local discretization error $\mathbf{d}^n$ can be estimated by $\hat{\mathbf{d}}^n$ using the Nordsieck representation for $\mathbf{q}$:

$$\hat{\mathbf{d}}^n := \frac{-H_n}{T_n - T_{n-K-1}} \left[\bar{\mathbf{q}}_1^n - \bar{\mathbf{p}}_1^n\right]. \tag{33}$$

Now

$$\hat{r}_C^n := \|\hat{\mathbf{d}}_L^n\| + \tau \|\hat{\mathbf{d}}_A^n\| \tag{34}$$

is the used weighted error norm for the coarse grid, which must satisfy $\hat{r}_C^n < \mathrm{TOL}_C$. Here $\tau \geq 0$ is a small non-negative relaxation number which must improve the convergence of the compound step. The optimal value is not known yet, but in our experiments $\tau = 0$ worked satisfactory.

For multirate methods we also need to consider the local discretization error $\mathbf{d}_A^{n,m} \in \mathbb{R}^{d_A}$ at the refined time-grid. At the refined time-grid, at the interface between latent and active areas, the DAE has been perturbed by the interpolated latent variables. The refined local discretization error equals

$$\mathbf{d}_A^{n,m} = \bar{\rho}_0^{n-1,m} \mathbf{q}_A(t_{n-1,m}, \mathbf{x}_A(.), \mathbf{x}_L(.)) + h_{n-1,m} \mathbf{j}_A(t_{n-1,m}, \mathbf{x}_A(.), \mathbf{x}_L(.)) + \mathbf{b}_A^{n-1,m},$$

where

$$\mathbf{b}_A^{n-1,m} = \bar{\rho}_1^{n-1,m} \mathbf{q}_A(t_{n-1,m-1}, \mathbf{x}_A(.), \mathbf{x}_L(.)) + \ldots + \bar{\rho}_k^{n-1,m} \mathbf{q}_A(t_{n-1,m-k}, \mathbf{x}_A(.), \mathbf{x}_L(.)).$$

We also consider the perturbed local discretization error $\tilde{\mathbf{d}}_A^{n,m}$, which is the residual of the refinement scheme with fixed interpolated slow part after inserting the exact active solution. Clearly, $\tilde{\mathbf{d}}_A^{n,m}$ satisfies

$$\tilde{\mathbf{d}}_A^{n,m} = \bar{\rho}_0^{n-1,m}\mathbf{q}_A(t_{n-1,m}, \mathbf{x}_A(.), \hat{\mathbf{x}}_L^{n-1,m}) + h_{n-1,m}\mathbf{j}_A(t_{n-1,m}, \mathbf{x}_A(.), \hat{\mathbf{x}}_L^{n-1,m}) + \tilde{\mathbf{b}}_A^{n-1,m},$$

where

$$\tilde{\mathbf{b}}_A^{n-1,m} = \bar{\rho}_1^{n-1,m}\mathbf{q}_A(t_{n-1,m-1}, \mathbf{x}_A(.), \hat{\mathbf{x}}_L^{n-1,m-1}) + \ldots + \bar{\rho}_k^{n-1,m}\mathbf{q}_A(t_{n-1,m-k}, \mathbf{x}_A(.), \hat{\mathbf{x}}_L^{n-1,m-k}).$$

The perturbed local discretization error $\tilde{\mathbf{d}}_A^{n-1,m}$ behaves as $O(h_{n-1,m}^{k+1})$ and can be estimated in a similar way as $\mathbf{d}^n$ from the perturbed Nordsieck matrices of the refinement phase.

$$\hat{\tilde{\mathbf{d}}}_A^{n,m} := \frac{-h_{n-1,m}}{t_{n-1,m} - t_{n-1,m-k-1}}\mathbf{B}_A\left[\bar{\mathbf{q}}_1^{n-1,m} - \bar{\mathbf{p}}_1^{n-1,m}\right]. \tag{35}$$

The norm of $\|\hat{\tilde{\mathbf{d}}}_A^{n,m}\|$ will be denoted by

$$\hat{r}_A^{n-1,m} := \|\hat{\tilde{\mathbf{d}}}_A^{n,m}\|. \tag{36}$$

For singlerate methods the accuracy can completely be controlled by controlling the local discretization errors. This is no longer the case for multirate methods, where also the interpolation errors play an important role. Let $\hat{\mathbf{x}}_L(t)$ be the interpolation-based function which is exact at the coarse or refined time-grid. Then the interpolation errors $\mathbf{r}^n, \mathbf{r}^{n-1,m}$ are defined as the maximum errors of $\hat{\mathbf{x}}_L(t)$ between the time-points of the coarse and refined time-grids, respectively. If the interpolation order is equal to the integration order, they again have the familiar behaviour $\mathbf{r}^n = O(H_n^{K+1}), \mathbf{r}^{n-1,m} = O(h_{n-1,m}^{k+1})$.

Lemma 1 *For the interpolation errors $\boldsymbol{r}^n, \boldsymbol{r}^{n-1,m}$ it applies that*

$$\|\boldsymbol{r}^n\| \le \|\hat{\boldsymbol{r}}^n\|, \quad \|\boldsymbol{r}^{n-1,m}\| \le \|\hat{\boldsymbol{r}}^{n-1,m}\|, \tag{37}$$

where

$$\hat{\boldsymbol{r}}^n := \frac{1}{4}\frac{-H_n}{T_n - T_{n-K-1}}(\bar{\boldsymbol{x}}_1^n - \bar{\boldsymbol{y}}_1^n), \quad \hat{\boldsymbol{r}}^{n-1,m} := \frac{1}{4}\frac{-h_{n-1,m}}{t_{n-1,m} - t_{n-1,m-k-1}}(\bar{\boldsymbol{x}}_1^{n-1,m} - \bar{\boldsymbol{y}}_1^{n-1,m}). \tag{38}$$

Proof Let $\mathbf{x}(t)$ be the exact solution and $\hat{\mathbf{x}}$ a polynomial of degree K which interpolates $\mathbf{x}$ at the previous K time-points. At the coarse time-grid we have the following asymptotic behaviour:

$$\begin{aligned} \mathbf{x}(t) - \hat{\mathbf{x}}(t) &= (t - T_{n-K})\cdots(t - T_{n-1})(t - T_n)\frac{\mathbf{x}^{(K+1)}(\tau)}{(K+1)!}, \quad \tau \in (T_{n-K}, T_n) \\ &= (t - T_{n-K})\cdots(t - T_{n-1})(t - T_n)\frac{\mathbf{x}^{(K+1)}(T_n)}{(K+1)!} + O(H_n^{K+2}). \end{aligned} \tag{39}$$

