TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Modelling and discretization of circuit problems

Citation for published version (APA):
Günther, M., Feldmann, U., & Maten, ter, E. J. W. (2005). *Modelling and discretization of circuit problems*. (CASA-report; Vol. 0537). Technische Universiteit Eindhoven.

Document status and date:
Published: 01/01/2005

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 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

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.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 06. Nov.. 2023

Modelling and discretization of circuit problems

Michael Günther

University of Wuppertal, Departement of Mathematics, Gaußstr. 20, D-42119 Wuppertal, Germany

Uwe Feldmann

Infineon Technologies, Balanstr. 73, D-81541 München, Germany

Jan ter Maten

Philips Research Laboratories (NatLab), Electronic Design & Tools, Analogue Simulation, Prof. Holstlaan 4, NL-5656 AA Eindhoven, The Netherlands

Contents

Preface

Microelectronics is the core technology for numerous industrial innovations. Progress in microelectronics is highlighted by milestones in chip technology, i.e. microprocessor and memory chips. This ongoing increase in performance and memory density — accompanied with decreasing prices — would not have been possible without extensive use of computer simulation techniques, especially circuit simulation.

An important analysis type in circuit simulators is time domain analysis, which calculates the time-dependent (transient) behaviour of electrical signals in a circuit responding to time varying input signals. A network description of the circuit is generated automatically in computer-aided electronics-design systems from designer's drafts or fabrication data files. An input processor translates this network description into a data format reflecting the mathematical model of the system. The mathematical network equations are based on the application of basic physical laws like energy or charge conservation onto network topology and characteristic equations for the network elements. This automatic modeling approach preserves the topological structure of the network and does not aim at systems with a minimal set of unknowns. Hence an initial-value problem of differential-algebraic equations (DAEs) is generated which covers characteristic time constants of several orders of magnitude (stiff equations) and suffers from poor smoothness properties of modern transistor model equations.

In the first part of this article (Chapter I–III) we aim at filtering out the numerical analysis aspects time domain analysis is based on: The numerical integration of the very special differential-algebraic network equations. This task comprises the simulation core of all simulation packages. Although modelling, discretization and numerical integration can be clearly distinguished as different steps, all these levels are strongly interwoven (and therefore also somehow hidden) in commercial packages.

In Chapter I we discuss how these mathematical models are generated on the basis of a network approach with compact (lumped) models. The structural properties of these DAE models can be described by the DAE-index concept. We will learn in Chapter II that these properties are fixed by the topological structure of the network model in most cases. However, if more general models for the network elements are incorporated, or refined models are used to include second order and parasitic effects then special circuit configurations may be built, which render ill-conditioned problems. These investigations form the basis for constructing numerical integration schemes that are tailored to the respective properties of the network equations. In Chapter III we describe the direct integration approach based on multi-step schemes, which is used in the extremely widespread simulator SPICE [170] and has become a standard since almost 30 years. We include in our discussion a comparison with one-step methods, since recent developments have revealed an interesting potential for such schemes.

The second part (Chapters IV and V) deals with two challenges circuit simulation is faced actually in industry: The simulation of very large circuits with up to millions of transistors such as memory chips on the one hand, and oscillatory circuits with eventually widely separated time constants, appearing in radio frequency (RF) design on the other hand. For the reason of efficiency and robustness, and to make numerical simulation feasible at all, the time domain approach discussed in the first part has to be adapted in both cases. These fields are very much driven by actual industrial needs, and hence are rapidly evolving. So we can in the second part

only describe the state of the art, rather than present an established mathematical theory, as is meanwhile available for numerical integration of DAE systems. Nevertheless we hope that the second part as well as the first one helps to get some feeling about the nature of the underlying problems and the attempts to solve them, and may be useful for both mathematical researchers and the users of the codes.

Chapter I

DAE-Systems — the modelling aspect

In computational engineering the network modelling approach forms the basis for computer-aided analysis of time-dependent processes in multibody dynamics, process simulation or circuit design. Its principle is to connect compact elements via ideal nodes, and to apply some kind of conservation rules for setting up equations. The mathematical model, a set of so-called network equations, is generated automatically by combining network topology with characteristic equations describing the physical behaviour of network elements under some simplifying assumptions. Usually, this automatic modelling approach tries to preserve the topological structure of the network and does not take care to get systems with a minimal set of unknowns. As a result, coupled systems of implicit differential and nonlinear equations, shortly, differential-algebraic equations (DAEs), of the general type

$$f(x, \dot{x}, t) = 0 \quad \text{with } \det\left(\frac{\partial f}{\partial \dot{x}}\right) \equiv 0$$

may be generated. From a mathematical point of view, these systems may represent ill-posed problems, and hence are more difficult to solve numerically than systems of ordinary differential equations (ODEs).

In this first Chapter we have to answer the questions on *how* and *why*: How does one generate the differential-algebraic network equations that model the circuits? And why using at all a DAE approach with a redundant set of network variables, and not an ODE model?

In the subsequent Chapters II and III answers are given to the remaining questions: What are the structural properties of the arising DAE systems? Do they have an impact on numerical discretization? Which integration schemes are used to solve the systems numerically in a robust and efficient manner?

