Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational and Applied Mathematics 185 (2006) 441–459
JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS
www.elsevier.com/locate/cam

Time domain analog circuit simulation☆

J.G. Fijnvandraat[a], S.H.M.J. Houben[b], E.J.W. ter Maten[c,*], J.M.F. Peters[a]

[a]*Philips Research Laboratories, Eindhoven, The Netherlands*
[b]*Magma Design Automation, Eindhoven, The Netherlands*
[c]*Philips Research Laboratories, Eindhoven and Eindhoven University of Technology, ED&T/AS, Prof. Holstlaan 4, 5656 AA, Eindhoven, The Netherlands*

Received 28 March 2003

Abstract

Recent developments of new methods for simulating electric circuits are described. Emphasis is put on methods that fit existing datastructures for backward differentiation formulae methods. These methods can be modified to apply to hierarchically organized datastructures, which allows for efficient simulation of large designs of circuits in the electronics industry.

MSC: 65L06; 65L80

Keywords: Transient analysis; Modified nodal analysis; Differential-algebraic equations; Hierarchical datastructures; Recursion; BDF; Radau; Rosenbrock–Wanner; MEBDF; Parallelism; Model reduction; Table modeling; Multi-rate; Multi-tone; Mixed-signal; Mixed-level

1. Introduction

Circuit simulation applies to electronic networks, in which electronic elements like resistors, capacitors, inductors, transistors and sources are connected into one big system. Similar networks one also encounters in other network systems through which a flow can be observed. One can think of a network of rivers and canals, or a network of a sewerage system, or of a blood circulation system. Even traffic flow can be

☆ Presented at IWTAM-2002, Bari, Italy, 18–20 December 2002.

* Corresponding author.
E-mail address: jan.ter.maten@philips.com (E.J.W. ter Maten).

0377-0427/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2005.03.021

studied by a network system. In fact a circuit simulator is well suited to analyse these other systems as well, because the properties of electronic elements can very well be used in defining properties of special sections in the other networks. A circuit simulator is also very often applied to so-called reduced order models of more refined models arising in other application areas.

In analog circuit simulation a lot of different analyses can be applied such as DC or steady-state analysis, transient analysis, AC-analysis (linear frequency domain analysis, after linearization around a DC-solution), noise analysis, harmonic balance (nonlinear frequency domain analysis for harmonic distortion analysis), pole-zero analysis (which requires a generalized eigenvalue analysis), mixed-signal analysis (for circuits with analog as well as digital waveforms), and dedicated analyses for problems with waveforms in the radio frequency (RF) range (1–10 GHz): periodic steady-state (PSS) analysis, periodic AC, periodic noise.

Here we will focus on transient analysis, which is the analysis most heavily used. For this type of analysis most circuit simulators apply the well-known backward differentiation formulae (BDF) integration method to the system of differential-algebraic equations (DAE) that arises after applying the so-called modified nodal analysis to the network description of the circuit. In the Sections 2 and 3 we will briefly describe the system of equations. Next, in Section 4, we recall some results that allow for determining the DAE-index by checking the topology of the network. In Section 5 we will pay attention to the hierarchical framework and its options to speed up analysis. The form of the DAEs does not elegantly meet the specifications of several standard implementations of algorithms for time integration (even for BDF). Also these methods do not comply with hierarchical datastructures where recursion could be exploited. Hence, applying these methods generally requires serious modification. The drawback is that the algorithms are not easily changed later on in order to study alternatives. Clearly, there is a big need to speed up transient simulations while maintaining or increasing the robustness of the methods. In the Sections 6 and 7 some interesting developments can be recognized that will be addressed briefly, amongst them methods that have better stability properties than BDF, while also offering higher accuracy (Radau, MEBDF, Rosenbrock–Wanner). Section 8 gives requirements for, as well as first developments of, new methods that could be applied to advanced problems involving multi-rate, multi-tone, and/or mixed-signal and mixed-level aspects.

2. Kirchhoff Laws

The equations that govern the behaviour of an electronic circuit are formulated in terms of physical quantities along the branches and at the nodes of the network. A simple example is shown in Fig. 1, involving a current source j_1, an inductor l_1, two capacitors, c_1 and c_2, and a resistor r_1. The basic laws deal with the currents i_k through the branches between the nodes $0, \ldots, 3$, and the voltage differences over the branches:

- Kirchhoff's Current Law (KCL): $\sum_{k\in\text{cutset}} i_k = 0$.
- Kirchhoff's Voltage Law (KVL): $\sum_{k\in\text{loop}} v_k = 0$.

Here a cutset is a set of elementary branches B_c that bridges two disjoint sub-networks B_L, B_R, such that $B_L \cup B_\text{c} \cup B_R$ is the whole network. The KCL allows to set up (current) equations with voltages as unknowns, the KVL can be used the other way around. The laws can be formulated very compactly by

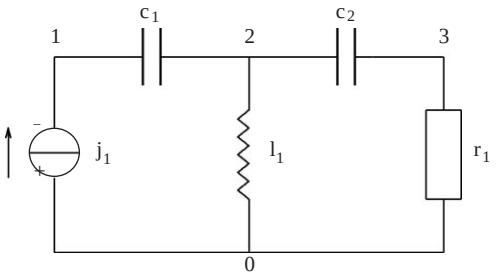


Fig. 1. Simple example circuit.

introducing an incidence matrix, which is completely determined by the topology of the circuit. Assuming n nodes and b branches, the incidence matrix $A \in \mathbb{R}^{n\times b}$ is defined by

$$a_{ij} = \begin{cases} 1 & \text{if branch } j \text{ arrives at node } i, \\ -1 & \text{if branch } j \text{ departs from node } i, \\ 0 & \text{if branch } j \text{ is not incident at node } i. \end{cases}$$

With this a direction of the branches is defined, but the actual choice is irrelevant. For the above example we may choose the form

$$\begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccccc} j_1 & c_1 & c_2 & l_1 & r_1 \end{array} \\ \begin{pmatrix} -1 & 0 & 0 & -1 & -1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \end{pmatrix} \end{array}.$$

Clearly, because all elements are two-nodal, $\sum_i a_{ij} = 0$ and $\mathrm{Rank}(A) \leqslant n-1$ (but for a proper circuit $\mathrm{Rank}(A) = n-1$).

If we introduce the vector $\mathbf{i}_b$ of branch currents, and the vector $\mathbf{v}_b$ of branch voltages, as well as the vector $\mathbf{v}_n$ of nodal voltages, the laws can be summarized as

- KCL: $A\mathbf{i}_b = 0$.
- KVL: $A^{\mathrm{T}}\mathbf{v}_n = \mathbf{v}_b$.

3. Modified nodal analysis

In nodal analysis one assumes current-defined, voltage-controlled components with branch constitutive relations of the form $i_b = \tilde{j}(t, v_b)$ (resistors, current sources) or $i_b = (\mathrm{d}/\mathrm{d}t)\tilde{q}(t, v_b)$ (capacitors) for the particular elements involved, which in vector form sum up to a linear combination and for which after applying the KCL and KVL equivalent properties of A yield

$$\mathbf{i}_b = \frac{\mathrm{d}}{\mathrm{d}t}\tilde{\mathbf{q}}(t, \mathbf{v}_b) + \tilde{\mathbf{j}}(t, \mathbf{v}_b),$$

$$\Rightarrow A\mathbf{i}_b = \frac{\mathrm{d}}{\mathrm{d}t} A\tilde{\mathbf{q}}(t, \mathbf{v}_b) + A\tilde{\mathbf{j}}(t, \mathbf{v}_b),$$

$$\Rightarrow \mathbf{0} = \frac{\mathrm{d}}{\mathrm{d}t} A\tilde{\mathbf{q}}(t, A^{\mathrm{T}}\mathbf{v}_n) + A\tilde{\mathbf{j}}(t, A^{\mathrm{T}}\mathbf{v}_n).$$