We easily derive the upper bound $\max_{t\in[T_{n-1},T_n]}\|\mathbf{x}(t) - \hat{\mathbf{x}}(t)\|$ for all $t \in [T_{n-1}, T_n]$, which satisfies up to $O(H_n^{K+2})$:

$$\begin{aligned} \max_{\mathrm{t}\in[\mathrm{T_{n-1}},\mathrm{T_n}]}\|\mathbf{x}(t) - \hat{\mathbf{x}}(t)\| &\le \textstyle\prod_{j=2}^{K}(T_n - T_{n-j})\max_{\mathrm{t}\in[\mathrm{T_{n-1}},\mathrm{T_n}]}|(t - T_{n-1})(t - T_n)|\|\frac{\mathbf{x}^{(K+1)}(T_n)}{(K+1)!}\| \\ &= \tfrac{1}{4}\textstyle\prod_{j=2}^{K}(T_n - T_{n-j})(T_n - T_{n-1})^2\|\frac{\mathbf{x}^{(K+1)}(T_n)}{(K+1)!}\| \\ &= \|\tfrac{1}{4}H_n^2(H_{n-1} + H_n)\cdots(H_{n-K+1} + \cdots + H_n)\frac{\mathbf{x}^{(K+1)}(T_n)}{(K+1)!}\|. \end{aligned} \tag{40}$$

Because

$$\|\frac{1}{4}H_n^2(H_{n-1}+H_n)\cdots(H_{n-K+1}+\cdots+H_n)\frac{\mathbf{x}^{(K+1)}(T_n)}{(K+1)!}\| = \frac{H_n}{4(T_n-T_{n-K-1})}\|\bar{\mathbf{x}}_1^n-\bar{\mathbf{y}}_1^n\|+O(H_n^{K+2}),$$

it follows that

$$\max_{t\in[T_{n-1},T_n]}\|\mathbf{x}(t)-\hat{\mathbf{x}}(t)\| \leq \frac{H_n}{4(T_n-T_{n-K-1})}\|\bar{\mathbf{x}}_1^n-\bar{\mathbf{y}}_1^n\|. \tag{41}$$

Now it indeed follows that $\|\mathbf{r}^n\| \leq \|\hat{\mathbf{r}}^n\|$ if $\hat{\mathbf{r}}^n = \frac{1}{4}\frac{-H_n}{T_n-T_{n-K-1}}(\bar{\mathbf{x}}_1^n-\bar{\mathbf{y}}_1^n)$. At the refined time-grid we can prove in a similar way that also $\|\mathbf{r}^{n-1,m}\| \leq \|\hat{\mathbf{r}}^{n-1,m}\|$ where $\hat{\mathbf{r}}^{n-1,m}$ is given in (38). □

Before we state the following theorem, we need to define the coupling matrix.

Definition 3 *The coupling matrix* $\boldsymbol{K}_{n-1,m} \in \mathbb{R}^{d_A\times d_L}$ *is defined by*

$$\boldsymbol{K}_{n-1,m} := \boldsymbol{B}_A\frac{d}{dt}\left[\boldsymbol{C}(t_{n-1,m},\boldsymbol{x}(t_{n-1,m}))\right]\boldsymbol{B}_L^T + \boldsymbol{B}_A\boldsymbol{G}(t_{n-1,m},\boldsymbol{x}(t_{n-1,m}))\boldsymbol{B}_L^T, \tag{42}$$

where $\boldsymbol{C}(t,\boldsymbol{x}) = \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{x}}(t,\boldsymbol{x})$, $\boldsymbol{G}(t,\boldsymbol{x}) = \frac{\partial \boldsymbol{j}}{\partial \boldsymbol{x}}(t,\boldsymbol{x})$.

This coupling matrix appears to be essential for the error analysis.

Theorem 3 *The error* $\boldsymbol{d}_A^{n-1,m}$ *of the active part satisfies*

$$\|\boldsymbol{d}_A^{n-1,m}\| \leq \|\tilde{\boldsymbol{d}}_A^{n-1,m}\| + h_{n-1,m}\|\boldsymbol{K}_{n-1,m}\|\|\boldsymbol{B}_L\boldsymbol{r}^{n-1,m}\|. \tag{43}$$

Proof From now on we use the abbreviations $t = t_{n-1,m}, h = h_{n-1,m}, \bar{\rho}_0 = \bar{\rho}_0^{n-1,m}, \tilde{\mathbf{b}}_A = \tilde{\mathbf{b}}_A^{n-1,m}, \mathbf{b} = \mathbf{b}_{n-1,m}$. Clearly, the errors $\mathbf{d}^{n,m}, \tilde{\mathbf{d}}^{n-1,m}$ satisfy the following relationship

$$\begin{aligned}\|\mathbf{d}_A^{n-1,m}-\tilde{\mathbf{d}}_A^{n-1,m}\| &= \|\bar{\rho}_0(\mathbf{q}_A(t,\mathbf{x}_A(t),\mathbf{x}_L(t)) - \mathbf{q}_A(t,\mathbf{x}_A(t),\hat{\mathbf{x}}_L^{n-1,m}))\\ &+ h(\mathbf{j}_A(t,\mathbf{x}_A(t),\mathbf{x}_L(t)) - \mathbf{j}_A(t,\mathbf{x}_A(t),\hat{\mathbf{x}}_L^{n-1,m})) + \mathbf{b}_A - \tilde{\mathbf{b}}_A\|\\ &\leq \|\bar{\rho}_0\frac{\partial \mathbf{q}_A}{\partial \mathbf{x}_L} + h\frac{\partial \mathbf{j}_A}{\partial \mathbf{x}_L} + \frac{\partial \mathbf{b}_A}{\partial \mathbf{x}_L}\|\max_{0\leq j\leq K}\|\hat{\mathbf{x}}_L^{n-1,m-j} - \mathbf{x}_L(t_{n-1,m-j})\|.\end{aligned} \tag{44}$$

Here we assumed that

$$\|\tilde{\mathbf{b}}_A - \mathbf{b}_A\| \leq \|\frac{\partial \mathbf{b}_A}{\partial \mathbf{x}_L}\|\max_{0\leq j\leq K}\|\hat{\mathbf{x}}_L^{n-1,m-j} - \mathbf{x}_L(t_{n-1,m-j}))\|. \tag{45}$$

Note that $\mathbf{K}_{n-1,m}$ satisfies

$$\mathbf{K}_{n-1,m} \doteq \frac{\bar{\rho}_0}{h}\frac{\partial \mathbf{q}_A}{\partial \mathbf{x}_L}(t,\mathbf{x}_A(t),\mathbf{x}_L(t)) + \frac{\partial \mathbf{j}_A}{\partial \mathbf{x}_L}(t,\mathbf{x}_A(t),\mathbf{x}_L(t)) + \frac{1}{h}\frac{\partial \mathbf{b}_A}{\partial \mathbf{x}_L}(t,\mathbf{x}_A(t),\mathbf{x}_L(t)). \tag{46}$$

Because of Lemma 1 we get

$$\|\mathbf{d}_A^{n-1,m} - \tilde{\mathbf{d}}_A^{n-1,m}\| \leq h\|\mathbf{K}_{n-1,m}\|\|\mathbf{r}_L^{n-1,m}\|. \tag{47}$$

From (47) it immediately follows that (43) is fulfilled. □

Theorem 3 states that the interpolation error because of the interface equals $\|\mathbf{K}_{n-1,m}\|\|\mathbf{r}_L^{n-1,m}\|$. We estimate an upper bound at the coarse time-grid by

$$\hat{r}_I^n := \|\hat{\mathbf{K}}_n\|\|\hat{\mathbf{r}}_L^n\|. \tag{48}$$

Table 1: Multirate error estimates for several types of error estimation.