Let us start our discussion with

1 The Schmitt trigger—an introductory example

The Schmitt trigger [133] shown in Fig. 1 is used to transform analogue ones into digital signals. The circuit is characterized by two stable states: If the input signal V_{in} exceeds a certain threshold, then the associated stable state is obtained as an output signal at node 5. The circuit consists of five linear resistors with conductances $G_1, \ldots, G_5$ between the input voltage source V_{in} and node 1, the power supply voltage source V_{DD} and nodes 2 and 5, between node 3 and ground, and between nodes 2 and 4. The dynamic behaviour of the circuit is caused by the linear capacitor with capacitance C_0 between nodes 2 and 4. The nonlinear characteristic is introduced by two bipolar transistors of npn type at nodes 1,2,3 and 4,5,3.

To derive a mathematical model for the Schmitt trigger that determines the time-dependent voltage courses of the five node potentials $u_1, \ldots, u_5$ at nodes $1, \ldots, 5$, we may build up the current balances for all nodes except ground. To apply this so-called *nodal analysis*, we first have to replace all branch currents by voltage-depending functions. For the one-port elements

capacitor and *resistor*, the characteristic equation relating branch current $I(t)$ and branch voltage $U(t)$ is given in admittance form, i. e. $I(t)$ is given explicitly as a function of $U(t)$:

- *Ohm's law for a linear resistor:* $I(t) = GU(t)$ with *conductance* G.
- *Faraday's law for a linear capacitor:* $I(t) = C\dot{U}(t)$ with *capacitance* C.

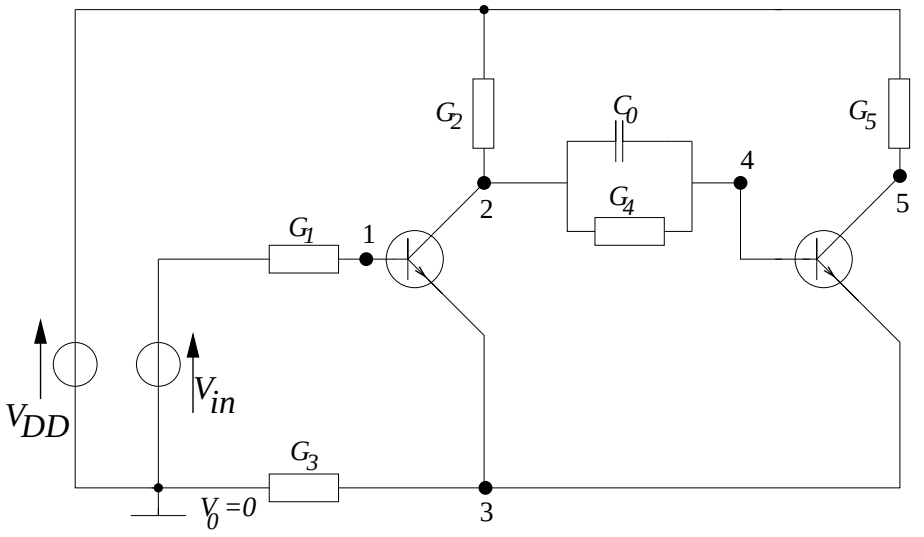


Figure 1: Schmitt trigger circuit

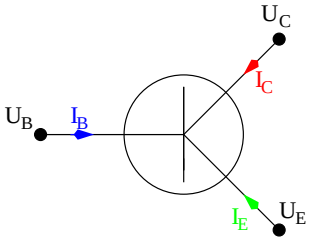


Figure 2: Bipolar transistor of npn type

Using a compact model, the multi-port element *bipolar transistor of npn-type* shown in Fig. 2 can be described by three branch currents $I_B(t)$, $I_C(t)$ and $I_E(t)$ entering the base, collector and emitter terminal of the transistor with corresponding node potentials $U_B(t)$, $U_C(t)$ and $U_E(t)$. If $U_C(t) > U_B(t) > U_E(t)$ then the three currents are given in a first order model by

$$\begin{aligned} I_B(t) &= g(U_B(t) - U_E(t)), \\ I_C(t) &= \alpha \cdot g(U_B(t) - U_E(t)), \\ I_E(t) &= -(1+\alpha) \cdot g(U_B(t) - U_E(t)), \end{aligned}$$

with the characteristic exponential current $g(U) := \beta \cdot [\exp(U/U_T) - 1]$ of a pn-junction. The parameter α denotes the amplification factor, β the saturation current and U_T the thermal voltage at room temperature. For more details see [97, 133].