If we assume that $\mathrm{Rank}(A) = n - 1$, after grounding the kth nodal voltage to v_k, we can leave out the kth row of A (which we denote by $\hat{A}$) and we can formulate a system of equations in $\hat{\mathbf{v}}_n = \mathbf{v}_n - v_k\mathbf{e}_k$ (in which we also leave out the kth coordinate):

$$\mathbf{0} = \frac{\mathrm{d}}{\mathrm{d}t} \hat{A}\tilde{\mathbf{q}}(t, \hat{A}^{\mathrm{T}}\hat{\mathbf{v}}_n + v_k A^{\mathrm{T}}\mathbf{e}_k) + \hat{A}\tilde{\mathbf{j}}(t, \hat{A}^{\mathrm{T}}\hat{\mathbf{v}}_n + v_k A^{\mathrm{T}}\mathbf{e}_k),$$

where $\mathbf{e}_k \in \mathbb{R}^n$ is the kth unit vector. With $\mathbf{x} := \hat{\mathbf{v}}_n$, $\mathbf{q}(t, \mathbf{x}) := \hat{A}\tilde{\mathbf{q}}(t, \hat{A}^{\mathrm{T}}\hat{\mathbf{v}}_n + v_k A^{\mathrm{T}}\mathbf{e}_k)$ and $\mathbf{j}(t, \mathbf{x}) := \hat{A}\tilde{\mathbf{j}}(t, \hat{A}^{\mathrm{T}}\hat{\mathbf{v}}_n + v_k A^{\mathrm{T}}\mathbf{e}_k)$ we arrive at

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{q}(t, \mathbf{x}) + \mathbf{j}(t, \mathbf{x}) = 0,$$

which in general is a DAE, because some clusters of resistive branches may occur, which will be purely defined by algebraic equations.

In practice, one also considers voltage-defined, current-controlled components with branch equations of the form

$$v_j = \frac{\mathrm{d}}{\mathrm{d}t}\bar{q}(t, i_j) + \bar{j}(t, i_j),$$

which can be used to describe voltage sources and inductors. To cover also these equations we introduce $\mathbf{v}_b = (\mathbf{v}_{b_1}^{\mathrm{T}}, \mathbf{v}_{b_2}^{\mathrm{T}})^{\mathrm{T}}$ and $\mathbf{i}_b = (\mathbf{i}_{b_1}^{\mathrm{T}}, \mathbf{i}_{b_2}^{\mathrm{T}})^{\mathrm{T}}$, respectively, both of length $b = b_1 + b_2$, in which b_1 corresponds to the already considered current-defined, voltage-controlled part and b_2 to the voltage-defined, current-controlled part. We will consider the $\mathbf{i}_{b_2}$ as additional unknowns and start from

$$\mathbf{i}_{b_1} = \frac{\mathrm{d}}{\mathrm{d}t}\tilde{\mathbf{q}}(t, \mathbf{v}_{b_1}, \mathbf{i}_{b_2}) + \tilde{\mathbf{j}}(t, \mathbf{v}_{b_1}, \mathbf{i}_{b_2}),$$

$$\mathbf{v}_{b_2} = \frac{\mathrm{d}}{\mathrm{d}t}\bar{\mathbf{q}}(t, \mathbf{v}_{b_1}, \mathbf{i}_{b_2}) + \bar{\mathbf{j}}(t, \mathbf{v}_{b_1}, \mathbf{i}_{b_2}).$$

[For simplicity we have omitted the $\mathbf{v}_{b_2}$ as variables at the right-hand sides.] After applying KCL and KVL

$$A_1\mathbf{i}_{b_1} + A_2\mathbf{i}_{b_2} = 0, \quad \text{(KCL)},$$

$$A_1^{\mathrm{T}}\mathbf{v}_n = \mathbf{v}_{b_1}, \quad A_2^{\mathrm{T}}\mathbf{v}_n = \mathbf{v}_{b_2}, \quad \text{(KVL)}$$

one has

$$-A_2\mathbf{i}_{b_2} = \frac{\mathrm{d}}{\mathrm{d}t} A_1\tilde{\mathbf{q}}(t, A_1^{\mathrm{T}}\mathbf{v}_n, \mathbf{i}_{b_2}) + A_1\tilde{\mathbf{j}}(t, A_1^{\mathrm{T}}\mathbf{v}_n, \mathbf{i}_{b_2}),$$

$$A_2^{\mathrm{T}}\mathbf{v}_n = \frac{\mathrm{d}}{\mathrm{d}t}\bar{\mathbf{q}}(t, A_1^{\mathrm{T}}\mathbf{v}_n, \mathbf{i}_{b_2}) + \bar{\mathbf{j}}(t, A_1^{\mathrm{T}}\mathbf{v}_n, \mathbf{i}_{b_2})$$

and thus, after grounding,

$$-\hat{A}_2\mathbf{i}_{b_2} = \frac{\mathrm{d}}{\mathrm{d}t}\hat{A}_1\tilde{\mathbf{q}}(t,\hat{\mathbf{y}},\mathbf{i}_{b_2}) + \hat{A}_1\tilde{\mathbf{j}}(t,\hat{\mathbf{y}},\mathbf{i}_{b_2}),$$

$$\hat{A}_2^{\mathrm{T}}\hat{\mathbf{v}}_n + v_k A_2^{\mathrm{T}}\mathbf{e}_k = \frac{\mathrm{d}}{\mathrm{d}t}\bar{\mathbf{q}}(t,\hat{\mathbf{y}},\mathbf{i}_{b_2}) + \bar{\mathbf{j}}(t,\hat{\mathbf{y}},\mathbf{i}_{b_2}), \quad \text{where}$$

$$\hat{\mathbf{y}} = \hat{A}_1^{\mathrm{T}}\hat{\mathbf{v}}_n + v_k A_1^{\mathrm{T}}\mathbf{e}_k,$$

which, after re-arranging everything to the right-hand side, again has the form (DAE)

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{q}(t,\mathbf{x}) + \mathbf{j}(t,\mathbf{x}) = 0. \tag{1}$$

Clearly, this can be written in a further standard form

$$\mathbf{f}(t,\mathbf{x},\dot{\mathbf{x}}) = \frac{\partial}{\partial t}\mathbf{q}(t,\mathbf{x}) + \frac{\partial}{\partial x}(\mathbf{q}(t,\mathbf{x}))\dot{\mathbf{x}} + \mathbf{j}(t,\mathbf{x}) = 0,$$

which allows for applying several standard time integrators, but at the price of requiring second derivatives when solving nonlinear systems. Of course, standard methods can also be easily applied to (1) after introducing $\mathbf{q}$ as additional unknown, but in most cases this doubles the number of unknowns. In practice, one formulates time integrators directly for DAEs of the form (1) [20]. Note that systems of this form also arise in studying dynamics in multibody systems. In the electronic circuit application one has to keep in mind that the unknown $\mathbf{x}$ contains all the nodal voltages $\hat{\mathbf{v}}_n$ and (only) the currents of the voltage-defined, current-controlled elements, thus two different physical quantities, of which the last one, one may argue, was introduced by some artificial procedure that facilitated the modeling of the network. Hence, in practice, in convergence and accuracy criteria, one mostly restricts oneself to the $\hat{\mathbf{v}}_n$.