$\hat{r}$	local discr. error	local scaled error	combined error
$\hat{r}_C^n$	$\|\hat{\mathbf{d}}_L^n\| + \tau\|\hat{\mathbf{d}}_A^n\|$	$\|\hat{\mathbf{e}}_L^n\| + \tau\|\hat{\mathbf{e}}_A^n\|$	$\|\hat{\mathbf{r}}_L^n\| + \|\hat{\mathbf{e}}_L^n\| + \tau(\|\hat{\mathbf{r}}_A^n\| + \|\hat{\mathbf{e}}_A^n\|)$
$\hat{r}_I^n$	$\|\hat{\mathbf{K}}_n\|\|\hat{\mathbf{r}}_L^n\|$	$\|[\mathbf{B}_A\mathbf{J}_n\mathbf{B}_A^T]^{-1}\hat{\mathbf{K}}_n\|\|\hat{\mathbf{r}}_L^n\|$	$\|[\mathbf{B}_A\mathbf{J}_n\mathbf{B}_A^T]^{-1}\hat{\mathbf{K}}_n\|\|\hat{\mathbf{r}}_L^n\|$
$\hat{\tilde{r}}_A^{n-1,m}$	$\|\hat{\tilde{\mathbf{d}}}_A^{n-1,m}\|$	$\|\hat{\tilde{\mathbf{e}}}_A^{n-1,m}\|$	$\|\hat{\mathbf{r}}_A^{n-1,m}\| + \|\hat{\tilde{\mathbf{e}}}_A^{n-1,m}\|$

Here the coupling matrix $\mathbf{K}$ could be discretized at the coarse time-grid as follows

$$\mathbf{K}_{n-1,m} \doteq \frac{1}{H_n}\mathbf{B}_A\left[\mathbf{C}(T_n,\mathbf{x}^n) - \mathbf{C}(T_{n-1},\mathbf{x}^{n-1})\right]\mathbf{B}_L^T + \mathbf{B}_A\mathbf{G}(T_n,\mathbf{x}^n)\mathbf{B}_L^T =: \hat{\mathbf{K}}_n. \tag{49}$$

Now $\|\mathbf{d}_A^{n-1,m}\|$ can be bounded by the local error bound

$$\hat{r}_A^{n-1,m} := \hat{\tilde{r}}_A^{n-1,m} + h_{n-1,m}\hat{r}_I^n. \tag{50}$$

It is also possible to work with other error definitions than the local discretization error, like the local scaled error or the interpolation error. Let $\mathbf{J}_n = \rho_0^n\mathbf{C}(.) + H_n\mathbf{G}(.)$, then the local scaled error estimate satisfies

$$\hat{\mathbf{e}}_n := \mathbf{J}_n^{-1}\hat{\mathbf{d}}_n. \tag{51}$$

Besides the discretization errors also the interpolation errors could be included, which results in the next combined error estimate

$$\hat{\sigma}_n := \frac{1}{4}\|\hat{\mathbf{r}}_n\| + \|\hat{\mathbf{e}}_n\|. \tag{52}$$

The last error estimate could be useful for the long-term behaviour of stiff circuits which approximately satisfy $\mathbf{Gx} = \mathbf{s}$.

For all types it holds that $\hat{r}_C^n = O(H_n^{K+1})$ and $\hat{\tilde{r}}_A^{n-1,m} = O(h_{n-1,m}^{k+1})$ can be estimated and controlled by the already existing error control mechanism of the normal transient simulation. However, for the interpolation error $\hat{r}_I^n$ we have to analyze the equation (43) more profoundly. Recall that the local discretization error satisfies

$$\|\mathbf{d}_A^{n-1,m}\| \le \|\tilde{\mathbf{d}}_A^{n-1,m}\| + h_{n-1,m}\|\mathbf{K}_{n-1,m}\|\|\mathbf{r}_L^{n-1,m}\|.$$

For

$$\tilde{\mathbf{e}}_A^{n-1,m} := [\mathbf{B}_A\mathbf{J}_{n-1,m}\mathbf{B}_A^T]^{-1}\tilde{\mathbf{d}}_A^{n-1,m} \tag{53}$$

it is easy to prove that

$$\|\mathbf{e}_A^{n-1,m}\| \quad \le \quad \|\tilde{\mathbf{e}}_A^{n-1,m}\| + h_{n-1,m}\|[\mathbf{B}_A\mathbf{J}_{n-1,m}\mathbf{B}_A^T]^{-1}\mathbf{K}_{n-1,m}\|\|\mathbf{r}_L^{n-1,m}\|. \tag{54}$$

Because of the hierarchical structure of circuit models the original circuit is always solvable, which implies that $\mathbf{J}_n$ is invertible. Here we assume that $\mathbf{B}_A\mathbf{J}_{n-1,m}\mathbf{B}_A^T$ is an invertible matrix which can be approximated by $\mathbf{B}_A\mathbf{J}_n\mathbf{B}_A^T$, which is only the case if the active part is a solvable system. Conditions for this property have been given in paragraph 1.4. Table 1 shows the error estimates for the investigated error types.

3.2 Error control

Adaptive stepsize control of H_n and $h_{n,m}$ can be used to keep $\hat{r}_I^n = O(H_n^{K+1})$ and $\hat{r}_A^{n-1,m} = O(h_{n-1,m}^{k+1})$ close to $\theta\mathrm{TOL}_I$ and $\theta\tilde{\mathrm{TOL}}_A$ respectively, where $0 < \theta < 1$ is a safety factor. The

definitions of these error estimates can be found in (34,36,48,50) of the previous paragraph. In [9, 11, 17, 19] one can read how control theory can be applied to design proper stepsize controllers. For the Compound-Fast method we assumed the following error model, where $\hat{r}_* = \|\hat{\mathbf{d}}_*\|, \phi_* = \|\hat{\Phi}(.,.)\|$.

$$\begin{aligned} \hat{r}_C^n &= \phi_C^n H_n^{K+1}, && (55)\\ \hat{r}_A^{n,m} &= \hat{\hat{r}}_A^{n,m} + h_{n,m}\hat{r}_I^n, && (56)\\ \hat{\hat{r}}_A^{n,m} &= \tilde{\phi}_A^{n,m} h_{n,m}^{k+1}, && (57)\\ \hat{r}_I^n &= \phi_I^n H_n^{K+1}. && (58) \end{aligned}$$

By means of the stepsizes H_n and $h_{n,m}$ the goal is to satisfy the following error bounds:

$$\begin{aligned} \hat{r}_C^n &\leq \mathrm{TOL}_C, && (59)\\ \hat{\hat{r}}_A^{n,m} &\leq \mathrm{TOL}_A. && (60) \end{aligned}$$

