Error analysis of BDF compound-fast multirate method for differential-algebraic equations

Verhoeven, A.; Beelen, T.G.J.; El Guennouni, A.; ter Maten, E.J.W.; Mattheij, R.M.M.; Tasic, B.

Published: 01/01/2006

Document Version
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the author’s version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 11. Dec. 2018
Error analysis of BDF compound-fast multirate method for differential-algebraic equations

by

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

A. Verhoeven, T.G.J. Beelen, A. El Guennouni, E.J.W. ter Maten, R.M.M. Mattheij, B. Tasić

1 Technische Universiteit Eindhoven averhoev@tue.nl
2 Philips Research Laboratories
3 Magma Design Automation

1 Introduction

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

\[ \frac{d}{dt} [q(t, x)] + j(t, x) = 0, \]  

where \( x \in \mathbb{R}^d \) represents the state of the circuit. A common analysis is the transient analysis, which computes the solution \( x(t) \) 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 methods, like the BDF-methods. Each iteration, 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 both cases, it would be attractive to integrate these parts with a larger timestep.

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. Let \( B_A \in \mathbb{R}^{d_A \times d} \) and \( B_L \in \mathbb{R}^{d_L \times d} \) with \( d_A + d_L = d \) be the permutation mappings. Then the variables and functions can be split in active (A) and latent (L) parts:

\[ x = B_A^T x_A + B_L^T x_L, \]
\[ q(t, x) = B_A^T q_A(t, B_A x, B_L x) + B_L^T q_L(t, B_A x, B_L x), \]
\[ j(t, x) = B_A^T j_A(t, B_A x, B_L x) + B_L^T j_L(t, B_A x, B_L x). \]

Now equation (1) is equivalent to the following partitioned system:

\[ \frac{d}{dt} [q_A(t, x_A, x_L)] + j_A(t, x_A, x_L) = 0, \]
\[ \frac{d}{dt} [q_L(t, x_A, x_L)] + j_L(t, x_A, x_L) = 0. \]

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

\[ \frac{d}{dt} [q_1(t, x_1, \ldots, x_k)] + j_1(t, x_1, \ldots, x_k) = 0, \]
\[ \vdots \]
\[ \frac{d}{dt} [q_k(t, x_1, \ldots, x_k)] + j_k(t, x_1, \ldots, x_k) = 0. \]
Now we need the permutation mappings $B_i \in \mathbb{R}^{d \times d}$ for $i = 1, \ldots, k$ with the properties:

$$B_i B_j^T = \begin{cases} 
I & \text{if } i = j \\
0 & \text{if } i \neq j
\end{cases}$$

Then the variables and functions can be split in their several activity parts:

$$x = B_1^T x_1 + \ldots + B_k^T x_k,$$

$$q(t, x) = B_1^T q_1(t, B_1 x, \ldots, B_k x) + \ldots + B_k^T q_k(t, B_1 x, \ldots, B_k x),$$

$$j(t, x) = B_1^T j_1(t, B_1 x, \ldots, B_k x) + \ldots + B_k^T j_k(t, B_1 x, \ldots, B_k x).$$

(5)

1.2 Multirate time-integration

In contrast to classical integration methods, multirate time-integration methods integrate both parts of (3) 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 $q_n$. If the two time-grids are synchronized, $T_n = t_{n,0} = t_{n-1,q_n}$ holds for all $n$.

1.3 Overview of this paper

This paper investigates the Compound-Fast multirate version of the multistep BDF method. First, section 2 describes how this method can be efficiently implemented. Because of the interpolation errors, the solution of the active part will be perturbed. In order to keep the accuracy sufficiently high, an error control mechanism is presented in section 3. Section 4 shows some numerical test examples, which show how the presented theory works in practice. Finally, section 5 finishes with the conclusions and lists some problems that currently are being investigated.