4. Topological analysis

Well-posedness of the circuit equations cannot be established by considering each circuit branch in isolation. Rather, the topology, i.e., the way in which the branches are interconnected, plays an essential role. For an in-depth analysis, see [11]. As an example, consider the circuit shown at the left in Fig. 2. This circuit contains two ideal voltage sources *a* and *b*. The potential in the nodes 0 and 1 will be denoted

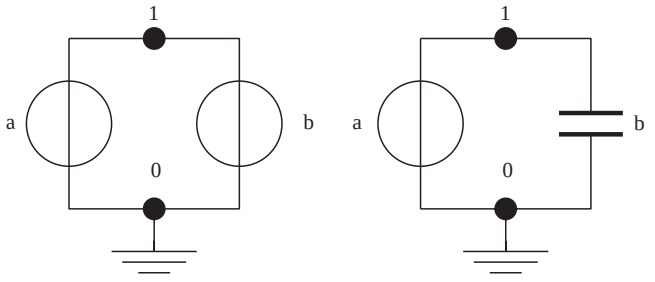


Fig. 2. Left: A loop of voltage sources. Right: A capacitor/voltage source loop.

as $v_{n,0}$ and $v_{n,1}$, respectively. Because of KVL, the voltage difference v_a over a is equal to $v_{n,1} - v_{n,0}$; similarly, the voltage difference v_b over b is equal to $v_{n,1} - v_{n,0}$. Obviously, if $v_a \neq v_b$ then there is no solution. If $v_a = v_b$, there are infinitely many solutions, since any amount of current could flow through the loop. Thus the circuit equations corresponding to this circuit are not well-posed. In general, necessary conditions for the circuit equations to be well-posed are that no loops of voltage sources occur, nor cutsets that consist of branches of current sources. A circuit simulator usually detects such topologies and gives an error message if it encounters them. This is a form of *topological analysis*, i.e., analysis on the solution behaviour of the circuit which involves the interconnections between the circuit branches.

A more subtle problem arises in the circuit shown at the right in Fig. 2. Assume that voltage source a is time-dependent, i.e., $v_a = v_a(t)$. The capacitor b has the branch equation $i_b = Cv_a'(t)$. Observe that the current depends on the first derivative of the source term. This may lead to numerical problems; we typically have that the local discretization error made for i_b is one order lower than the order of the time integration method [17]. This phenomenon is called *order reduction*. Another problem arises when $v_a(t)$ is not differentiable. $v_a(t)$ might be a block pulse coming from a digital part of the circuit, or it might be a white noise source. In these cases, the solution only exists in a generalized sense. To classify these cases the concept of DAE-index has been introduced. A DAE with DAE-index 0 is just an ordinary differential equation. When the DAE-index equals m with $m \geqslant 1$, derivatives up to order $m - 1$ of the input sources are needed to uniquely define the solution of the system of DAE. For the current example, the resulting system of DAEs has DAE-index 2 (for definitions and generalizations we refer to [6,11,17,25]).

It is clear that one wants to avoid circuits that define DAEs with DAE-index greater than 1. Here topological analysis can be of help. Indeed, apart from detecting pathological configurations, topological analysis can be helpful in the simulation of correct circuits. A typical result, derived and discussed in [11,20,32], is as follows.

Theorem 1. *Assume*

- *The circuit graph and the element characterizations are time independent.*
- *There are no cutsets of current sources.*
- *There are no loops of voltage sources.*
- *All L-I cutsets do not contain current-controlled sources.*
- *All V-C loops do not contain voltage-controlled sources.*
- *All components are passive.*

Then the system (1) *has DAE-index* $m \leqslant 1$.

5. Hierarchical simulation

In practice, electronic circuits (circuit 'model') are defined in a hierarchical way using sub-circuits (sub-model). This allows for re-use of very large, carefully designed sub-circuits from a model-library (for instance describing interconnect). But also the compact models for transistors easily fit in a hierarchical description. Thinking in sub-circuits allows to define, design and test local functionality and to use the result as a parameterized building block that is connected to a parent circuit by its terminal nodes (terminals). To capture full functionality the sub-circuit may have internal nodes. Of course, each sub-circuit may give rise to more sub-sub-circuits, etc. An example of a parent-circuit with two sub-circuits is shown in Fig. 3. Several circuit simulators generate a flat description to apply the analysis to. While integrating,

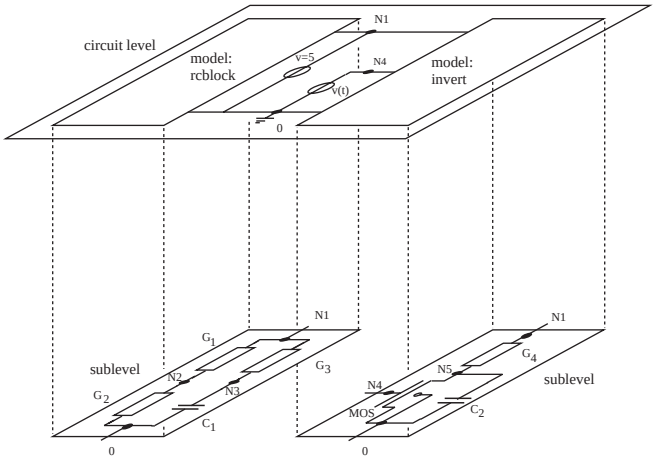


Fig. 3. Example of a parent circuit with two, parallel, sub-circuits.

special algebraic techniques are needed to solve the large intermediate linear systems efficiently. However, it is clear from the figure above, that the original description already offers inherent parallelism and that at each level the local systems may be quite moderate. There even may be repetition of sub-circuits (which occurs frequently in memory circuits). This is fully exploited by hierarchical simulators like HSIM [22] and Pstar [29].

To describe the hierarchical problem in more detail we split the unknowns $\mathbf{x}^{(j)}$ in sub-circuit j in *terminal* ($\mathbf{y}^{(j)}$) and *internal* ($\mathbf{z}^{(j)}$) unknowns:

$$\mathbf{x}^{(j)} = \begin{pmatrix} \mathbf{y}^{(j)} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \mathbf{z}^{(j)} \end{pmatrix}.$$

It is assumed that the equations for the internal unknowns, $(\mathrm{d}/\mathrm{d}t)\mathbf{f}_1^{(j)}(t, \mathbf{y}^{(j)}, \mathbf{z}^{(j)}) + \mathbf{f}_2^{(j)}(t, \mathbf{y}^{(j)}, \mathbf{z}^{(j)}) = 0$, say, uniquely determine $\mathbf{z}^{(j)}$ for given $\mathbf{y}^{(j)}$ (which is just elimination of the internal unknowns from the system). A simple example in which this assumption is violated is given by a sub-circuit consisting of one voltage source E, of which the internal unknown i_E (current through E) is not determined by the terminal voltages only. In this case the introduction of a terminal current as additional unknown at the parent circuit level and lifting the equation for i_E to the parent level solves the modeling problem (after which the i_E is determined in an implicit way by the whole system). This can be generalized to sequences of currents of voltage-defined elements.

For communication purposes between a parent and a sub-circuit we introduce an additional incidence matrix

$$B_j(i,k) = \begin{cases} 1 & \text{if unknown } k^{\text{sub}} \equiv i^{\text{parent}}, \\ 0 & \text{else.} \end{cases}$$

Then: $\mathbf{x}^{\text{parent}} = B_j \mathbf{y}^{(j)}$ and $\mathbf{y}^{(j)} = B_j^{\mathrm{T}} \mathbf{x}^{\text{parent}}$ express the sub-circuit terminals $\mathbf{y}^{(j)}$ in terms of $\mathbf{x}^{\text{parent}}$ and the other way around. Now, assuming N parallel sub-circuits, the following system of results:

$$\frac{\mathrm{d}}{\mathrm{d}t} \sum_{j=1}^{N} B_j \mathbf{q}^{(j)}(t, \mathbf{w}^{(j)}) + \sum_{j=1}^{N} B_j \mathbf{j}^{(j)}(t, \mathbf{w}^{(j)}) + \frac{\mathrm{d}}{\mathrm{d}t} \mathbf{q}^{\text{parent}}(t, \mathbf{x}) + \mathbf{j}^{\text{parent}}(t, \mathbf{x}) = 0,$$

$$\forall j: \ \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{f}_1^{(j)}(t, B_j^{\mathrm{T}}\mathbf{x}, \mathbf{z}^{(j)}) + \mathbf{f}_2^{(j)}(t, B_j^{\mathrm{T}}\mathbf{x}, \mathbf{z}^{(j)}) = 0,$$

where $\mathbf{w}^{(j)} = B_j^{\mathrm{T}}\mathbf{x} + \mathbf{z}^{(j)}$. In this way, elimination of the internal unknowns at the sub-circuit levels gives way to a remaining system at the parent level that has as size the number of unknowns at this level. The resulting approach is a 'Frontal Solver'-like one. Note, that the internal unknowns at the sub-circuit level, $\mathbf{z}^{(j)}$, in general have to be found by solving a nonlinear system of equations.