Because $\hat{r}_C^n$ can be measured and only depends on H_n, this value is completely controlled by H_n. The second condition is more difficult to satisfy, because $\hat{r}_A^{n,m}$ can not be estimated directly, while it depends on H_n and $h_{n,m}$ both. Note that the term $h_{n,m}\hat{r}_I^n$ depends on $h_{n,m}$ and H_n, because EPUS (Error Per Unit Step) control is used. For EPS control, the factor $h_{n,m}$ can be removed. Nevertheless, it appears that EPS control is much more unstable from a numerical point of view. Therefore, we restrict ourselves to EPUS control. Let $h_{\max}$ be the maximum allowed step in the refinement phase. If

$$\begin{aligned} \hat{\hat{r}}_A^{n,m} &\leq \tilde{\mathrm{TOL}}_A = (1-w)\mathrm{TOL}_A && (61)\\ h_{\max}\hat{r}_I^n &\leq \mathrm{TOL}_I = w\mathrm{TOL}_A && (62) \end{aligned}$$

where $w \in (0,1)$ is a properly chosen balance number, we have

$$\hat{r}_A^{n,m} \leq \hat{\hat{r}}_A^{n,m} + h_{\max}\hat{r}_I^n < \mathrm{TOL}_A.$$

Now $\hat{\hat{r}}_A^{n,m}$ can be controlled by $h_{n,m}$ and $\hat{r}_I^n$ can be controlled by H_n. It is also possible to apply other techniques than this one, like linearization or alternative models. On the next pages we will derive which value of w is optimal.

The stepsize controllers can be based on the derived error model. The compound steps are used to track $\hat{r}_C^n$ and $\hat{r}_I^n$ close to TOL_C and TOL_I, respectively. Furthermore the errors $\hat{\hat{r}}_A^{n,m}$ are controlled by the refined time-grid in order that it is close to close to $\tilde{\mathrm{TOL}}_A$. The next elementary multirate stepsize controllers could be used:

$$\begin{aligned} H_{C,n+1} &= \left(\frac{\theta\mathrm{TOL}_C}{\hat{r}_C^n}\right)^{\frac{1}{K+1}} H_n, && (63)\\ H_{I,n+1} &= \left(\frac{\theta\mathrm{TOL}_I}{h_{\max}\hat{r}_I^n}\right)^{\frac{1}{K+1}} H_n, && (64)\\ h_{n,m+1} &= \left(\frac{\theta\tilde{\mathrm{TOL}}_A}{\hat{\hat{r}}_A^{n,m}}\right)^{\frac{1}{k+1}} h_{n,m}. && (65) \end{aligned}$$

Here, $\theta \in (0,1)$ is a safety factor which reduces the number of rejected timesteps. The next compound step is computed as follows:

$$H_{n+1} = \min\{H_{C,n+1}, H_{I,n+1}\}. \quad (66)$$

One also can use other more advanced stepsize controllers, e.g. digital linear stepsize controllers [13].

3.3 Optimal value for the balance number

It appears that the balance number w in (61),(62) can be chosen in an optimal way. This optimal value is called $w_{\text{opt}} \in (0,1)$. First we need the following Lemma.

Lemma 2 *Let $f \in \mathcal{C}^\infty((0,1),\mathbb{R}^+)$ such that*

$$\forall x \in (0,1) \;\; f(x) = A\,(1-x)^{-\frac{1}{r}} + B\,x^{-\frac{1}{s}},$$

where $A, B \in \mathbb{R}^+$ and $r, s \in \mathbb{N}$. Then it holds that

$$\forall x \in (0,1) \;\; f(x) \geq f(x_*),$$

where $x_* \in (0,1)$ is the unique solution of

$$A\,s\,x_*^{1+\frac{1}{s}} = B\,r\,(1-x_*)^{1+\frac{1}{r}}.$$

If $r = s$ one explicitly has

$$x_* = \frac{1}{1+\left(\frac{A}{B}\right)^{\frac{r}{1+r}}} \in (0,1),$$

such that

$$\forall x \in (0,1) \;\; f(x) \geq A\left(1+\left(\frac{A}{B}\right)^{\frac{-r}{1+r}}\right)^{\frac{1}{r}} + B\left(1+\left(\frac{A}{B}\right)^{\frac{r}{1+r}}\right)^{\frac{1}{r}}. \tag{67}$$

Proof Because $f \in \mathcal{C}^\infty((0,1),\mathbb{R}^+)$ and is unbounded for $x \in \{0,1\}$, f has at least one minimum $x_* \in (0,1)$, which satisfies $f'(x_*) = 0$. Because on $(0,1)$ the part $A(1-x)^{-\frac{1}{r}}$ is monotonously decreasing and the part $Bx^{-\frac{1}{s}}$ is monotonously increasing it can be proved that this minimum is unique. Because

$$f'(x) = \frac{A\,s\,x^{1+\frac{1}{s}} - B\,r\,(1-x)^{1+\frac{1}{r}}}{r\,s\,x^{1+\frac{1}{s}}(1-x)^{1+\frac{1}{r}}},$$

we get the following equation for x_*:

$$A\,s\,x_*^{1+\frac{1}{s}} = B\,r(1-x_*)^{1+\frac{1}{r}}.$$

If $r = s$ it is easily derived that

$$\frac{x_*}{1-x_*} = \left(\frac{B}{A}\right)^{\frac{r}{r+1}}, \text{ or } x_{**} = \frac{\left(\frac{B}{A}\right)^{\frac{r}{r+1}}}{1+\left(\frac{B}{A}\right)^{\frac{r}{r+1}}} = \frac{1}{1+\left(\frac{A}{B}\right)^{\frac{r}{r+1}}}.$$

Then indeed (67) follows immediately. □

Theorem 4 *Let w_{max} be defined by*

$$w_{max} := \frac{TOL_C}{TOL_A}\frac{h_{max}\hat{r}_I^n}{\hat{r}_C^n} \tag{68}$$

and assume that $w_* \in (0,1)$ solves

$$G(K+1)w_*^{1+\frac{1}{K+1}} = (k+1)(1-w_*)^{1+\frac{1}{k+1}}, \tag{69}$$

where $G := E\,q\left(\frac{\theta TOL_A}{\hat{\bar{r}}_A}\right)^{\frac{-1}{k+1}}\left(\frac{\theta TOL_A}{h_{max}\hat{r}_I}\right)^{\frac{1}{K+1}}$. Then the optimal value for $w \in (0,1)$ equals

$$w_{opt} = \min\{w_{max}, w_*\}. \tag{70}$$

Proof There exists a number $w_{\max}$ such that for $w \leq w_{\max}$ the first constraint (62) becomes dominant. This means that w always must be smaller than $w_{\max}$ because otherwise the tolerance level $\tilde{\mathrm{TOL}}_A = (1-w)\mathrm{TOL}_A$ for the refinement becomes too small. This value $w_{\max}$ can be determined by the property

$$\frac{\mathrm{TOL}_C}{\hat{r}_C^n} = \frac{w_{\max}\mathrm{TOL}_A}{h_{\max}\hat{r}_I^n},$$

which indeed implies (68).