2 BDF Compound-Fast multirate algorithm

There are several multirate approaches for partitioned systems [1, 2, 3, 10, 13, 12] but we will consider the "Compound-Fast" version of the BDF methods. This method performs the following four steps:

1. The complete system is integrated at the coarse time-grid.
2. The latent interface variables are interpolated at the refined time-grid.
3. The active part is integrated at the refined time-grid, while the values at the latent interface are given.
4. The active solution at the coarse time-grid is updated.
2.1 The multirate framework

The BDF integration of order \( K \) on the coarse time-grid needs the Lagrange basis polynomial \( L^n(t) \) on \( \{T_{n-K}, \ldots, T_n\} \), with

\[
L^n(T_{n-j}) = \begin{cases} 
1 & \text{if } j = 0 \\
0 & \text{if } j \neq 0 
\end{cases}
\]

Let \( \mathbf{L}^n \in \mathbb{R}^{1 \times (K+1)} \) be the corresponding Nordsieck vector \([4]\) of \( L^n(t) \) which can be expressed as

\[
L^n(t) = \sum_{i=0}^{K} \mathbf{Y}_{i+1}^n \left( \frac{t - T_n}{H_n} \right)^i.
\]

Here \( \mathbf{Y}_{i+1}^n \) is the \( i+1 \)-th element of the Nordsieck vector

\[
\mathbf{L}^n = \left( L^n(T_n), H_n \frac{d}{dt} L^n(T_n), \ldots, \frac{H_n}{k!} \frac{d^k}{dt^k} L^n(T_n) \right).
\]

Clearly, \( \mathbf{L}^n \) is just a way how the polynomial \( L^n(t) \) can be efficiently stored and processed. Furthermore, also the Nordsieck vectors \( \mathbf{Y}^n, \mathbf{X}^n, \mathbf{P}^n, \mathbf{Q}^n \) are needed, which represent the local Predictor and Corrector polynomials for \( x(t) \) and \( q(t, x(t)) \), respectively \([4]\). For instance, the Predictor polynomial \( \mathbf{Y}^n(t) \) for \( x(t) \) on \([T_{n-1}, T_n]\) satisfies

\[
\mathbf{Y}^n(t) = \sum_{i=0}^{k} \mathbf{Y}_{i+1}^n \left( \frac{t - T_n}{H_n} \right)^i.
\]

Now \( \mathbf{Y}_{i+1}^n \) is the \( i+1 \)-th column of the Nordsieck vector\(^4\)

\[
\mathbf{Y}^n = \left( \mathbf{Y}^n(T_n), H_n \frac{d}{dt} \mathbf{Y}^n(T_n), \ldots, \frac{H_n}{k!} \frac{d^k}{dt^k} \mathbf{Y}^n(T_n) \right).
\]

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 \( \{t_{n-1,m-k}, \ldots, t_{n-1,m}\} \), 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 \( \mathbf{Y}^{n-1,m}, \mathbf{X}^{n-1,m}, \mathbf{P}^{n-1,m}, \mathbf{Q}^{n-1,m} \), which represent the local refined Predictor and Corrector polynomials for \( x(t) \) and \( q(t, x(t)) \), respectively.

Now, the refined Predictor polynomial \( \mathbf{Y}^{n-1,m}(t) \) for \( x(t) \) on \([t_{n-1,m-1}, t_{n-1,m}]\) satisfies

\[
\mathbf{Y}^{n-1,m}(t) = \sum_{i=0}^{k} \mathbf{Y}_{i+1}^{n-1,m} \left( \frac{t - t_{n-1,m}}{h_{n-1,m}} \right)^i,
\]

where

\[
\mathbf{Y}^{n-1,m} = \left( \mathbf{Y}^{n-1,m}(t_{n-1,m}), h_{n-1,m} \frac{d}{dt} \mathbf{Y}^{n-1,m}(t_{n-1,m}), \ldots, \frac{h_{n-1,m}}{k!} \frac{d^k}{dt^k} \mathbf{Y}^{n-1,m}(t_{n-1,m}) \right).
\]

Note that for the refined Nordsieck vectors in practice only the active part is stored, because the latent part is not refined. Figure (2.2) 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.