Now we have collected all ingredients to apply nodal analysis (i.e. apply Kirchhoffs' current law) to nodes 1 to 5. We get:

$$\begin{aligned} \boxed{1} \quad & 0 = G_1 \cdot (u_1 - V_{in}) + g(u_1 - u_3), \\ \boxed{2} \quad & 0 = G_2 \cdot (u_2 - V_{DD}) + C_0 \cdot (\dot{u}_2 - \dot{u}_4) + G_4 \cdot (u_2 - u_4) + \alpha \cdot g(u_1 - u_3), \\ \boxed{3} \quad & 0 = -(1+\alpha) \cdot g(u_1 - u_3) + G_3 \cdot u_3 - (1+\alpha) \cdot g(u_4 - u_3), \\ \boxed{4} \quad & 0 = G_4 \cdot (u_4 - u_2) + C_0 \cdot (\dot{u}_4 - \dot{u}_2) + g(u_4 - u_3), \\ \boxed{5} \quad & 0 = G_5 \cdot (u_5 - V_{DD}) + \alpha \cdot g(u_4 - u_3), \end{aligned}$$

Reformulated as a linear implicit system, we have

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & C_0 & 0 & -C_0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & -C_0 & 0 & C_0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \dot{u}_1 \\ \dot{u}_2 \\ \dot{u}_3 \\ \dot{u}_4 \\ \dot{u}_5 \end{pmatrix} + \begin{pmatrix} G_1 \cdot (u_1 - V_{in}) + g(u_1 - u_3) \\ G_2 \cdot (u_2 - V_{DD}) + G_4 \cdot (u_2 - u_4) + \alpha \cdot g(u_1 - u_3) \\ G_3 \cdot u_3 - (1+\alpha) \cdot g(u_1 - u_3) - (1+\alpha) \cdot g(u_4 - u_3) \\ G_4 \cdot (u_4 - u_2) + g(u_4 - u_3) \\ G_5 \cdot (u_5 - V_{DD}) + \alpha \cdot g(u_4 - u_3) \end{pmatrix} = 0\,. \tag{1.1}$$

The 5×5 capacitance matrix is not regular and has only rank 1: The *network equations* (1.1) are a mixed system of one differential equation (difference of lines 2 and 4) and four algebraic equations (line 1, sum of lines 2 and 4, line 3, line 5). Hence, it is impossible to transform this system of *differential-algebraic equations* (DAEs) analytically to a system of ordinary differential equations (ODEs) by pure algebraic transformations.

With this example in mind, we can now inspect the mathematical modelling of electrical circuits — the set-up of differential-algebraic network equations — in the general case.

2 Principles and basic equations

In contrast to a field theoretical description based on Maxwell's equations, which is not feasible due to the large complexity of integrated electric circuits, the network approach is based on integral quantities — the three spatial dimensions of the circuit are translated into the network topology. The time behaviour of the system is given by the network quantities *branch currents* $I(t) \in \mathbb{R}^{n_I}$, *branch voltages* $U(t) \in \mathbb{R}^{n_I}$ and *node voltages* $u(t) \in \mathbb{R}^{n_u}$, the voltage drop of the nodes versus the ground node. As will be seen later, it may be convenient to include more physical quantities like *electrical charges* $q(t) \in \mathbb{R}^{n_q}$ and *magnetic fluxes* $\phi(t) \in \mathbb{R}^{n_\phi}$ into the set of variables as well.

Network topology laws. The network model consists of elements and nodes, and the latter are assumed to be electrically ideal. The composition of basic elements is governed by Kirchhoff's laws which can be derived by applying Maxwell's equations in the stationary case to the network topology:

— Kirchhoff's voltage law (KVL). The algebraic sum of voltages along each loop of the network must be equal to zero at every instant of time. Often this law is used only for getting a relation between branch voltages $U(t)$ and node voltages $u(t)$ in the form:

$$A^\top \cdot u(t) = U(t) \tag{2.1}$$

with an incidence matrix $A \in \{-1, 0, 1\}^{n_u \times n_I}$, which describes the branch-node connections of the network graph.

— Kirchhoff's current law (KCL). The algebraic sum of currents traversing each cutset of the network must be equal to zero at every instant of time. As a special case we get that the sum of currents leaving any circuit node [1] is zero:

$$A \cdot I(t) = 0. \tag{2.2}$$

When applying KCL to the terminals of an element, one obtains by integration over time the requirement of *charge neutrality*, that is the sum of charges q_{kl} over all terminals k of

[1] The sign is only a matter of convention.

an element l must be constant:

$$\sum_{k(l)} q_{kl} = \text{const.} \tag{2.3}$$

Hereby the constant can be set to zero without loss of generality.

Basic elements and their constitutive relations. Besides these purely topological relations additional equations are needed for the subsystems to fix the network variables uniquely. These so-called characteristic equations describe the physical behaviour of the network elements.

One-port or *two-terminal* elements given in Fig. 3 are described by equations relating their branch current I and branch voltage $U = u_+ - u_-$. Here the arrows in the figure indicate that the branch current is traversing from the "+"-node of higher potential u_+ to the "–"-node of lower potential u_-. The characteristic equations for the basic elements resistor, inductor and capacitor are derived by field theoretical arguments from Maxwell's equations assuming quasistationary behaviour [165]. In doing so, one abstracts on Ohmic losses for a resistor, on generation of magnetic fluxes for an inductor, and on charge storage for a capacitor, by neglecting all other effects. The set of basic elements is completed by ideal independent, i.e. purely time-dependent current and voltage sources.