If $w \leq w_{\max}$ we consider the computational work load $W_{\mathrm{mult}} = W_{\mathrm{mult}}(w)$ for the multirate method, which has been defined in (12). Because of the error models (57) and (58) and the definition of $w_{\max}$ we have for $w < w_{\max}$

$$h = \left(\frac{(1-w)\theta\mathrm{TOL}_A}{\hat{\tilde{\phi}}_A}\right)^{\frac{1}{k+1}}, \quad H = \left(\frac{w\,\theta\mathrm{TOL}_A}{h_{\max}\hat{\phi}_I}\right)^{\frac{1}{K+1}}. \tag{71}$$

Thus

$$W_{\mathrm{mult}} = W_R T\left(\frac{(1-w)\theta\mathrm{TOL}_A}{\hat{\tilde{\phi}}_A}\right)^{\frac{-1}{k+1}} + W_C T\left(\frac{w\,\theta\mathrm{TOL}_A}{h_{\max}\hat{\phi}_I}\right)^{\frac{-1}{K+1}} = D_R(1-w)^{\frac{-1}{k+1}} + D_C w^{\frac{-1}{K+1}}, \tag{72}$$

where

$$D_R = W_R T\left(\frac{\theta\mathrm{TOL}_A}{\hat{\tilde{\phi}}_A}\right)^{\frac{-1}{k+1}}, \quad D_C = W_C T\left(\frac{\theta\mathrm{TOL}_A}{h_{\max}\hat{\phi}_I}\right)^{\frac{-1}{K+1}}.$$

Because $w < w_{\max}$, it follows that

$$\frac{D_R}{D_C} = \frac{W_R}{W_C}\frac{H}{h}\left(\frac{\theta\mathrm{TOL}_A}{\hat{\tilde{r}}_A}\right)^{\frac{-1}{k+1}}\left(\frac{\theta\mathrm{TOL}_A}{h_{\max}\hat{r}_I}\right)^{\frac{1}{K+1}}.$$

Clearly, W_{mult} is unbounded for $w = 0$ or $w = 1$, which implies that $w \in (0,1)$. We apply Lemma 2 to (72) in order to find the optimal balance number w_*. Thus it follows for W_{mult}, where $A = D_R, B = D_C$ and $r = k+1, s = K+1$ that in $(0, w_{\max})$ the optimal value w_* indeed solves (69) with $G = \frac{D_R}{D_C}$. □

If $K = k$ we are able to compute the analytical solution of (69), which is equal to

$$w_* = \frac{1}{1 + G^{\frac{K+1}{K+2}}} \in (0,1).$$

Because then

$$G = E\,q\left(\frac{\hat{\tilde{r}}_A}{h_{\max}\hat{r}_I}\right)^{\frac{1}{K+1}},$$

we have

$$w_* = \frac{1}{1 + (E\,q)^{\frac{K+1}{K+2}}\left(\frac{\hat{\tilde{r}}_A}{h_{\max}\hat{r}_I}\right)^{\frac{1}{K+1}}}. \tag{73}$$

From this expression we can conclude that the optimal value w_* becomes very small if $q \gg 1$ and $\hat{\tilde{r}}_A \gg h_{\max}\hat{r}_I$ which is the case if the active part behaves much faster and is nearly decoupled. Note that an upper bound for H disturbs this optimality because then the steps do not satisfy (71). If H has to be limited it is always better to decrease w such that the limited H exactly fits for our case. Then the refinement can be done with larger steps.

4 Numerical experiments

In order to test the previous theoretical results we applied it to a large number of test examples both in MATLAB and in Pstar[3]. First we show some typical features of the multirate methods for two small linear test examples. Afterwards, we show the results for four circuit models, see Figure 4. The last two circuit models are real-world circuit designs which have been modeled and simulated with the multirate implementation within Pstar. More information about these numerical experiments, where also dynamical partitioning techniques are used, can be found in [16, 18, 21].

4.1 Linear test examples

First we look at a circuit model satisfying the next system of DAEs

$$\begin{array}{lrcl}
V_1: & \frac{V_1}{R_1} + C_1(\dot{V}_1 - \dot{V}_2) + j_1 & = & 0,\\
V_2: & C_1(\dot{V}_2 - \dot{V}_1) + \frac{1}{R_2}(V_2 - V_3) & = & 0,\\
V_3: & \frac{1}{R_2}(V_3 - V_2) - i_E^1 & = & 0,\\
V_4: & i_E^1 - i_E^2 & = & 0,\\
V_5: & i_E^2 + \frac{1}{R_3}(V_5 - V_6) & = & 0,\\
V_6: & \frac{1}{R_3}(V_6 - V_5) - i_L & = & 0,\\
V_7: & i_L + C_2\dot{V}_7 + \frac{V_7}{R_4} + j_2 & = & 0,\\
i_E^1: & V_4 - V_3 & = & 0,\\
i_E^2: & V_5 - V_4 & = & 0,\\
i_L: & L\frac{di_L}{dt} - (V_7 - V_6) & = & 0.
\end{array} \tag{74}$$

Its circuit diagram is shown in the upper part of Figure 2. Afterwards we also look at the next modified circuit model, corresponding to the lower circuit diagram in Figure 2.

$$\begin{array}{lrcl}
V_1: & \frac{V_1}{R_1} + C_1(\dot{V}_1 - \dot{V}_2) + j_1 & = & 0,\\
V_2: & C_1(\dot{V}_2 - \dot{V}_1) - i_L & = & 0,\\
V_3: & i_L + \frac{1}{R_2}(V_3 - V_4) & = & 0,\\
V_4: & \frac{1}{R_2}(V_4 - V_3) - i_E^1 & = & 0,\\
V_5: & i_E^1 - i_E^2 & = & 0,\\
V_6: & i_E^2 + \frac{1}{R_3}(V_6 - V_7) & = & 0,\\
V_7: & \frac{1}{R_3}(V_7 - V_6) + C_2\dot{V}_7 + \frac{V_7}{R_4} + j_2 & = & 0,\\
i_E^1: & V_5 - V_4 & = & 0,\\
i_E^2: & V_6 - V_5 & = & 0,\\
i_L: & L\frac{di_L}{dt} - (V_3 - V_2) & = & 0.
\end{array} \tag{75}$$

In both cases we have the following parameter values $R_1 = 10\ \Omega$, $R_2 = R_3 = R_4 = 1\ \Omega$, $L = 0.1$, $\omega_s = 2000\frac{\pi}{8}, \omega_f = 10\omega_s$, $C_1 = C_2 = \frac{1}{\omega_s^2 L}$ and sources $j_1 = \sin(\omega_s t), j_2 = \sin(\omega_f t)$. Clearly it consists of a slow part (the left part) and a fast part, because of the filter behaviour of the circuit and the different frequencies of the current sources j_1 and j_2. Note that both circuit models include additional shorts, which do not affect the solution but make the currents between both parts explicit available. For the first circuit model we considered the partitions: $(\mathbf{x}_L, \mathbf{x}_A) = \big((V_1, V_2, V_3, i_E^1), (V_4, \ldots, V_7, i_E^2, i_L)\big)$ and $(\mathbf{x}_L, \mathbf{x}_A) = \big((V_1, V_2, V_3), (V_4, \ldots, V_7, i_E^1, i_E^2, i_L)\big)$. Both partitions correspond to current interpolation and voltage interpolation, respectively. Figure 3 shows the numerical solution of the active part for this first circuit model on $[0, 10^{-3}]$s. It turns

[3]This is the inhouse analog circuit simulator provided by NXP Semiconductors and also used at Philips.