\(^4\) In fact, the Nordsieck vector is a matrix if \( d > 1 \).
2.2 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:

$$\alpha_n q(T_n, x_n) + H_n j(T_n, x_n) + \beta_n = 0,$$

where $\alpha_n$ is an order-dependent parameter and $\beta_n$ is a vector which represents the history of the numerical integration. We can easily express $\alpha_n$ and $\beta_n$ in terms of $L^n$ and $P^n$ by $\alpha_n = L^n$ and $\beta_n = P^n_2 - \alpha_n P^n_0$ [4, 17]. Here $P^n$ is the Nordsieck vector of the Predictor polynomial $P^n(t)$, which is set to be equal to $Q^{n-1}(t)$ if the integration order is constant. For the Nordsieck vectors this means that $P^n = Q^{n-1} T_n$, where $T_n$ is a real matrix $\in \mathbb{R}^{(K+1)x(K+1)}$. For more information about this transformation $T_n$ the reader is referred to [15] and [17]. Usually the initial guess for the solution of (6) is computed by extrapolation, so $x_n = y^n_1$. 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 weighting factor $\omega \ll 1$ in front of the active part. For a compound step, the active part of $x_n$ still must be improved by the refinement phase. After the refinement, we use the updated $x_n$ and $q_n$ to correct the complete predictor polynomials $P^n(t)$ and $Y^n(t)$:

$$Q^n(t) = P^n(t) + (q(T_n, x_n) - P^n(T_n))L^n(t)$$
$$X^n(t) = Y^n(t) + (x_n - Y^n(T_n))L^n(t)$$

For the Nordsieck vectors this means: $Q^n = P^n + (q_n - q_n)E_n$, $X^n = Y^n + (x_n - x_n)E_n$.

2.3 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:

$$\alpha_{n-1,m} q_A(t_{n-1,m}, y_{n-1,m}, \hat{z}_{n-1,m}) + h_{n-1,m} j_A(t_{n-1,m}, y_{n-1,m}, \hat{z}_{n-1,m}) + \beta_{n-1,m} = 0,$$

where $\hat{z}_{n-1,m}$ is the interpolated latent part. We can compute $\hat{z}_{n-1,m}$ from interpolation-based functions which are constructed between the compound step and the refinement phase. Another possibility is to compute it from the Nordsieck vector $X^n$ if it is already corrected for the latent part. Then we can compute directly $\hat{z}_{n-1,m} = B_A X^n E_{n-1,m}$, where $E_{n-1,m}$ is just a row vector. The parameters $\alpha_{n-1,m}, \beta_{n-1,m}$ can be computed in a similar way as for the compound step: $\alpha_{n-1,m} = L_{n-1,m}^2 - \alpha_{n-1,m} \beta_{n-1,m} = P_{n-1,m}^2 - \alpha_{n-1,m} P_{n-1,m}^0$. The predictor Nordsieck vectors $P_{n-1,m}, Y_{n-1,m}$ are similarly computed by $P_{n-1,m} = Q_{n-1,m} T_{n-1,m}, \ Y_{n-1,m} = X_{n-1,m} - T_{n-1,m}$. In general, it is not very difficult to solve (9) because we have an accurate initial guess $\hat{y}_{n-1,m} = B_A \hat{Y}_{n-1,m}$. The solution is used to correct the predictor polynomials. For the Nordsieck vectors we get: $Q_{n-1,m} = P_{n-1,m} + (q_A(t_{n-1,m}, y_{n-1,m}, \hat{z}_{n-1,m}) - q_A)E_{n-1,m}$, $X_{n-1,m} = Y_{n-1,m} + (y_{n-1,m} - \hat{y}_{n-1,m})L_{n-1,m}$. Finally, if $t_{n-1,m} > T_n$ the refined corrector polynomials are evaluated at $T_n$: $B_A x_n = B_A \hat{X}_{n-1,m} E_n, B_A q_n = B_A Q_{n-1,m} E_n$, where $E_n$ is again just a row vector.
2.4 Remarks

The stability of the above method has been treated in [13]. For other variants see also [2, 6, 10, 12]. For specific aspects related to DAEs see: [5, 12].

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. Although the orders \( K_n \) and \( k_n,m \) can be variable, from now on they are assumed to be fixed at \( K_n = K, k_n,m = k \). First 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 Error analysis

Local discretization error

In this section we generalize some techniques in [9, 11] to our multirate case. The local discretization error \( \delta^n \) of the compound phase still has the same behaviour \( \delta^n = O(H^{K+1}) \). This error can be estimated by \( \tilde{\delta}^n \) using the Nordsieck vectors for \( q \):

\[
\tilde{\delta}^n = \frac{H_n}{T_n - T_{n-K-1}} [\hat{Q}_n^n - \hat{P}_1^n].
\]

Now \( f_C^n = ||B_L \tilde{\delta}^n|| + \tau ||B_A \delta^n|| \) is the used weighted error norm, which must satisfy \( f_C^n < TOL_C \). Here \( \tau > 0 \) is a relaxation number which must improve the convergence of the compound step. The optimal value is not known yet.