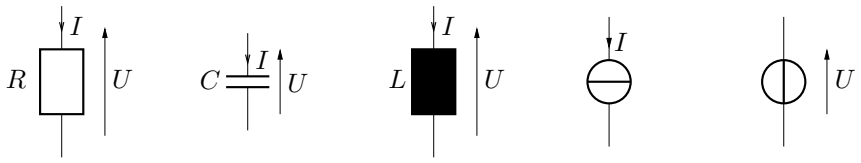


Figure 3: Basic network elements: Linear resistor ($I = U/R = G \cdot U$), capacitor ($I = C \cdot \dot{U}$), inductor ($U = L \cdot \dot{I}$), independent current source ($I = s_1(t)$) and independent voltage source ($U = s_2(t)$).

Figure 4: Controlled sources: Voltage/current controlled current source ($I = \imath(U_{control}, I_{control})$), voltage/current controlled voltage source ($U = v(U_{control}, I_{control})$).

Interconnects and semiconductor devices (i.e. transistors) are modelled by multi-terminal elements (*multi-ports*), for which the branch currents entering any terminal and the branch voltages across any pair of terminals are well-defined quantities. One should note that also for these elements Kirchhoff's laws are valid, i. e. the sum of all branch currents flowing into the element is zero, and the sum of branch voltages along an arbitrary closed terminal loop is zero. Hence, n-terminal elements are uniquely determined by $n-1$ branch currents into $n-1$ terminals and $n-1$ branch voltages between these $n-1$ terminals, and a reference pole. Controlled current sources are used to describe the static branch current; alternatively, controlled voltage sources may be used to describe branch voltages, see Fig. 4 for the symbols. Dynamic behaviour is described by inserting capacitive or inductive branches between the terminals.

These constitutive relations describe the terminal characteristic, which in the *classical approach* is a relation between terminal currents I and branch voltages U and/or their time derivatives for each terminal. If the terminal currents are explicitly given, then the element equations are said to be in *admittance form* (independent current source, voltage/current controlled current source, linear resistor and linear capacitor); if the branch voltages are explicitly given, then the equations are said to be in *impedance form* (independent voltage source, voltage/current controlled voltage source and linear inductor).

As a more flexible and universal approach a *charge/flux-oriented formulation* can be taken for energy storing elements which reflects better the underlying physics of the circuit devices, see e. g. Calahan [31], Chua and Lin [38], Ward and Dutton [253]. It requires the inclusion of terminal charges q and branch fluxes ϕ into the set of network variables.

3 Conventional versus charge/flux oriented formulation

At this point the reader may pose the question why charges and fluxes are introduced at all to model characteristic equations of energy storing elements in a charge/flux oriented way — and not the classical capacitors and inductors are used. The answer to this question contains modelling, numerical and software engineering aspects. We will now focus on the first two aspects. Arguments for the latter will be addressed in the next section.

Modelling. The use of a charge/flux-oriented formulation can be motivated by inspecting the case of nonlinear capacitors and inductors: $C = C(U)$, $L = L(I)$. The problem here is that there is no generic way to get the capacitor current and inductor voltage in this case. Rather there exist different approaches in the literature. We discuss them for capacitors here, the relations for inductors are similar. See Table 1 for an overview.

Conventional formulation	Charge/flux-oriented formulation
Linear capacitor $I = C \cdot \dot{U}$	Charge/current defining element $q = q_C(U, I), \quad I = \mu(U) \cdot q + \dot{q} = I_{dc} + \dot{q}$
Linear inductor $U = L \cdot \dot{I}$	Flux/voltage defining elements $\phi = \phi_L(U, I), \quad U = \dot{\phi}$

Table 1: Constitutive relations for energy storing elements

The most popular is an interpretation of C as *differential capacitance* , with

$$I = C(U) \cdot \dot{U}$$

as capacitor current. However also an interpretation of C as *general nonlinear capacitance* with

$$I = \frac{d}{dt}(C(U) \cdot U)$$

can be found. This interpretation can be transformed into the first one by using

$$\tilde{C}(U) = \frac{\partial C(U)}{\partial U} \cdot U + C(U)$$

as differential capacitance.

A more natural access for the handling of nonlinear capacitances would be to introduce the terminal charges

$$q = q_C(U)$$

and apply the formula

$$I = \frac{dq}{dt}$$

for getting the capacitor current. Another argument for this approach is, that for the classical capacitance definition the controlling branch voltage U is restricted to be the voltage drop over the capacitor itself, which is too much restrictive to handle large classes of circuits. Hence charge-oriented models are highly desirable, not only for getting more flexibility but also since they are consistent with the physical reality: Both static and dynamic behaviour can be derived from one single set of equations, see the equation for the current in the second column of Table 1 [169]. Unfortunately, it is in practice often too difficult to develop such models for real circuit elements with sufficient accuracy. So this ideal principle is often violated in practice, and static and dynamic behaviour are modeled separately.

Charge conservation. A mixture of modelling and numerical aspects is the possibility to correctly model and analyse the charge flow in the circuit. In the following, we will concentrate on the latter item.

The original intent of the charge/flux-oriented formulation was to assure *charge conservation*. This property is crucial for the analysis of many analog circuits like switched capacitor filters, charge pumps, dynamic memories etc., which work on the basis of properly balanced charge flow.

In the following we merely look at charge conservation. The relations for flux conservation are similar. Ideally, charge conservation is assured if

- the principle (2.3) of charge neutrality is observed for each charge storing element, and
- during numerical integration of the network equations no erroneous charges are "created".