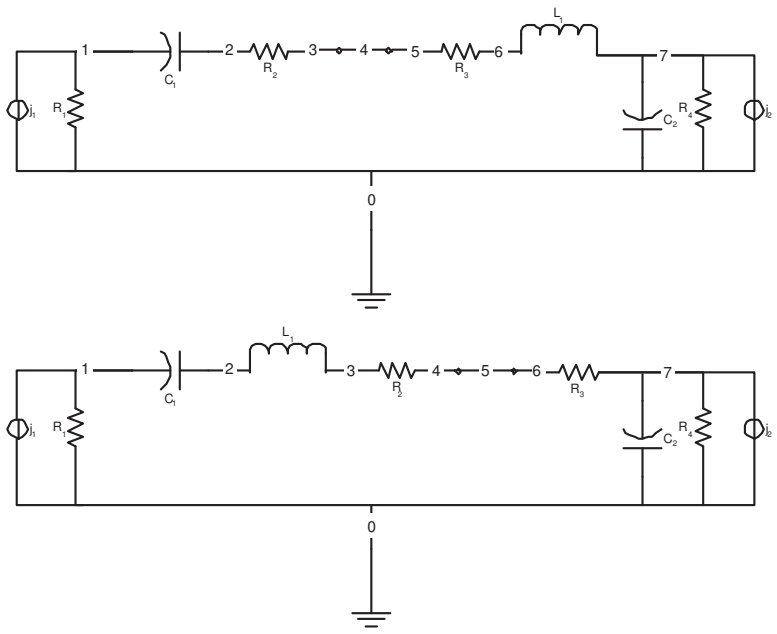


Figure 2: Circuit diagrams of two linear test examples. The nodes 3,4,5 and 4,5,6 are directly connected by shorts, respectively.

out that current interpolation leads to non-smooth behaviour, but the results of voltage interpolation are very good.

For the second circuit model we also considered current interpolation and voltage interpolation: $(\mathbf{x}_L, \mathbf{x}_A) = \big((V_1, \ldots, V_4, i_L, i_E^1), (V_5, V_6, V_7, i_E^2)\big)$ and $(\mathbf{x}_L, \mathbf{x}_A) = \big((V_1, \ldots, V_4, i_L), (V_5, V_6, V_7, i_E^1, i_E^2)\big)$. Now it turns out that current interpolation works well, while with voltage interpolation the active currents are not well computed. Thus current interpolation does not work for the first circuit model, while voltage interpolation does not work for the second model. The reason for this behaviour is that for the first model the active circuit contains the inductor. This ensures that the differential index of the active DAE is larger than one. In this case current interpolation gives problems such that the active voltages are not correctly approximated. However, with voltage interpolation everything is well computed.

For the second model the active circuit does not contain the inductor. This ensures that the index is one. Thus current interpolation gives no problems such that the currents and voltages are correctly approximated. However, with voltage interpolation the active currents are not well computed because they depend now on the derivative of the interpolated slow voltages.

From these experiments it follows that the differential index of the active part of the DAE is very important. If the index is larger than one, the active solution will depend on the higher order derivatives of the inputs. This implies that we get problems with linear interpolation because then the active solution will become discontinuous. For circuit models it is well-known that they are composed of subcircuits in a hierarchical way. If it is known that these subcircuits have index 1 we can exploit this property to get a good partition, i.e. make partitions with interfaces along boundaries of subcircuits [8].

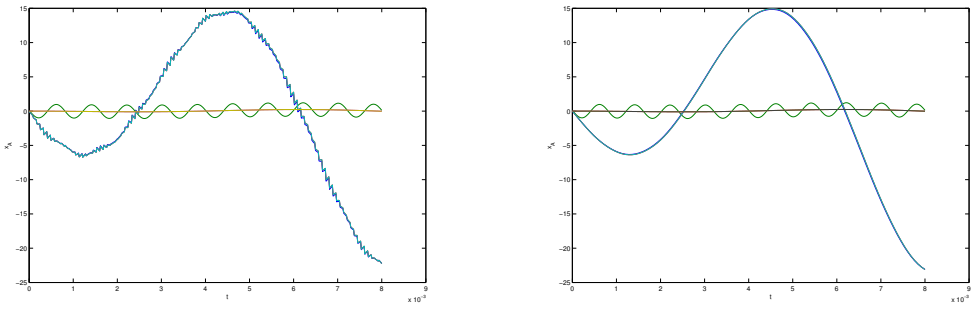


Figure 3: Numerical solution of active part for current (left) and voltage interpolation (right). The low-frequent signal (blue) becomes non-smooth for current interpolation.

4.2 Inverter chain

Consider the circuit model of an inverter chain, which is described in more detail in [1]. It is a chain consisting of 500 inverters. If we excite the first node with a short pulse, a voltage wave is traveling through the chain from left to right. This means that on $[0, 10]$ only the first 8 nodes are activated yet. We applied a BDF Compound-Fast algorithm on $[0, 10]$ with order 1 on the coarse grid and order 2 on the refined grid. During the compound phase we only look at the error of the latent part ($\tau = 0$). During the refinement only the active part, which are the 8 activated nodes, is simulated. The tolerance levels were equal to $\mathrm{TOL}_L = 1$, $\mathrm{TOL}_A = 1$ and $\mathrm{TOL}_C = \mathrm{TOL}_L$. Because of the latency of the slow part, the solution can be determined by just 5 compound steps and 93 refinement steps, while an uniform BDF-method with $k = 1$ would have used about 93 timesteps.

4.3 Matrix circuit

Figure 4b shows another test example, which is a scalable circuit with $M \times N$ subcircuits. The subcircuits are connected with C-elements which can filter the voltages and currents. The circuit is driven by M voltage sources which can have different frequencies. The location of the active part is controlled by the C-elements and the voltage sources. We used the voltage sources $e_i = \frac{5}{2}(1 - \cos(\omega_i t))$, where $\omega_1 = 100 \cdot 10^9$ and, for $i > 1$, $\omega_i = 10^9$. Furthermore, $M = 5$ and $N = 10$ and the subcircuits are inverter models. The C-elements were chosen such that the 3 subcircuits S_{11}, S_{12}, S_{13} are active and nearly decoupled from the other subcircuits. Thus they form an active part because they are activated by a voltage source with higher frequency. We are interested in the results of multirate for several parameter values of the C-elements.