At the refined time-grid the DAE has been perturbed by the interpolated latent variables. The local discretization error \( \delta^{n,m} \) is defined as the residue after inserting the exact solution in the BDF scheme of the refinement phase. Due to the interpolation at the interface, during the refinement instead of \( \delta^{n,m} \) a perturbed local error \( \bar{\delta}^{n,m} \) is estimated. Each refinement step, \( x_A^{n-1,m} \) is computed from the following scheme:

\[
\alpha_{n-1,m} q_A(t_{n-1,m}, x_A^{n-1,m}, x_L^{n-1,m}) + h_{n-1,m} j_A(t_{n-1,m}, x_A^{n-1,m}, x_L^{n-1,m}) + \beta_{n-1,m} = 0.
\]

Here \( \beta_{n-1,m} \) is a constant which depends on the previous values of \( x_A \) and \( x_L \). Let \( \beta_{n-1,m} \) be the constant for exact values of \( x_L \), then we assume that

\[
\bar{\beta}_{n-1,m} = \beta_{n-1,m} + \frac{\partial \beta_{n-1,m}}{\partial x_L} (x_L^{n-1,m} - x_L(t_{n-1,m})).
\]

Using the notation \( t = t_{n-1,m}, h = h_{n-1,m}, \alpha = \alpha_{n-1,m}, \bar{\beta} = \bar{\beta}_{n-1,m}, \beta = \beta_{n-1,m} \), the error \( \delta^{n,m} \) satisfies

\[
B_A \delta^{n-1,m} = \alpha q_A(t, x_A(t), x_L(t)) + \beta j_A(t, x_A(t), x_L(t))
\]

\[
= \alpha q_A(t, x_A(t), x_L^{n-1,m}) + \beta j_A(t, x_A(t), x_L^{n-1,m}) + \bar{\beta} + hK_{n-1,m}(x_L(t) - x_L^{n-1,m})
\]

\[
= B_A \delta^{n-1,m} + \frac{h}{4} hK_{n-1,m} B_L \rho^{n-1,m}.
\]

Here \( \rho^{n-1,m} \) is the interpolation error at the refined grid and \( K_{n-1,m} \) is the coupling matrix which satisfies

\[
K_{n-1,m} = \frac{\partial q_A}{\partial x_A}(t, x_A(t), x_L(t)) + \frac{\partial q_A}{\partial x_L}(t, x_A(t), x_L(t)) + \frac{\partial j_A}{\partial x_A}(t, x_A(t), x_L(t)) + \frac{\partial j_A}{\partial x_L}(t, x_A(t), x_L(t))
\]

\[
= \frac{d}{dt} \left[ \frac{\partial q_A}{\partial x_A}(t, x_A(t), x_L(t)) + \frac{\partial q_A}{\partial x_L}(t, x_A(t), x_L(t)) \right] + \frac{\partial j_A}{\partial x_A}(t, x_A(t), x_L(t)).
\]
Because of the asymptotic behaviour it turns out that the error estimate $B_A \delta^{n-1,m}$ satisfies

$$B_A \delta^{n-1,m} \doteq B_A \delta^{n-1,m} + \frac{1}{4} h \tilde{K}_{n-1,m} B_L \rho^{n-1,m}. \quad (15)$$

A possible estimate of the coupling matrix is e.g.

$$\tilde{K}_n = \frac{1}{h_n} B_A \left[ \frac{\partial g}{\partial x}(T_n, x^n) - \frac{\partial g}{\partial x}(T_{n-1}, x^{n-1}) \right] B_L^T + B_A \frac{\partial \tilde{g}}{\partial x}(T_n, x^n) B_L^T. \quad (16)$$

The perturbed local discretization error $B_A \delta^{n,m}$ behaves as $O(h^{k+1,n,1,m})$ and can be estimated in a similar way as $\delta^n$.