Another option is to write the whole system as a flat circuit problem in which all unknowns at all levels occur. All analyses give rise to nonlinear systems in which expressions and matrices are linear combinations of parts with the time-derivatives and the remaining parts. Hence in the remainder of this section we will confine ourself to consider just a nonlinear system of equations. The flat system to start from may be written as follows:

Algorithm 1 General nonlinear solver

Let $\mathbf{x}^{(0)}$ and $\mathbf{z}^{(j)}$ be given
for all $l = 0, \ldots, L-1$ **do**
  Let $[\mathbf{z}^{(j)}]^{(0)} = \mathbf{z}^{(j)}$
  for all $k = 0, \ldots, K-1$ **do**
    Solve $\mathbf{Z}_j \Delta[\mathbf{z}^{(j)}]^{(k)} = -\mathbf{f}_j(B_j^{\mathrm{T}}\mathbf{x}^{(l)}, [\mathbf{z}^{(j)}]^{(k)})$
    Define $[\mathbf{z}^{(j)}]^{(k+1)} = [\mathbf{z}^{(j)}]^{(k)} + \Delta[\mathbf{z}^{(j)}]^{(k)}$
  end for
  Let $[\mathbf{z}^{(j)}] = [\mathbf{z}^{(j)}]^{(K)}$ and $[\mathbf{x}^{(l)}]^{(0)} = \mathbf{x}^{(l)}$
  for all $s = 0, \ldots, S-1$ **do**
    Solve $(\mathbf{A} + \sum_{j=1}^{N}[\mathbf{Z}_j]^{-1}\mathbf{B}_j)\Delta\mathbf{x}^{(s)} = -\mathbf{f}_0([\mathbf{x}^{(l)}]^{(s)}, \mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(N)})$
    Define $[\mathbf{x}^{(l)}]^{(s+1)} = [\mathbf{x}^{(l)}]^{(s)} + \Delta\mathbf{x}^{(s)}$
  end for
  Let $\mathbf{x}^{(l+1)} = [\mathbf{x}^{(l)}]^{(S)}$
end for

$$\mathbf{f}_0(\mathbf{x}, \mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(N)}) = 0,$$
$$\mathbf{f}_j(B_j^{\mathrm{T}}\mathbf{x}, \mathbf{z}^{(j)}) = 0, \quad j = 1, \ldots, N,$$

which can be solved in several ways. Let $\mathbf{A} = \partial\mathbf{f}_0/\partial\mathbf{x}$, $\mathbf{C}_j = \partial\mathbf{f}_0/\partial\mathbf{z}^{(j)}$, $\mathbf{Z}_j = \partial\mathbf{f}_j/\partial\mathbf{z}^{(j)}$, and $\mathbf{B}_j = \partial\mathbf{f}_j/\partial\mathbf{y}^{(j)}B_j^{\mathrm{T}}$. Here $\mathbf{A}$ and $\mathbf{Z}_j$ are square matrices. Standard Newton–Raphson uses as Jacobian matrix

$$\mathbf{Y} = \begin{bmatrix} \mathbf{A} & \mathbf{C}_1 & \mathbf{C}_2 & \ldots & \mathbf{C}_N \\ \mathbf{B}_1 & \mathbf{Z}_1 & & & \\ \mathbf{B}_2 & & \mathbf{Z}_2 & & \\ \vdots & & & \ddots & \\ \mathbf{B}_N & & & & \mathbf{Z}_N \end{bmatrix},$$

which has an arrow-like shape. Both algorithms can be covered by the general Algorithm 1. The sub-circuit approach is similar to a hierarchical domain decomposition with 'overlap' (because of the terminals).

Note that in the above algorithm the k-loop can be done in parallel. If $K = S = 1$ the algorithm is just Newton–Raphson for a flat problem. If $K = \infty$ and $S = \infty$ (iteration until convergence) this is the approach mentioned before: nonlinear elimination of the internal unknowns $\mathbf{z}^{(j)}$. A variant that ensures quadratic convergence of the overall procedure can be found in [18]. In [4] an efficient linear solver is described that directly applies to a flat matrix $\mathbf{Y}$ after revealing its inherent arrow-like structure.

In Pstar [29] each parent or sub-circuit is typically treated by looping over all its components, being simple elements (like capacitors, resistors, etc.), or sub-circuits (implying recursion), or devices (the compact transistor models), or items like parameters (evaluate expression).

Algorithm 2 Pstar's ordered items loop [29]

```
for all j = 0, ..., J − 1 (ordered items loop in a parent circuit) do
    if Item(j) = sub-circuit then
        Recursion
    else if Item(j) = device then
        1 Step 'Recursion'
    else if Item(j) = element then
        Do actions for element
    else if Item(j) = parameter then
        Do actions for parameter
    else
        Default actions
    end if
end for
```

A typical action for an element or for a device is to determine its contribution to the right-hand side of the Newton–Raphson correction equation and its contribution to the partial Jacobians $C = (\partial/\partial\mathbf{x})\mathbf{q}(t, \mathbf{x})$ and $G = (\partial/\partial\mathbf{x})\mathbf{j}(t, \mathbf{x})$. For basic, linear, 2-node elements between two nodes a and b the contributions ΔC and ΔG to C and G, respectively, can be compactly written as

Element	Definition	ΔC	ΔG
$R(a,b)$	$i = g * v$		$g\mathbf{e}\mathbf{e}^T$
$R(a,b)$	$v = R * i$		$\mathbf{e}\tilde{\mathbf{e}}^T + \tilde{\mathbf{e}}\mathbf{e}^T - R\tilde{\mathbf{e}}\tilde{\mathbf{e}}^T$
$C(a,b)$	$q = C * v$	$C\mathbf{e}\mathbf{e}^T$	
$L(a,b)$	$\phi = L * i$	$-L\tilde{\mathbf{e}}\tilde{\mathbf{e}}^T$	$\mathbf{e}\tilde{\mathbf{e}}^T + \tilde{\mathbf{e}}\mathbf{e}^T$
$J(a,b)$	$j = j_0$		
$E(a,b)$	$E = E_0$		$\mathbf{e}\tilde{\mathbf{e}}^T + \tilde{\mathbf{e}}\mathbf{e}^T$

Algorithm 3 The main Pstar recursions for time integration [29]

for all $i = 0, \ldots, I-1$ (Time step iteration) **do**
  for all $k = 0, \ldots, K-1$ (Newton iteration) **do**
    Recursion I (Bottom-Up): Assembly of the matrix and the right-hand side [$A\ \Delta\mathbf{x}^{n+1} = \mathbf{b} \equiv -\mathbf{F}(\mathbf{x}^n)$] and the partial decomposition [$A = UL$, $L\mathbf{x}^{n+1} = \mathbf{c} \equiv U^{-1}\mathbf{b}$].
    Add the resulting Schur complement for the terminal unknowns to the corresponding rows and columns of the matrix at the parent circuit.
    Add the part of $\mathbf{c}$ for the terminals to the corresponding coordinates of the right-hand side at the parent circuit level.
    Recursion II (Top-Down): Obtain the terminal values from the parent circuit. Solve the remaining lower-triangular system for the new solution [$\Delta\mathbf{x}^{n+1} = L^{-1}\mathbf{c}$, $\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta\mathbf{x}^{n+1}$. The terminal value at the top level is just the ground value.
  end for
  Recursion III (Bottom-Up): Determine the discretization-error estimation.
end for

Here R, C, L, J, E stand for resistor, capacitor, inductor, current source and voltage source, respectively. Furthermore q stands for the charge stored at a capacitor, and ϕ stands for the flux through an inductor. In the 3rd and 4th column, $\mathbf{e} = \mathbf{e}_a - \mathbf{e}_b$ and $\tilde{\mathbf{e}} = \mathbf{e}_c$ in which $\mathbf{e}_a$, $\mathbf{e}_b$, and $\mathbf{e}_c$ are the unit vectors for $\mathbf{v}_n^a$, $\mathbf{v}_n^b$ and $\mathbf{i}_{\text{Element}}$, respectively.