In practice, the latter condition can be replaced by the weaker requirement that

- the charge error due to numerical procedures can be made arbitrarily small if the network equations are solved with increasing accuracy.

How can charge conservation be obtained with the different formulations? First we look at the *conventional approach*. Here charges are obtained indirectly via numerical integration of capacitances $C = C(u)$:

$$q(t) = q(t_0) + \int_{u(t_0)}^{u(t)} C(v)dv.$$

Here t_0 is the starting time, and t is the actual time point. The numerically computed voltage u will differ from the exact value u^*:

$$u(t) = u^*(t) + \triangle u(t).$$

So we obtain as a first-order approximation from the capacitance-oriented formulation

$$q(t) \approx q(t_0) + \int_{u(t_0)}^{u(t)} C(v^*)dv + \int_{u(t_0)}^{u(t)} \frac{\partial C}{\partial v} \triangle v dv,$$

while the exact value for the charge is given by

$$q^*(t) = q(t_0) + \int_{u(t_0)}^{u^*(t)} \frac{\partial q(v^*)}{\partial v} dv.$$

Insertion yields

$$q(t) \approx q^*(t) + \int_{u(t_0)}^{u^*(t)} \left[C(v^*) - \frac{\partial q(v^*)}{\partial v} \right] dv + \left(\int_{u(t_0)}^{u(t)} - \int_{u(t_0)}^{u^*(t)} \right) C(v^*)dv + \int_{u(t_0)}^{u(t)} \frac{\partial C}{\partial v} \triangle v dv.$$

The latter two integrals can be made arbitrarily small by improving the accuracy of the numerical procedures. However the first integral is independent of numerical accuracy, and hence the charge obtained from the conventional formulation will approximate the exact value only if

$$C(u^*) - \frac{\partial q_C(u^*)}{\partial u} = 0 \tag{3.1}$$

holds. This requirement concerns the capacitance model. It means that with the conventional formulation charge conservation can only be obtained if the *capacitance matrix* is the Jacobian of a charge vector, i. e. *has a generic function*. A sufficient condition is that the capacitance is controlled only by the branch voltage of the capacitor itself. So in these cases there is a chance to get charge conservation even with a capacitance-oriented formulation, provided that the numerical solution is sufficiently accurate. However, in many models developed so far, the requirement (3.1) is violated — because it implies additional restrictions on the capacitor model for real circuit devices, which is difficult to develop anyway. A well known counterexample is the model of Meyer for MOS capacitances, which has been discussed extensively in the literature,

since it has been found that it violates the charge conservation requirement [167, 210, 253]. See [98] for more details.

With the *charge/flux-oriented formulation* it is not difficult to obtain charge conservation. The first requirement is automatically met with the construction, that for each charge storing element one terminal charge is just the negative sum of all others. To check the second requirement, we expand the numerical approximation of the charge vector around the exact solution:

$$\begin{aligned} q(t) &= q_C(u(t)) = q_C(u^*(t)) + \frac{\partial q_C(u^*(t))}{\partial u} \cdot \triangle u + O(\triangle u^2) \\ &= q^*(t) + \frac{\partial q_C(u^*)}{\partial u} \cdot \triangle u + O(\triangle u^2) \end{aligned}$$

Hence $q(t)$ will approximate the exact charge, as $\triangle u$ becomes smaller with increasing numerical accuracy.

4 Modified Nodal Analysis

The electrical network is now fully described by both Kirchhoff's laws and the characteristic equations in charge/flux oriented formulation. Based on these relations, most computer programs employ one of three schemes to set up the network equations: *Sparse Tableau Approach* (STA, see Hachtel et al. [107]), *Nodal Analysis* (NA, see Chua and Lin [38], or *Modified Nodal Analysis* (MNA, see Ho et al. [114]).

STA is rather canonical: All basic equations are set up explicitly in a system which contains all network variables as unknowns, i.e. node voltages u, branch voltages U and branch currents I. However, even for small circuits, a very large number of mainly short equations is generated.

We should mention that for theoretical investigations mostly an even more flexible extension of STA called *Hybrid Analysis* is used, which takes Kirchhoff's equations in their general form for loops of branch voltages and cutsets of branch currents rather than (2.1, 2.2).

NA. Contrary to STA, the aim of NA is to keep the network equations as compact as possible, so the vector of unknowns x contains only node voltages u. Since voltage sources have no admittance representation, they need a special treatment [38] by which the number of KCL equations and of components of x is reduced by one for each voltage source. Hence the components u_1 of x are a subset of the node voltages u.

Current-controlled sources are difficult to implement, and inductors may lead to integro-differential network equations. Thus NA is not well suited for modelling circuits which contain these elements.

MNA represents a compromise between STA and NA, combining the advantages of both methods. It shares the universality with STA, but has the advantage of a smaller number of unknowns and equations: In addition to the node voltages, the branch currents $\jmath_V$ and $\jmath_L$ of just those elements are included into the vector x of unknowns which have no simple characteristic equations in admittance form, i.e. voltage sources and inductors/flux sources. Therefore it is most commonly used in industrial applications to generate the network equations.