Let $L$ be the interpolation order, then it can be shown that $\frac{1}{4} ||\tilde{K}_n B_L \rho^{n-1,m}||$ is less than

$$r_L^n = \frac{1}{4} \frac{H_n}{T_n - T_{n-L-1}} ||\tilde{K}_n B_L \left[ \bar{X}_n - \bar{Y}_n \right]||. \quad (17)$$

This error estimate $r_L^n$ has the asymptotic behaviour $r_L^n = O(H_n^{L+1})$. It follows that $||B_A \delta^{n,m}||$ satisfies:

$$||B_A \delta^{n-1,m}|| \leq \frac{r_L^{n-1,m}}{h} + h r_L^n =: r_A^{n-1,m}. \quad (18)$$

If $r_L^n \leq TOL_I = \sigma TOL_A$ and $r_A^{n-1,m} \leq TOL_A = (1-\sigma)TOL_A$ then $r_A^{n-1,m} \leq TOL_A + h TOL_I = TOL_A$. The weighting factor $0 < \sigma < \frac{1}{h_{n-1,m}}$ is chosen such that $(r_C^n / TOL_C) \leq (r_L^n / TOL_I)$.

Multi-rate error estimation

We analysed the behaviour of the local discretization error of the perturbed active part. However we also work with other derived error estimates, like the local error or the combined local error which includes the interpolation errors. The last one is useful for the long-term behaviour of stiff circuits which approximately satisfy $Gx = \sigma$. Let $J_n$ be the Jacobian, then the local error satisfies

$$d_n = J_n \delta_n.$$ 

If $\rho_n$ is the interpolation error, based on $x(t)$, then the combined error estimate satisfies $\sigma_n = \frac{1}{2} ||\rho_n|| + ||d_n||$. Note that the interpolation and the discretization errors must at least have the same order. See also [15].

Because the refinement is just a normal transient simulation of a smaller perturbed system, we use the already available error estimates $r_A^{n-1,m}$. For the compound phase we have error estimates for the discretization error and the interpolation error, $r_C^n$ and $r_L^n$ respectively. The first one can be generalized to the local or the combined error estimates, in an obvious way. For all cases it holds that $r_C^n = O(H_n^{L+1})$ can easily be estimated and controlled.

For the interpolation error $r_L^n$ we have to analyze the equation (13) more profoundly. Because the local discretization error satisfies

$$B_A \delta^{n-1,m} \doteq B_A \delta^{n-1,m} + \frac{1}{4} h \tilde{K}_{n-1,m} B_L \rho^{n-1,m},$$

yielding

$$B_A d^{n-1,m} \doteq [B_A J^{n-1,m} B_A^T]^{-1} (B_A \delta^{n-1,m} + \frac{1}{4} h \tilde{K}_{n-1,m} B_L \rho^{n-1,m}) = B_A d^{n-1,m} + \frac{1}{4} h [B_A J^{n-1,m} B_A^T]^{-1} \tilde{K}_{n-1,m} B_L \rho^{n-1,m}.\quad (19)$$

If we approximate $J^{n-1,m} \doteq J^n$, it follows that

$$B_A d^{n-1,m} \approx B_A \tilde{d}^{n-1,m} + \frac{1}{4} h [B_A J^n B_A^T]^{-1} \tilde{K}_n B_L \rho^n.\quad (20)$$

Thus also the local error can be estimated in a similar way:
\[ \| B_A \hat{d}^{n-1,m} \| \leq \| B_A d^{n-1,m} \| + \frac{1}{4} h \| B_A J^n B_A^T \|^{-1} K_n B_L \beta^n. \]

Because the interpolation error \( B_A \hat{\beta}^{n-1,m} \) does not depend on the perturbation, we get
\[ \frac{1}{4} \| B_A \hat{\beta}^{n-1,m} \| + \| B_A \hat{d}^{n-1,m} \| \leq \frac{1}{4} \| B_A \beta^{n-1,m} \| + \| B_A d^{n-1,m} \| + \frac{1}{4} h \| B_A J^n B_A^T \|^{-1} K_n B_L \beta^n. \]

Table (3.1) shows the error estimates for the investigated cases.