Depending on actions being done before or after the recursions and passing data to or lifting data from a sub-circuit, or device, one can speak of Top-Down and of Bottom-Up recursions. Both types of recursions are found in Pstar [29].

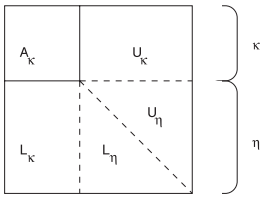
For instance, a typical time integration loop is shown in Algorithm 3. The partial Gaussian Elimination procedure easily accommodates a hierarchical procedure. At a sub-circuit level we decompose the local matrix A in a UL-decomposition (which is due to ordering the terminal unknowns first)

$$A = \begin{pmatrix} A_{\kappa,\kappa} & A_{\kappa,\eta} \\ A_{\eta,\kappa} & A_{\eta,\eta} \end{pmatrix} = UL,$$

$$U = \begin{pmatrix} I & U_\kappa \\ \emptyset & U_\eta \end{pmatrix} \quad \text{and} \quad U^{-1} = \begin{pmatrix} I & \tilde{U}_k \\ 0 & U_\eta^{-1} \end{pmatrix},$$

$$L = \begin{pmatrix} A_\kappa & \emptyset \\ L_\kappa & L_\eta \end{pmatrix}.$$

Clearly: $A_{\eta,\eta} = U_\eta L_\eta$, $L_\kappa = U_\eta^{-1} A_{\eta,\kappa}$, $\tilde{U}_k = -A_{\kappa,\eta} A_{\eta,\eta}^{-1}$, $U_\kappa = -\tilde{U}_k U_\eta$. Also $\text{Diag}(U_\eta) = I$ (which, usually, is not stored). After decomposition the matrix looks like



The Schur complement, A_κ, is added to the parent level as $B_j A_\kappa B_j^{\mathrm{T}}$.

6. DC analysis

A basic analysis is to determine the steady-state, or DC, solution $\mathbf{x}_{\mathrm{DC}}$, which solves $\mathbf{j}(\mathbf{x}_{\mathrm{DC}}) = 0$. In fact many other analyses use as first step an implicit DC analysis. For instance, time integration is usually started by disturbing the DC-solution by time-dependent current or voltage sources. In this case no explicit initial solution is given.

In general the steady-state problem is a nonlinear problem that is much harder to solve than an intermediate nonlinear problem in a time integration procedure. Hence, in this case, the Newton-procedure is combined with continuation methods ($g_{\min}$-stepping) and damping strategies. Even pseudo-transient procedures may be applied. Another problem that has to be dealt with is the occurrence of floating areas (areas that become decoupled when connecting capacitors are skipped).

An analysis one particularly is interested in is fault analysis in which one studies the effect when a resistor is disturbed, or when it is replaced by an 'open' ($g = 0$) or by a short ($R = 0$). Note that the sparse matrix contribution of a linear resistor $R(a, b)$ (with $i = g * v$) to the Jacobian matrix G also allows for efficient fault simulation by applying a so-called one-step relaxation [30]. The Sherman–Morrison–Woodbury formula [21] for expressing the inverse of a perturbed matrix in terms of the inverse of the unperturbed one, allows to re-use the last UL-decomposition of the Jacobian of the DC-analysis. For instance, for an open fault one introduces a defect of $-g\mathbf{e}\mathbf{e}^{\mathrm{T}}$ to the Jacobian and one may use the following expression for the inverse of the Jacobian:

$$J^{-1} = \left[I - \frac{G^{-1}\mathbf{e}\mathbf{e}^{\mathrm{T}}}{-\dfrac{1}{g} + \mathbf{e}^{\mathrm{T}} G^{-1}\mathbf{e}} \right] G^{-1}.$$

Note that fault analysis is a particular application of a more general sensitivity analysis. Here also the decomposition of the Jacobian is re-used, but in the case of the above matrix update a more accurate Jacobian is used than in the general procedure for sensitivity analysis.

7. Time integration

In circuit design, transient analysis is most heavily used. Consequently, there is a constant interest in methods that offer better performance with respect to robustness as well as to reduction of CPU-time.

Because the underlying circuit equations are DAE, an issue for robustness is how well the time integrator behaves when problems of higher index have to be treated: does reduction of the order of integration happen, and does one obtain consistent solutions (i.e., does the solution stay on the manifold defined by the algebraic conditions). Other points of interest are stability conditions [16], and the damping behaviour along the imaginary axis. In fact one prefers no damping along a large part of the imaginary axis, but one insists on damping at infinity.

Common methods for time integration of the differential-algebraic circuit equations are the BDF methods (see [20] to apply Runge–Kutta methods). BDF Methods do not suffer from reduction of the order of integration when applied to DAEs of higher index, and generate consistent solutions [31]. These and other properties, as e.g., variable stepsize control, made these methods very popular for circuit simulation, however at the cost of being conditionally stable when the order of integration exceeds 2. Improvements were looked for in combining BDF with the Trapezoidal Rule [19] (less damping), or in new methods, e.g., Implicit Runge–Kutta methods (IRK) like Radau-methods [10] (3rd order L-stable IRK with options for parallelism), or CHORAL [15] (embedded Rosenbrock–Wanner method of order "(2)3", stiffly accurate and L-stable). A second order L-stable integration method with a so-called high pass filter along the imaginary axis that easily fits an environment developed for BDF-methods is described in [20]. A recent study on Modified Extended BDF methods [7] was also motifated by this.

Other approaches, that start from a boundary-value problem point of view, are Generalized BDF-methods (GBDF) [5], but these can be applied to initial-value problems as well. Parallelism for this last case is considered in [23].

7.1. BDF, Trapezoidal Rule

Consider a linear s-step method to approximate $\mathbf{q}(t_i, \mathbf{x}_i)$ at $t_i = t_0 + i\Delta t$ by

$$\mathbf{q}(t_{i+s}, \mathbf{x}_{i+s}) \approx \sum_{j=0}^{s-1} \alpha_j \mathbf{q}(t_{i+j}, \mathbf{x}_{i+j}) + \Delta t \sum_{j=0}^{s} \beta_j \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{q}(t_{i+j}, \mathbf{x}_{i+j}),$$

with $s \geqslant 1$, $\alpha_0, \alpha_1, \ldots, \alpha_{s-1} \in \mathbb{R}$ and $\beta_0, \beta_1, \ldots, \beta_s \in \mathbb{R}$. Furthermore, $\alpha_0^2 + \beta_0^2 > 0$. To satisfy the algebraic constraints at t_{i+s} we will restrict ourselves to $\beta_s \neq 0$. Assuming $\alpha_s = -1$

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{q}(t_{i+s}, \mathbf{x}_{i+s}) &\approx \frac{-1}{\beta_s \Delta t} \sum_{j=0}^{s} \alpha_j \mathbf{q}(t_{i+j}, \mathbf{x}_{i+j}) - \frac{1}{\beta_s} \sum_{j=0}^{s-1} \beta_j \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{q}(t_{i+j}, \mathbf{x}_{i+j}) \\ &= \frac{1}{\beta_s \Delta t}\mathbf{q}(t_{i+s}, \mathbf{x}_{i+s}) + \frac{-1}{\beta_s \Delta t} \sum_{j=0}^{s-1} \alpha_j \mathbf{q}(t_{i+j}, \mathbf{x}_{i+j}) \\ &\quad + \frac{1}{\beta_s} \sum_{j=0}^{s-1} \beta_j \mathbf{j}(t_{i+j}, \mathbf{x}_{i+j}).\end{aligned}$$

Hence, at t_{i+s}, (1) is approximated by an equation of the form

$$\frac{1}{\beta_s \Delta t}\mathbf{q}(t_{i+s}, \mathbf{x}_{i+s}) + \mathbf{j}(t_{i+s}, \mathbf{x}_{i+s}) + \mathbf{b}_{i+s} = 0.$$