Charge/flux oriented formulation of MNA. To set up the MNA network equations, KCL (2.2) is applied to each node except ground, and the admittance form representation for the branch current of resistors, current sources, capacitors and charge sources is directly inserted. The impedance form equations of voltage sources, inductors, and flux sources are explicitly added to the system of equations. They implicitly define the branch currents of these elements. Finally, all branch voltages are converted into node voltages with the help of KVL (2.1). Splitting the incidence matrix A into the element related incidence matrices A_C, A_L, A_R, A_V and A_I for charge and flux storing elements, resistors, voltage and current sources,

one obtains from MNA the network equations in charge/flux oriented formulation [67]:

$$\begin{aligned} A_C\dot{q} + A_R r(A_R^\top u, t) + A_L \jmath_L + A_V \jmath_V + A_I \imath(A^\top u, \dot{q}, \jmath_L, \jmath_V, t) &= 0, && (4.1a)\\ \dot{\phi} - A_L^\top u &= 0, && (4.1b)\\ v(A^\top u, \dot{q}, \jmath_L, \jmath_V, t) - A_V^\top u &= 0, && (4.1c)\\ q - q_C(A_C^\top u) &= 0, && (4.1d)\\ \phi - \phi_L(\jmath_L) &= 0 && (4.1e) \end{aligned}$$

with *node voltages* u,
branch currents through voltage and flux controlled elements $\jmath_V$ and $\jmath_L$,
charges and fluxes q and ϕ,
voltage dependent resistors r,
voltage and current dependent charge and flux sources q_C and ϕ_L,
controlled current and voltage sources $\imath$ and v.

For an illustration of A_C, A_L, A_R, A_V and A_I we refer to the following example.

The Schmitt trigger again. Let us now return to the Schmitt trigger introduced in Section 1. To apply Modified Nodal Analysis, we have to introduce additional nodes 6 and 7 as terminals of the voltage sources V_{in} and V_{DD}. This yields additional node potentials u_6, u_7 and branch currents $\jmath_{V_1}, \jmath_{V_2}$ through the voltage sources V_{in} and V_{DD} as new network variables. With $u := (u_1, \ldots, u_7)^\top$ and $\jmath_V := (\jmath_{V_1}, \jmath_{V_2})^\top$, the network equations (1.1) for the Schmitt trigger can be written in the charge/flux-oriented form (4.1a,4.1c,4.1d) by defining

$$A_C = \begin{pmatrix} 0\\1\\0\\-1\\0\\0\\0 \end{pmatrix}, A_R = \begin{pmatrix} 1&0&0&0&0\\0&1&0&1&0\\0&0&1&0&0\\0&0&0&-1&0\\0&0&0&0&1\\-1&0&0&0&0\\0&-1&0&0&-1 \end{pmatrix}, A_V = \begin{pmatrix} 0&0\\0&0\\0&0\\0&0\\0&0\\1&0\\0&1 \end{pmatrix}, A_I = \begin{pmatrix} 1&0&0&0\\0&1&0&0\\-1&-1&-1&-1\\0&0&1&0\\0&0&0&1\\0&0&0&0\\0&0&0&0 \end{pmatrix},$$

$$C = C_0, \quad q_C(A_C^\top u) = CA_C^\top u, \qquad G = \operatorname{diag}(G_1, \ldots, G_5), \quad r(A_R^\top u, t) = GA_R^\top u,$$

$$v(A^\top u, \dot{q}, \jmath_L, \jmath_V, t) = \begin{pmatrix} V_{in}(t)\\ V_{DD}(t) \end{pmatrix}, \qquad \imath(A^\top u, \dot{q}, \jmath_L, \jmath_V, t) = \begin{pmatrix} g(u_1 - u_3)\\ \alpha \cdot g(u_1 - u_3)\\ g(u_4 - u_3)\\ \alpha \cdot g(u_4 - u_3) \end{pmatrix}.$$

Note that the circuit does not contain inductors. Hence $\jmath_L = \{\}$, and the contribution $A_L \jmath_L$ does not appear in (4.1a).

Conventional formulation of MNA. Inserting flux and charge relations (4.1d,4.1e) into the first equations, one achieves the analytically equivalent *conventional formulation* of MNA

$$\begin{aligned} A_C C(A_C^\top u) A_C^\top \dot{u} + A_R r(A_R^\top u, t) + A_L \jmath_L + A_V \jmath_V + A_I \imath(A^\top u, \dot{q}_C(A_C^\top u), \jmath_L, \jmath_V, t) &= 0, && (4.2a)\\ L(\jmath_L, t)\dot{\jmath_L} - A_L^\top u &= 0, && (4.2b)\\ A_V^\top u - v(A^\top u, \dot{q}_C(A_C^\top u), \jmath_L, \jmath_V, t) &= 0, && (4.2c) \end{aligned}$$

with generalized capacitance, inductance and conductance matrices

$$C(w) := \frac{\partial q_C(w)}{\partial w}, \quad L(w) := \frac{\partial \phi_L(w)}{\partial w} \quad \text{and} \quad G(w,t) := \frac{\partial r(w,t)}{\partial w}.$$

These matrices are positiv-definite, but not necessarily symmetrical, in contrast to the capacitance, inductance and conductance matrices gained from the two-terminal elements capacitor, inductor and resistor used, for example, in the Schmitt trigger example.