<table>
<thead>
<tr>
<th>f</th>
<th>local discr. error</th>
<th>local error</th>
<th>combined error</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tilde{r}_n )</td>
<td>( \frac{1}{2} | B_A \delta^n | + \tau | B_A \hat{\beta}^{n} | )</td>
<td>( | B_A \hat{d}^{n} | + \tau | B_A \hat{d}^{n} | )</td>
<td>( \frac{1}{4} | B_A \hat{\beta}^{n} | + | B_A \hat{d}^{n} | + \tau | B_A \hat{\beta}^{n} | + | B_A \hat{d}^{n} | )</td>
</tr>
<tr>
<td>( \hat{r}_A )</td>
<td>( \frac{1}{2} | B_A \delta_A | + \tau | B_A \hat{\beta}_A | )</td>
<td>( | B_A \hat{d}_A | + \tau | B_A \hat{d}_A | )</td>
<td>( \frac{1}{4} | B_A \hat{\beta}_A | + | B_A \hat{d}_A | + \tau | B_A \hat{\beta}_A | + | B_A \hat{d}_A | )</td>
</tr>
</tbody>
</table>

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

### 3.2 Error control

#### Error model

Adaptive stepsize control of \( H_n \) and \( h_{n,m} \) can be used to keep \( \hat{r}_n = O(H_{n+1}^p) \) and \( \hat{r}_A^{n-1,m} = O(h_{n-1,m}^p) \) close to \( \theta TOL_j \) and \( \theta TOL_A \) respectively, where \( 0 < \theta < 1 \) is a safety factor. In [13, 9, 14] one can read how control theory can be applied to properly designed stepsize controllers.

From several experiments it appears that the Compound-Fast algorithm works well for circuits with latency or multirate behaviour.

For the "Compound Step" method we have applied the following error model, where \( \hat{r}_* = ||\delta||, \hat{\phi}_* = ||\hat{\phi}\|\).  

\[
\begin{align*}
\hat{r}^C_n &= \phi^C_n H_{n+1}^p, \\
\hat{r}^A_{n,m} &= \hat{r}_A + h_{n,m} \hat{r}_n, \\
\hat{r}^A_n &= \hat{r}_A + h_{n,m} H_{n+1}^p, \\
\hat{r}_n &= \phi^C_n H_{n+1}^p.
\end{align*}
\]

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

\[
\begin{align*}
\hat{r}_n^C < TOL_C, \quad & (23) \\
\hat{r}_A^{n,m} < TOL_A. \quad & (24)
\end{align*}
\]

Because \( \hat{r}_n^C \) can be measured and only depends on \( H_n \), this value can be 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 \( h_{n,m} \hat{r}_n \) 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 numerical point of view. Therefore, we restrict ourselves to EPS control.

Assume that

\[
\begin{align*}
\hat{r}_n^{n,m} &= TOL_A = (1 - w)TOL_A \\
\hat{r}_n &= TOL_A = \frac{1}{h} TOL_A
\end{align*}
\]

where \( w \in (0, 1) \) is a balance number and \( h \) is the maximum refinement step. Then, we have

\[
\hat{r}_n^{n,m} < \hat{r}_A^{n,m} + h_{n,m} \hat{r}_n < TOL_A.
\]

Now \( \hat{r}_A^{n,m} \) can be controlled by \( h_{n,m} \) and \( \hat{r}_n \) can be controlled by \( H_n \). It is also possible to apply other techniques than this one, like linearization or alternative models.
Adaptive multirate stepsize control

The stepsize controllers can be based on the derived error model. If we want to track $\tau_C^n$ close to $\text{TOL}_C$ and $\tau_A^{n,m}$ close to $\text{TOL}_A$, the next elementary stepsize controllers could be used:

$$H_{n+1} = \left(\frac{\text{TOL}_C}{\tau_C^n}\right) H_n,$$

$$h_{n,m+1} = \left(\frac{\text{TOL}_A}{\tau_A^{n,m}}\right) h_{n,m}.$$  \hspace{1cm} (27)

Here, $\theta$ is a safety factor which satisfies $0 < \theta < 1$. Of course, also other more advanced stepsize controllers can be used, e.g. digital linear step size controllers [11]. The resulting error estimates have to satisfy the following error constraints:

$$\tau_C^n < \text{TOL}_C,$$

$$\tau_f^n < \text{TOL}_I,$$

$$\tau_A^{n,m} < \text{TOL}_A.$$  \hspace{1cm} (29)

$$h_{n,m+1} = \min\left\{\left(\frac{\text{TOL}_C}{\tau_C^n}\right) H_n, \left(\frac{\text{TOL}_I}{\tau_f^n}\right) H_n\right\}.$$  \hspace{1cm} (30)