The DAE-nature implies that to be successful, the methods have to be implicit. The BDF integration methods and the Trapezoidal Rule immediately fit into the above formulation. The methods are robust and allow an elegant Local Truncation Error estimation based on the observation that

$$\begin{aligned}\Lambda_{\mathrm{LTE}}(t) &\approx \Lambda_{\mathrm{LTE}}(t_{i+s})\\ &= C_{k+1}\Delta t^{k+1}\frac{\mathrm{d}^{k+1}}{\mathrm{d}t^{k+1}}\mathbf{q}(t_{i+s})\\ &\approx C'_{k+1}(\mathbf{q}(t_{i+s}) - \mathbf{q}^{(0)})\\ &\approx C'_{k+1}\frac{\partial \mathbf{q}}{\partial \mathbf{x}}(\mathbf{x}(t_{i+s}) - \mathbf{x}^{(0)}),\end{aligned}$$

(for some constant C'_{k+1}), where the Newton initialization $\mathbf{x}^{(0)}$ is predicted by extrapolation of appropriate order

$$\mathbf{x}^{(0)} = \mathrm{Extrap}(\mathbf{x}_{i+s-1}, \ldots, \mathbf{x}_{i-1}),$$

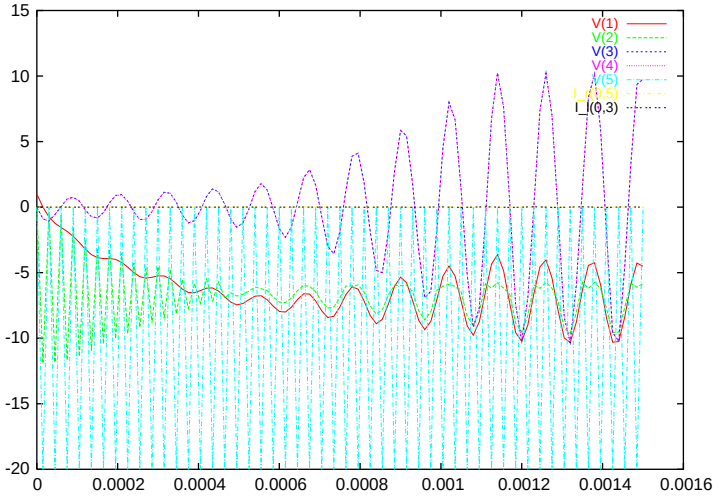
which may be saved or is easily re-evaluated, and $\mathbf{x}(t_{i+s})$ is approximated by the solution of the converged Newton process at time level t_{i+s}. [Note that $\mathbf{x} \in \mathrm{Ker}(\partial\mathbf{q}/\partial\mathbf{x})$ may be omitted.] An alternative is to re-use the Jacobian of the Newton–Raphson process to scale the Λ_{LTE} back to $\mathbf{x}(t_{i+s}) - \mathbf{x}^{(0)}$ [27]. In both cases an upperbound with $\|\mathbf{x}(t_{i+s}) - \mathbf{x}^{(0)}\|_2^2$ arises as controlling quantity. Note that this includes both differences of nodal voltages and the added branch currents. In practice, one restricts oneself to estimate the $\Lambda_{\mathrm{LTE}}(t)$ for the nodal voltage unknowns only. Automatic control techniques [28] are of interest for obtaining results with reduced noise when varying parameters.

We conclude this section with some remarks concerning the consistency of the solution $\mathbf{x}$. Let $\mathbf{P}$ be a time-independent projector of maximum rank that does not depend on the solution with the property $\mathbf{Pq}(t, \mathbf{x}) = 0$ [20]. This immediately implies $\mathbf{Pj}(t, \mathbf{x}) = 0$. With this operator it is easily shown that the BDF methods (for which $b_j = 0,\ \ j = 0, \ldots, s-1$), generate consistent solutions (i.e., they satisfy the algebraic constraints $\mathbf{Pj}(t_{i+s}, \mathbf{x}) = 0$ at time level t_{i+s}), even in the presence of inconsistent solutions at previous time levels. This property is also exploited in the Trap-BDF2 method [19]. For BDF, even the order of integration is independent of the DAE-index [6,31]. The well-known drawbacks are that the unconditionally stable methods are restricted to have at most integration order 2 and that these methods show too much damping along the imaginary axis.

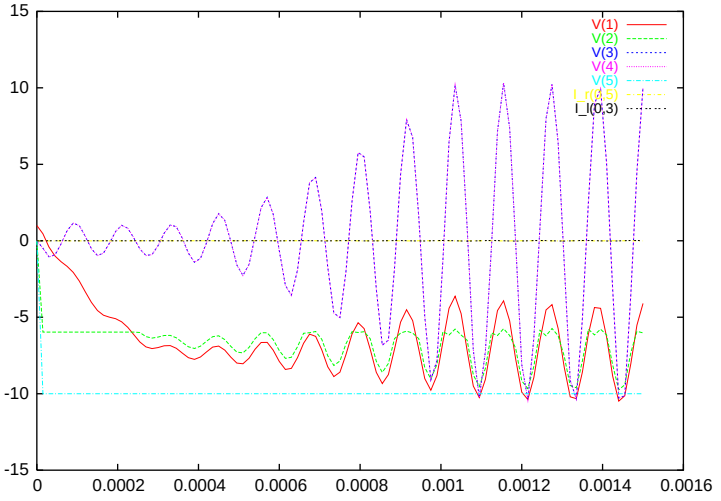
The Trapezoidal Rule has no damping on $i\mathbb{R}$, has integration order 2, but does generate a consistent solution at t_{i+s} only when $\mathbf{Pj}(t_{i+s-1}, \mathbf{x}) = 0$, which assumes a consistent initialization. The method shows reduction of convergence order with increasing DAE-index.

With the operator $\mathbf{P}$, a $\mathbf{P}$-Stabilized Trapezoidal Rule can be formulated, that guarantees consistency:

$$\frac{\mathbf{q}(\mathbf{x}_{n+1}) - \mathbf{q}(\mathbf{x}_n)}{\Delta t} + \frac{1}{2}[\mathbf{j}(\mathbf{x}_{n+1}) + (\mathbf{I} - \mathbf{P})\mathbf{j}(\mathbf{x}_n)] = 0. \tag{2}$$



Waveforms by the TrapezoidalRule without P-stabilisation.



Waveforms by the TrapezoidalRule without P-stabilisation.

Fig. 4. Waveforms of a circuit with DAE-index 2 obtained by the Trapezoidal Rule without and with **P**-stabilization.

The effect is easily demonstrated for a circuit with a DAE-index 2 unknown. Results are shown in Fig. 4. This approach easily generalizes to other linear multistep methods.

We finally note that after multiplying (2) with $(\mathbf{I} - \mathbf{P})$, the **P**-Stabilized Trapezoidal Rule shows a 2nd order time integration order for the equations in the Range of **q**.

7.2. Radau

In [10], IRK methods ere studied, in which each of the internal approximations could be computed concurrently on more processors. It is nice that the *sequential* costs of these methods are of the same

order as those of codes based on BDF, whereas the methods benefit from the advantages of IRKs over BDFs: no order barrier (like Dahlquist's order barrier) and being a one-step method it allows easy change of stepsize and commodation with discontinuities. A two-stage Radau IIA method (a third order, L-stable IRK method) was studied more closely, based on re-using as much of Pstar's current time integration procedure (that, historically, was optimized for BDF-methods and Trapezoidal Rule time integration). The L-stable Radau IIA methods with s-stages have order $2s-1$. These Radau methods have the next time point t_n as final stage level. This is of importance when dealing with DAEs: when all the algebraic equations are satisfied at all stage points, they are automatically satisfied in the step point in which case one automatically generates consistent approximations to the solution [13].