Structure of MNA network equations. Generally, the following properties hold: The matrices

$$\widetilde{C}(A_C^\top u) := A_C C(A_C^\top u) A_C^\top, \quad \text{and} \quad \widetilde{G}(A_R^\top u, t) := A_R G(A_R^\top u, t) A_R^\top$$

are usually very sparse and have structural symmetry.

In some respect, the fine structure of the network equations depends on the type of network elements, on the network topology and on the modelling level:

Type of network elements. There are the trivial conclusions, that the system degenerates to a purely algebraic one if the circuit contains neither capacitors nor inductors (energy storing elements), and that the system is homogeneous if there are no time-dependent elements. If there are no controlled sources, then the Jacobian matrix

$$D(A_R^\top u, t) := \begin{pmatrix} \widetilde{G}(A_R^\top u, t) & A_L & A_V \\ -A_L^\top & 0 & 0 \\ -A_V^\top & 0 & 0 \end{pmatrix} \tag{4.3}$$

with respect to $u, \jmath_L$ and $\jmath_V$ has structural symmetry.

Network topology. Due to Kirchhoff's laws, cutsets of current sources and loops of voltage sources are forbidden. This implies that the matrix (A_C, A_R, A_V, A_L) has full row rank and the matrix A_V has full column rank. If there is a loop of independent voltage sources and/or inductors, or a cutset of independent current sources and/or capacitors, the Jacobian matrix $D(A_R^\top u, t)$ is singular. In these cases no steady-state solution can be computed, and so most circuit analysis programs check and refuse these conditions, which are purely topological. But note that in the nonlinear case the Jacobian matrix also may become numerically singular, e. g. due to vanishing partial derivatives or in the case of bifurcation (in the autonomous case this deals with free oscillators).

The matrix $\widetilde{C}(A_C^\top u)$ is singular, if there are nodes which have no path to ground via energy storing elements. If the circuit contains voltage sources, the MNA network equations contain algebraic relations of type (4.1c) and (4.2c), resp. This is true in most circuits, and so mostly the equations are DAEs, i.e. the Jacobian matrix

$$B(A_C^\top u, t) := \begin{pmatrix} \widetilde{C}(A_C^\top u) & 0 & 0 \\ 0 & L(\jmath_L) & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

with respect to $\dot{u}, \dot{\jmath}_L$ and $\dot{\jmath}_V$ is singular.

Modelling level. Additionally, the modelling level defines some properties of the systems. $\widetilde{C}(A_C^\top u)$ is symmetrical in case of linear or nonlinear differential capacitances, but symmetry may be lost in case of general nonlinear capacitances or nonlinear charge models, as are used for example in MOS transistor models [97, 98].

Charge/flux oriented or conventional MNA? On which formulation — charge/flux oriented or conventional — should the numerical discretization be based, if MNA is used for the automatic generation of network equations? From a structural aspect, the conventional MNA formulation yields a standard form of numerical integration problems, while the charge/flux oriented formulation does not. There are however several reasons, not to transform (4.1) into (4.2) before applying numerical discretization schemes, although they are analytically equivalent:

Structure. (4.1) is of linear-implicit nonlinear form, while (4.2) is of nonlinear-implicit nonlinear form. This may have an impact on the choice of a suitable integrator.

Numerics. Information on the charge/flux level is lost in the conventional approach, and charge conservation may only be maintained approximately in numerical integration schemes.

Implementation. Implicit numerical integration schemes for the conventional MNA equations (4.2) require second partial derivatives of q_C und ϕ_L. These derivative informations, however, are not available in circuit simulation packages, may even not exist because of the lack of smoothness in transistor models.

5 Why differential-algebraic equations?

The charge/flux-oriented formulation of energy storing elements *and* MNA network equations supply us with a first argument for using differential-algebraic equations in electrical circuit modelling. In the following we will assemble more arguments why using a DAE approach with a redundant set of network variables, and not an ODE model.

First of all, one has to distinguish between two different ways to obtain ODE models:

- *Generating a state-space model with a minimal set of unknowns.* Drawbacks of this approach include software engineering, modelling, numerical and designer-oriented arguments. The state-space form cannot be generated in an automatic way, and may exist only locally. The use of independent subsystem modelling, which is essential for the performance of todays VLSI circuits, is limited, and the advantage of sparse matrices in the linear algebra part is lost. Finally, the topological information of the system is hidden for the designer, with state variables losing their technical interpretation.
- *Regularizing the DAE to an ODE model by including parasitic effects.* It is commonly believed that the DAE character of the network equations is only caused by a high level of abstraction, based on simplifying modelling assumptions and neglection of parasitic effects. So one proposal is to regularize a DAE into an ODE model by including parasitic effects. However, this will yield singularly perturbed problems, which will not at all be preferable to DAE models in numerical respect. Beyond it, refined models obtained by including parasitics may make things worse and lead to problems which are more ill-posed.

So we have to inspect feasibility of state-space formulation, subcircuit partitioning and regularization based on including parasitic effects.