Because the stepsize controller (27) does not take care with the error estimate $\tau_f^n$, it can happen that the second constraint (30) is not satisfied. An option is to use the following mixed controller which controls $\min\{\tau_C^n, \tau_f^n\}$, e.g.

$$H_{n+1} = \min\left\{\left(\frac{\text{TOL}_C}{\tau_C^n}\right) H_n, \left(\frac{\text{TOL}_I}{\tau_f^n}\right) H_n\right\}.$$  \hspace{1cm} (32)

Estimation of speed-up factor

Let $W_L$ and $W_A$ be the computational work per step for the compound phase and the refinement phase, satisfying $W_A/W_L = E$. Typically it holds that

$$E = O\left(\left(\frac{n_A}{n_L}\right)^\alpha\right),$$

where $\alpha \in (1, 3)$ depends on the type of the problem. Typically $\alpha \approx 1$ for small and medium-sized systems with expensive function evaluations, while $\alpha \rightarrow 3$ for large-scale systems because of the dominant linear algebra. Assume that $H_n = h$ and $h_{n,m} = \frac{h}{M}$. Then the multirate-potential number $M$ is defined as $M = E h > 0$. This number $M$ affects the speed-up factor and also the optimal weighting number $w_n$. Because for electrical circuits, the latent sub-circuits have much more variables than the active circuits, the number $E$ will be quite small.

A multirate method for (1) on $[0,T]$ will need the following computational work load

$$W_m = W_A \frac{T}{h} + W_L \frac{T}{H} = W_L T \left(\frac{1}{h} + \frac{1}{H}\right),$$

while a singlerate method with step $h_s$ would need $W_s = W_L \frac{T}{h_s}$. Thus we have the following speed-up factor

$$S = \frac{W_s}{W_m} = \frac{1}{E \frac{1}{h} + \frac{1}{H}} = \frac{1}{1 + E h_s} \frac{H}{1 + M h_s}.$$  \hspace{1cm} (34)

If $S > 1$ it is attractive to use for instance the multirate version of the BDF method when compared to a normal BDF time-integration. Note that $S = \frac{h}{1+M}$ for $h_s = h$. Clearly, if $h_s << H$ and the multirate-potential number $M$ is small, we get a large speed-up factor.
4 Numerical experiments

In order to test the previous theoretical results we applied it to a large number of test examples. This section shows the results for two circuit models.

4.1 Inverter chain

First, we look at the circuit model of the following 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 travelling 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 $TOLL = 1$, $TOLA = 1$ and $TOLC = TOLL$. Because of the latency of the slow part, the solution can be determined by just 5 compound steps and 93 refinement steps.

4.2 Multirate circuit

The following figure 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 $e_i = \frac{3}{2}(1 - \cos(\omega_i t))$, 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 $S_{11, 12, 13}$ are active and nearly uncoupled from the other subcircuits. Thus they form an active part because they are activated by a voltage source with higher frequency. We did an Euler Backward multirate simulation on $[0, 10^{-8}]$ with $\sigma = 0.5, \tau = 0$. We are interested in the results of multirate for several parameter values of the C-elements. From the following table and figure it follows that the BDF Compound-Fast multirate algorithm is able to produce accurate results in an efficient way for the uncoupled case but also for the weakly coupled case.

<table>
<thead>
<tr>
<th>$R_1$</th>
<th>$R_2$</th>
<th>$C$</th>
<th>$L$</th>
<th>$n_C$</th>
<th>$n_R$</th>
<th>$n_S$</th>
<th>mult. time (s)</th>
<th>sing. time (s)</th>
<th>$S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^4$</td>
<td>$10^4$</td>
<td>$10^{-3}$</td>
<td>$10^{-1}$</td>
<td>71</td>
<td>2810</td>
<td>2018</td>
<td>422</td>
<td>5491</td>
<td>13</td>
</tr>
<tr>
<td>$10^3$</td>
<td>$10^3$</td>
<td>$10^{-3}$</td>
<td>$10^{-1}$</td>
<td>94</td>
<td>2710</td>
<td>495</td>
<td>495</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Statistics of singlerate and multirate methods for Multirate circuit.
5 Conclusions

The BDF Compound-Fast multirate method appears to be a very powerful method for DAEs and in particular for circuit simulation. It is rather simple to control its local error including the interpolation errors for coupled systems. Also in practice it seems to be possible to get fast and accurate results with this type of integration schemes.

References