The particular two-stage Radau IIA method for

$$y' = f(t, y), \quad f, y \in \mathbf{R}^d, \quad t_0 \leqslant t \leqslant t_{\text{end}}, \quad y(t_0) = y_0 \tag{3}$$

required one LU-factorization of dimension d per time step and two evaluations of the right-hand side function f per iteration of the nonlinear solver. More precisely, the Radau IIA method is defined by

$$\begin{aligned} &R(Y_n) \equiv Y_n - [1\ 1] \otimes y_{n-1} - h(A \otimes I)F(Y_n) = 0, \\ &Y_n = \begin{pmatrix} g \\ y_n \end{pmatrix}, \quad A = \begin{bmatrix} \frac{5}{12} & -\frac{1}{12} \\ \frac{3}{4} & \frac{1}{4} \end{bmatrix}, \end{aligned} \tag{4}$$

where g, y_n contain approximations at the time levels $t_{n-1} + \frac{1}{3}h$ and t_n, respectively, and $F^{\mathrm{T}}(Y_n) = [f^{\mathrm{T}}(t_{n-1} + \frac{1}{3}h, g_n)\ f^{\mathrm{T}}(t_n, y_n)]$. Eq. (4) may be solved by a modified Newton method (which can be proved to converge)

$$(I - B \otimes hJ)(Y_n^{(j)} - Y_n^{(j-1)}) = -R(Y_n^{(j-1)}), \quad B = \begin{bmatrix} \frac{1}{6}\sqrt{6} & 0 \\ 4 - \frac{4}{3}\sqrt{6} & \frac{1}{6}\sqrt{6} \end{bmatrix}, \tag{5}$$

where J is the Jacobian matrix of f at some previous approximation.

For each transformation Q, $\tilde{A} = Q^{-1}AQ$ has the same complex pair of eigenvalues as A. Writing $\tilde{A} = \tilde{L}\tilde{U}$ one considers $B_Q = Q\tilde{L}Q^{-1}$ and one is interested in those matrices B_Q that have only one double real eigenvalue. The above matrix B can even be obtained by a lower-triangular matrix Q.

Note that B is lower-triangular and that only one matrix of size $d \times d$ has to be inverted. When dealing with more general matrices A of even size, the procedure above can be seen as a step when A is transformed to 2×2 block-diagonal form, in which case further parallelism can help to improve performance.

However, no elegant implementation could be derived, that could easily deal with a datastructure that was tailored to a particular BDF-implementation. Clearly, actual implementation requires a significant effort and to really appreciate the results of the sequential algorithm, benchmarking against other methods (like MEBDF [7]) or Trap-BDF2 [19] will be needed.

7.3. Rosenbrock–Wanner methods

In [15] embedded Rosenbrock–Wanner methods were studied more closely. The methods, being IRK, apply one Newton–Raphson iteration and accuracy control is based on controlling the time step. The method derived is tailored to the circuit equations, where evaluation of Jacobians is relatively cheap when compared to evaluation of the right-hand side. The order is "(2)3". The method is stiffly accurate, L-stable and shows moderate damping along a specified interval $[-i\omega, i\omega]$. Also one has damping for $i\omega$ when

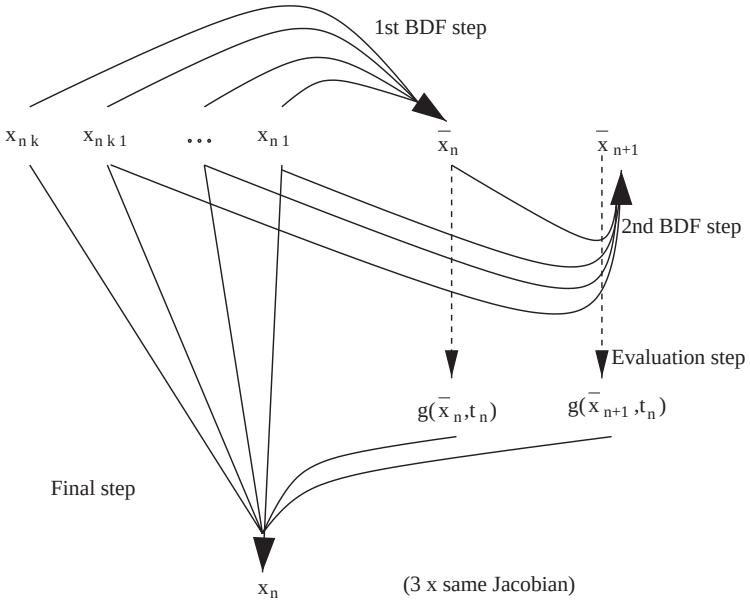


Fig. 5. Steps in the MEBDF.

$\omega \to \infty$. As in the Radau case, implementation did not easily fit an existing BDF-datastructure. However, the properties are rather promising and it is worth to test the method on a large class of industrial problems. Of interest too is that these methods are basis in studying a multi-rate approach in which different stepsizes are used on different locations of the circuit, depending on the activity of the solution [2,3].

7.4. MEBDF

In 1983, J. Cash proposed the MEBDF method, which combines better stability properties and higher order of integration than BDF, but requires more computations per step [8,9]. One timestep with the MEBDF method consists of three BDF steps and an evaluation step. This results in more work compared to BDF, but the order of integration increases with one for most circuits. This implies that for integration order 3 we normally apply the 3-step BDF method, while with the MEBDF method a 2-step method suffices. In [7] (see also [1]) MEBDF was studied when applied to circuit equations. In particular, the focus was on the following quasi-linear DAE:

$$A\mathbf{x}'(t) + \mathbf{g}(\mathbf{x}(t), t) = 0, \tag{6}$$

where $\mathbf{x}(t) \in \mathbb{R}^m$, $t \in \mathbb{R}$ and A is a constant $m \times m$ matrix. The steps in MEBDF are indicated in Fig. 5. For index 1 DAEs, the k-step MEBDF has integration order $k + 1$, while for index 2 DAEs, the order of the k-step MEBDF reduces to k [1,7]. The k-step BDF method does not suffer from order reduction: in either case the k-step BDF has order k [31].

k-step method	Convergence order		consistent solutions
	ODE, $DAE_{index=1}$	$DAE_{index=2}$	
BDF	k	k	+
MEBDF	k+1	k	+

	max. conv. order for A-stability		
	ODE	$DAE_{index=1}$	$DAE_{index=2}$
BDF method	2	2	2
MEBDF method	4	4	3

BDF generates consistent solutions, even if one starts with inconsistent initial values [7,31]. Because BDF serves in predicting for the extrapolation point, MEBDF inherits this property from BDF. This is in contrast to, for instance, solutions obtained by using the Trapezoidal Rule. This property is important when visualizing results for problems with discontinuities. It is known that computing consistent initial values from the DC operating point only requires solving an additional linear system [11]. Another way of getting consistent initial values is integrating forward, till there is no influence of the (inconsistent) initial values and then integrating backward.

The k-step MEBDF-methods are A-stable (see [16]) for $k \leqslant 3$, while for BDF this is restricted to the case $k \leqslant 2$ [8]. Thus, these MEBDF-methods 'break' Dahlquist's Law [16]: we have higher order methods with unconditional stability. When applied to index-1 DAEs, the 3-step MEBDF-method has order 4 and is A-stable. When applied to index-2 DAEs, the order of integration reduces (modestly) to 3, but the method's stability property remains, which compares favourably to the 3-step BDF method. For $k > 3$, the k-step MEBDF-methods become conditionally stable, but also here the stability conditions are better than those for BDF.

As a consequence, the MEBDF-methods are more suited for oscillatory problems than the BDF-methods [7,9].

Concerning implementation, we note that the MEBDF methods use two BDF-approximations at two subsequent time points to improve the result at the first time point. The approach looks attractive because implementation may re-use existing BDF-based datastructures efficiently. In the modified version, also the number of needed LU-factorizations is reduced to only 1. Variants also allow parallelism [12].

Before real implementation, a profound benchmarking on a large set of problems arising from circuit design is worth to be done [1]. Also the behaviour when applied to problems with delays needs to be studied.