First we did a numerical experiment for $R_1 = R_2 = 10^4, C = L = 10^{-3}$. For these values the C-elements behave like filters for the voltages and currents. We did an Euler Backward multirate simulation on $[0, 10^{-8}]$s with $w = 0.5, \tau = 0$ and tolerance levels $\mathrm{TOL}_C = \mathrm{TOL}_A = 10^{-1}$. We also did a normal Euler Backward simulation with the same tolerance levels. In both cases the timesteps were automatically controlled based on the tolerance levels. It turned out that the singlerate simulation required 2018 timesteps and a computational time of 5491 seconds, while the multirate simulation just required 71 compound steps, 2810 refinement steps and a computational time of 422 seconds. Thus we got a speed-up factor $S \approx 13$.

We also did another experiment where we compared a BDF2 singlerate method with a BDF2 multirate method for $\mathrm{TOL}_C = \mathrm{TOL}_A = 10^{-2}$. We used the balance numbers 0.5 and 10^{-4}. The errors were estimated by comparing the results with the numerical solution of a singlerate BDF2

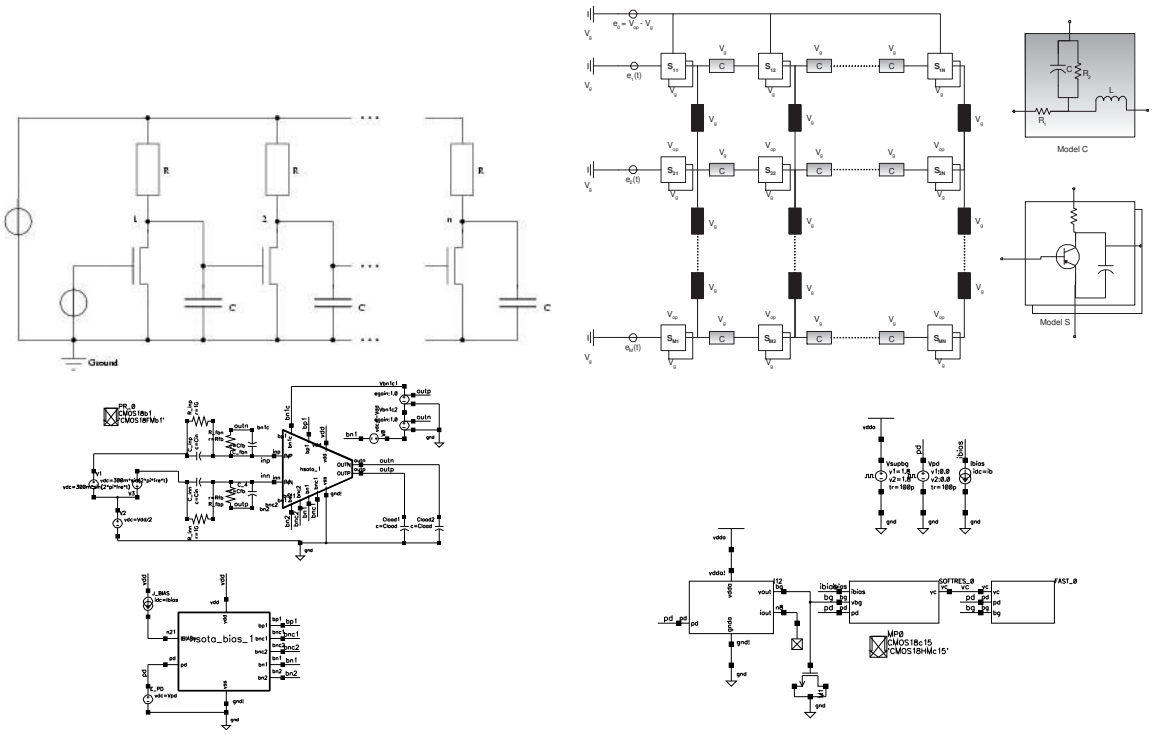


Figure 4: Diagrams of four multirate circuit models.

simulation with higher tolerance levels TOL_C =TOL_A = 10^{-3}. From Table 2 and Figure 5 it follows that the BDF Compound-Fast multirate algorithm is able to produce accurate results in an efficient way. Furthermore, it is very clear that too large values of the balance number decreases the efficiency. From the figure it can be derived that the errors are mainly caused by delay errors in the fast part, because the amplitude is much more accurate.

4.4 Pstar examples

The implementation of the multirate time integration algorithm in Pstar, allows us to obtain results for various circuits. We consider three practical examples, coming from the actual circuit design. The high-speed operational transconductance amplifier (HSOTA) and the charge pump are shown in Figures 4c and 4d.

In the first two circuits there are relatively large bias blocks that have practically constant dy-

Table 2: Statistics of singlerate and multirate BDF2 methods for the Matrix circuit.

method	w	n_C	n_R	comp. time (s)	S	max. error
singlerate		2937		7330		$5.8 \cdot 10^{-2}$
multirate	0.5	111	3765	668	11	$1.8 \cdot 10^{-1}$
multirate	10^{-4}	111	3002	612	12	$1.8 \cdot 10^{-1}$

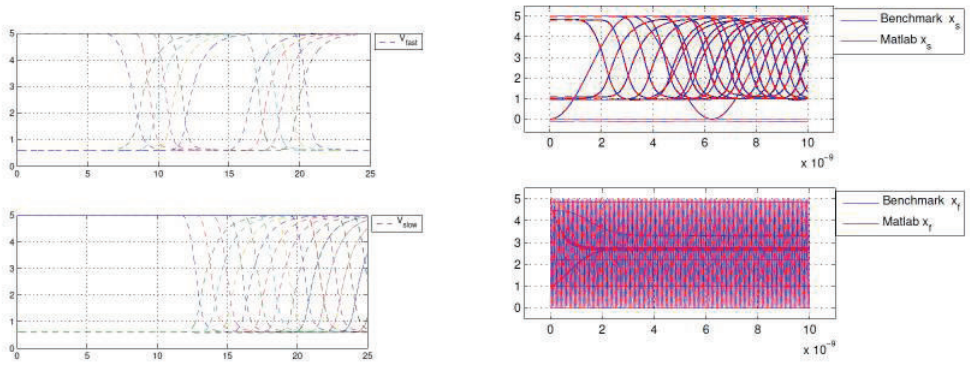


Figure 5: Numerical solution of slow and fast parts for the Inverter chain and the Matrix circuit.

namics, allowing small numbers of compound steps. Numerical results are summarized in Table 3. From the formula for the expected speed-up factor (13) we know that the multirate potential of circuit models depends on the multirate factor q and the work load ratio E. Only if $q >> 1$ and $0 < E << 1$ we can expect a large speed-up factor. The HSOTA model allows a very large multirate factor $q \approx 207$ but has a rather large work load ratio $E \approx 0.5$. Thus the expected speed-up factor equals just $S \approx \frac{1}{E+\frac{1}{q}} \approx 2$. Because the Charge Pump with $q \approx 68$ has a smaller workload ratio, $E \approx 0.12$, its estimated speedup factor is indeed larger.