8. Outlook and conclusion

High-performance circuit simulation has to deal with a lot of challenges for which at this moment several pragmatic 'engineering' approaches are used and where a sound mathematical background is lacking. In fact simulation has to become orders faster and has to deal with circuits that are orders larger than met before.

These days, circuit simulation becomes really mixed-signal analysis in which a large digital dominated circuitry is combined with a relatively small analog dominated one. In the digital part one can restrict oneself to less accurate compact models of transistors, while in the analog part one will insist on using accurate ones. The digital part is the area for model reduction, preferably done in the time domain. Here table-modeling is frequently encountered. By this combined approach in modeling we can speak of mixed-level simulation.

In this paper we emphasized to exploit the hierarchical circuit description. It is of interest to recognize hierarchy even in flat circuit descriptions. The hierarchical formulation gives rise to a natural form of parallelism and also to successful application of bypassing those hierarchical branches that will behave nearly identical to similar ones that have already been treated before. To different sub-circuits multi-rate time integration schemes may be applied. In practice a balance between a hierarchical formulation and some flattened one has to be found (in the last one feed-back loops are treated correctly in a more natural way).

Finally, in the analog part special techniques are needed for high-performance RF-simulation to deal with multi-tone problems. Here, interesting progress is found in [20,26,24]. We also refer to the overview given in [14].

References

[1] S. Allaart-Bruin, J. ter Maten, S. Verduyn Lunel, Modified Extended BDF time-integration methods, applied to circuit equations, in: W.H.A. Schilders, E.J.W. ter Maten, S.H.M.J. Houben (Eds.), Scientific Computing in Electrical Engineering, Proceedings of SCEE-2002, Eindhoven, ECMI Series Mathematics in Industry, vol. 4, Springer, Berlin, 2004, pp. 86–93.
[2] A. Bartel, Generalised multirate: two ROW-type versions for circuit simulation, M.Sc. Thesis, TU Darmstadt, NatLab. Unclassified Report 2000/804, Philips Research Laboratories, Eindhoven, 2000.
[3] A. Bartel, M. Günther, A multirate W-method for electrical networks in state-space formulation, J. Comput. Appl. Math. 147 (2002) 411–425.
[4] C.W. Bomhof, Iterative and parallel methods for linear systems, Ph.D. Thesis, Utrecht University, 2001.
[5] L. Brugnano, D. Trigiante, Solving Differential Problems by Multistep Initial and Boundary Value Methods, Gordon and Breach Science Publishers, London, 1998.
[6] K.E. Brenan, S.L. Campbell, L.R. Petzold, Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations, SIAM—Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996.
[7] S.M.A. Bruin, Modified extended BDF applied to circuit equations, M.Sc. Thesis, Free University of Amsterdam, NatLab. Unclassified Report 2001/826, Philips Research Laboratories, Eindhoven, 2001.
[8] J.R. Cash, The integration of stiff initial value problems in ODEs using modified extended backward differentiation formulae, Comput. Math. Appl. 9–5 (1983) 645–657.
[9] J.R. Cash, Modified extended backward differentiation formulae for the numerical solution of stiff initial value problems in ODEs and DAEs, J. Comput. Appl. Math. 125 (2000) 117–130.
[10] J.J.B. de Swart, Parallel software for implicit differential equations, Ph.D. Thesis, University of Amsterdam, 1997.
[11] D. Estévez Schwarz, C. Tischendorf, Structural analysis for electric circuits and consequences for MNA, Internat. J. Circuit Theory Appl. 28 (2000) 131–162.
[12] J.E. Frank, P.J. van der Houwen, Diagonalizable extended backward differential formulas, BIT 40-3 (2000) 497–512.
[13] E. Griepentrog, R. März, Differential-Algebraic Equations and their Numerical Treatment, Teubner, Stuttgart, 1986.
[14] M. Günther, U. Feldmann, J. ter Maten, Modelling and discretization of circuit problems, in: W.H.A. Schilders, E.J.W. ter Maten (Guest Eds.), Numerical Analysis in Electromagnetics, Special Volume of Handbook of Numerical Analysis, vol. XIII, Elsevier Science BV, Amsterdam, 2005.
[15] M. Günther, P. Rentrop, U. Feldmann, CHORAL—A one step method as numerical low pass filter in electrical network analysis, in: U. van Rienen, M. Günther, D. Hecht (Eds.), Scientific Computing in Electrical Engineering, Lecture Notes

in Computational Science and Engineering, vol. 18, Proceedings of SCEE-2000, Warnemünde, Springer, Berlin, 2001, pp. 199–215.
[16] E. Hairer, S.P. Nørsett, G. Wanner, Solving Ordinary Differential Equations I: Nonstiff Problems, second ed., Springer, Berlin, 1993.
[17] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems, second ed., Springer, Berlin, 1996.
[18] M. Honkala, J. Roos, M. Valtonen, New multilevel Newton–Raphson method for parallel circuit simulation, in: Proceedings of European Conference on Circuit Theory and Design (ECCTD'01), vol. 2, Espoo, Finland, 2001, pp. 113–116.
[19] M.E. Hosea, L.F. Shampine, Analysis and implementation of TR-BDF2, Appl. Numer. Math. 20 (1996) 21–37.
[20] S.H.M.J. Houben, Circuits in motion: the numerical simulation of electrical circuits, Ph.D. Thesis, Eindhoven University of Technology, 2003.
[21] A.S. Householder, A survey of some closed methods for inverting matrices, SIAM J. Appl. Math. (1957) 155–169.
[22] HSIM V2.0: User's Manual, Nassda Corporation, Santa Clara, CA, USA, 2002, http://www.nassda.com
[23] F. Iavernaro, F. Mazzia, Generalization of backward differentiation formulas for parallel computers, Numer. Algorithms 31 (1–4) (2002) 139–155.
[24] R. Pulch, Numerical techniques for solving multirate partial differential algebraic equations, in: W.H.A. Schilders, E.J.W. ter Maten, S.H.M.J. Houben (Eds.), Scientific Computing in Electrical Engineering, Proceedings of SCEE-2002, Eindhoven, ECMI Series Mathematics in Industry, vol. 4, Springer, Berlin, 2004, pp. 337–348.
[25] P.J. Rabier, W.C. Rheinboldt, Theoretical and numerical analysis of differential-algebraic equations, in: P.G. Ciarlet, J.L. Lions (Eds.), Handbook of Numerical Analysis, vol. VIII, Elsevier Science BV, Amsterdam, 2002, pp. 183–536.
[26] J.S. Roychowdhury, Multi-Time PDEs for dynamical system analysis, in: U. van Rienen, M. Günther, D. Hecht (Eds.), Scientific Computing in Electrical Engineering, Lecture Notes in Computational Science and Engineering, vol. 18, SCEE-2000, Warnemünde, Springer, Berlin, 2001, pp. 3–14.
[27] E.-R. Sieber, U. Feldmann, R. Schultz, H.H. Wriedt, Timestep control for charge conserving integration in circuit simulation, in: R.E. Banks, et al. (Eds.), Mathematical Modelling and Simulation of Electrical Circuits and Semiconductor Devices, International Series of Numerical Mathematics, vol. 117, Birkhäuser Verlag, Basel, 1994, pp. 103–113.
[28] G. Söderlind, Automatic control and adaptive time-stepping, Numer. Algorithms 31 (1–4) (2002) 281–310.
[29] E.J.W. ter Maten, Numerical methods for frequency domain analysis of electronic circuits, Surveys Math. Industry 8 (3–4) (1999) 171–185.
[30] M.W. Tian, C.-J.R. Shi, Nonlinear analog DC fault simulation by one-step relaxation, VTS Best Paper Award, 16th IEEE VLSI Test Symposium, 26–30 April, Monterey, California, 1998, pp. 126–131.
[31] C. Tischendorf, Solution of index-2 differential algebraic equations and its application in circuit simulation, Logos Verlag Berlin, Ph.D. Thesis, Humboldt University, zu Berlin, 1996.
[32] C. Tischendorf, Topological index calculation of DAEs in circuit simulation, Surveys Math. Industry 8 (3–4) (1999) 187–199.