We also considered a "Sampleadc" circuit, which is a large digital circuit consisting of two subcircuits with different time constants. The work load ratio $E \approx \frac{1}{14}$ is very small. Although the clock frequency ratio of both subcircuits is 40, the real multirate factor is just $q \approx 2$. This low multirate factor is due to the high frequency in the slow circuit during the switching times and the occurring delays in the slow system. Therefore it follows that also the large system with the slow clock is nearly always active. Thus the expected speed-up factor equals $S \approx \frac{1}{E+\frac{1}{q}} \approx 1.8$. Because of this low value this example has not been simulated with the multirate implementation of Pstar.

Only if $q >> 1$ and $0 < E << 1$ we can expect a large speed-up factor. For the HSOTA example the speed-up factor could be increased by using a better partition with smaller E and q. This could be done by hand but it is preferable to use an automatic procedure, which optimizes the estimated speed-up factor.

Table 3: Numerical results. The last example has not been simulated with the Pstar multirate implementation. Notation: d- number of unknowns, N_C- number of compound steps, N_R- number of refinement steps, N_S- number of single-rate steps, d_A- number of active unknowns, S- speed-up factor.

Circuit name	d	N_C	N_R	N_S	q	d_A/d	S
HSOTA	61	68	14092	14068	207	50%	2
Charge pump	249	151	10284	7419	68	12%	4.5
Sampleadc			$\approx 2N_C$		≈ 2	7%	1.8

5 Conclusions

The BDF Compound-Fast multirate method appears to be a very powerful method for DAEs and in particular for circuit simulation. We described in detail how it can be implemented by means of a coarse and a refined Nordsieck representation. It has been shown how the local discretization

error can be estimated and controlled. Here the interpolation errors because of the interface have been included. Also in practice we get fast and accurate results with this type of integration schemes. Automatic partitioning could help to find an optimal partition for which the speed-up factor is maximized.

Acknowledgment

The authors wish to thank the anonymous referees for their careful reading of the manuscript and their suggestions. They also would like to thank the circuit designers Mr. Govert Geelen (HSOTA) and Dr. Marq Kole (Charge Pump) from NXP Semiconductors, for allowing us to use their designs.

References

[1] A. Bartel, M. Günther. *A multirate W-method for electrical networks in state-space formulation*, J. of Comput. and Applied Maths., Vol. 147, pp. 411-425, 2002.

[2] C.W. Gear, D.R. Wells. *Multirate linear multistep methods*, BIT, 24 (1984), 484-502.

[3] A. El Guennouni, A. Verhoeven, E.J.W. ter Maten, T.G.J. Beelen. *Aspects of Multirate Time Integration Methods in Circuit Simulation Problems*, In: A. Di Bucchianico, R.M.M. Mattheij, M.A. Peletier, "Progress in Industrial Mathematics at ECMI 2004", pp 579-584, Springer, 2006.

[4] M. Günther, A. Kværnø, P. Rentrop. *Multirate partitioned Runge-Kutta methods* BIT, Vol.41, pp. 504-514, 2001.

[5] M. Günther, U. Feldmann, J. ter Maten. *Modelling and Discretization of Circuit Problems.* In: P. Ciarlet, W. Schilders, J. ter Maten: *Handbook of Numerical Analysis*, 2005.

[6] E. Hairer, S.P. Nørsett, G. Wanner. *Solving Ordinary Differential Equations I, nonstiff problems.* Springer, 1993.

[7] A. Kværnø. *Stability of multirate Runge-Kutta schemes*, The tenth Int. Conf. on Diff. Equ., Plovdiv, Bulgaria, Aug. 1999.

[8] J. ter Maten, A. Verhoeven, A. El Guennouni, Th. Beelen. *Multirate hierarchical time integration for electronic circuits*, In: PAMM (Proc. GAMM Annual Meeting 2005), Vol. 5, Issue 1, pp. 819-820, 2005.

[9] R.M.M. Mattheij, J. Molenaar. *Ordinary differential equations in theory and practice.* SIAM, 2002.

[10] V. Savcenco, W.H. Hundsdorfer, J.G. Verwer. *A multirate time stepping strategy for stiff ODEs*, BIT, Vol.47, pp. 137-155, 2007.

[11] A. Sjö. *Analysis of computational algorithms for linear multistep methods*, PhD Thesis, Lund University, 1999.

[12] S. Skelboe. *Adaptive partitioning techniques for ordinary differential equations*, BIT, Numerical Mathematics issue in memory of G. Dahlquist, 2006.

[13] G. Söderlind, L. Wang. *Adaptive time-stepping and computational stability*, Journal of Computational and Applied Mathematics, Vol.185, pp 225-243, 2006.

[14] M. Striebel, A. Bartel, M. Günther. *A Multirate ROW-Scheme for Index-1 Network Equations*, to appear in: Special Issue of Appl. Numer. Math, devoted to the 11th NUMDIFF Conference, held in Halle, Sept. 2006.

[15] M. Striebel, M. Günther. *A charge oriented mixed multirate method for a special class of index-1 network equations in chip design*, Applied Numerical Mathematics, Vol.53, pp. 489-507, 2005.

[16] A. Verhoeven. *Redundancy Reduction of IC Models by Multirate Time-Integration and Model Order Reduction* PhD Thesis, Technische Universiteit Eindhoven, 2008.

[17] A. Verhoeven, A. El Guennouni, E.J.W. ter Maten, R.M.M. Mattheij: *A general compound multirate method for circuit simulation problems*, In: A.M. Anile et al: *Scientific Computing in Electrical Engineering*, Series Mathematics in Industry, ECMI, Vol. 9, pp. 143-150, 2006.

[18] A. Verhoeven, B. Tasić, T.G.J. Beelen, E.J.W. ter Maten, R.M.M. Mattheij: *Automatic partitioning for multirate methods*, In: G. Ciuprina et al:*Scientific Computing in Electrical Engineering*, Series Mathematics in Industry, ECMI, Vol. 11, pp. 229-236, 2007.

[19] A. Verhoeven. *Automatic control for adaptive time stepping in electrical circuit simulation*, MSc Thesis[4], Technische Universiteit Eindhoven, Eindhoven, Technical Note TN-2004/00033, Philips Research Laboratories, Eindhoven, 2004.

[20] A. Verhoeven, T.G.J. Beelen, A. El Guennouni, E.J.W. ter Maten, R.M.M. Mattheij, B. Tasić. *Stability analysis of BDF Slowest first multirate methods*, Int. J. of Computer Mathematics, Vol.84, pp. 895-923, 2007.

[21] A. Verhoeven, T.G.J. Beelen, A. El Guennouni, E.J.W. ter Maten, R.M.M. Mattheij, B. Tasić. *Error analysis of BDF Compound-Fast multirate method for differential-algebraic equations*, Ext. abstract Copper Mountain, CASA-Report 06-10[5], 2006.

[4] `http://alexandria.tue.nl/extra2/afstversl/wsk-i/verhoeven2003.pdf`
[5] `ftp://ftp.win.tue.nl/pub/rana/rana06-10.pdf`