State-space formulation: State equations. It is well known that for a large class of nonlinear circuits it is possible to write the network equations as an explicit system of ordinary differential equations of first order, the so-called *State Equations* [38]. For this purpose the vector x_1 of unknowns is constructed only from capacitor voltages and inductor currents [2] — resp. of capacitor charges and inductor fluxes, if a charge/flux-oriented formulation is preferred. In case of special circuit configurations like loops of capacitors or cutsets of inductors, algebraic constraints on the state variables have to be observed (refer also to Section 7), so perhaps not all of them are included into x_1. The resulting system of equations is:

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, s(t), \dot{s}(t)), \\ x_2 &= f_2(x_1, s(t), \dot{s}(t)). \end{aligned}$$

Here s describes independent sources, and x_2 contains those network variables which are not included in x_1, e. g. voltages of nodes to which no capacitive element is connected, or branch currents of voltage sources. The second equation serves for computing x_2, once the set of explicit differential equations (first part) has been solved for x_1 [17]. Note that the equations may contain time derivatives of the input source waveforms.

However, as we can conclude from an extensive literature, the existence of this form is not at all trivial for general classes of nonlinear circuits [38, 39, 161], and therefore the algorithms for setting up the equations are difficult to program and time consuming. Furthermore, compared to NA or MNA, the number of unknowns is extremely large for actual integrated circuits containing a large number of parasitic capacitors. And most important, the structure of the equations does not reflect the structure of the circuit. Therefore this formulation is no longer used in actual circuit simulation programs.

Subcircuit partitioning. The design of memory chips and advanced digital/analog circuits demands the numerical simulation of networks with several ten thousand transistors. Parallel simulation is then valuable to reduce runtime, which otherwise would be prohibitive for such

[2] Note that in network theory just the latter variables are called *state variables*, which is somewhat different from the use of this notation in numerics.

large applications. For this purpose, domain decomposition methods may be employed, requiring to partition the circuit into subblocks which are decoupled by introducing virtual voltage and/or current sources as coupling units at the boundaries [5, 258] (see Sect. 15).

Regard now, for example, two subcircuits only connected by resistive paths, with only linear energy storing elements and resistors, and without any sources. With the partitioned vectors of node voltages $u = (u_1, u_2)^\top$ and branch currents through inductors $\jmath_L = (\jmath_{L1}, \jmath_{L2})^\top$, the network equations read

$$\begin{aligned} A_C\dot{q} + A_R r(A_R^\top u) + A_L\jmath_L + A_V\jmath_V &= 0, && \text{(5.1a)}\\ \dot{\phi} - A_L^\top u &= 0, && \text{(5.1b)}\\ A_V^\top u &= 0, && \text{(5.1c)} \end{aligned}$$

where

$$\begin{aligned} &A_C := \operatorname{diag}(A_{C_1}, A_{C_2}), \quad q = q_C(A_C^\top u) := \operatorname{diag}(C_1, C_2)A_C^\top u,\\ &A_L := \operatorname{diag}(A_{L_1}, A_{L_2}), \quad \phi = \phi_L(\jmath_L) := \operatorname{diag}(L_1, L_2)\jmath_L,\\ &r(A_R^\top u) = GA_R^\top u \end{aligned}$$

and (5.1c) describes the virtual voltage sources and $\jmath_V$ are their branch currents. We will have to deal with DAE models even if the designers assure that all subcircuits are represented by ODE models! This is easily explained by the fact that the network equations (5.1) correspond to Lagrange equations of the first kind: Defining the electric and magnetic energies of both networks by

$$V(u) = \frac{1}{2}\sum_{i=1}^{2} u_i^\top A_{C\,i}C_iA_{C_i}^\top u_i, \quad T(\jmath_L) = \frac{1}{2}\sum_{i=1}^{2} \jmath_{L_i}^\top L_i\jmath_{L_i}$$

yields the Lagrangian

$$\mathcal{L} := T(\jmath_L) - V(u) + \lambda^\top\left(A_V^\top u - 0\right),$$

where the characteristic equations for the virtual voltage sources are added via Lagrangian multipliers λ. $\mathcal{L}$ fulfills the equation

$$\frac{\mathrm{d}}{\mathrm{dt}}\frac{\partial\mathcal{L}}{\partial\dot{x}} - \frac{\partial\mathcal{L}}{\partial x} = \frac{\partial\mathcal{W}}{\partial\dot{x}}$$

for $x = \{q, \phi, \lambda\}$, $\dot{x} = \{\jmath_L, u, \jmath_V\}$, with the dissipative function $\mathcal{W}$ given by

$$\mathcal{W} := \sum_{i=1}^{2} \jmath_{L_i}^\top A_{L_i}^\top u_i + \frac{1}{2}u^\top A_R G A_R^\top u.$$

Here we used the integral quantities charges and fluxes as state variables and Lagrangian multipliers:

$$q(t) := \int_0^t \jmath_L(\tau)\,d\tau, \quad \phi(t) := \int_0^t u(\tau)\,d\tau, \quad \lambda(t) := \int_0^t \jmath_V(\tau)\,d\tau.$$

One notes that the Lagrangian depends only on derivatives of the state variables. This is caused by the fact that the characteristic equations for energy storing elements are differential equations of first order in the state variables u and $\jmath_L$.

Regularization based on including parasitic effects. In general, regularization is based on the assumption that the differential-algebraic form of the network equations is caused by a too high level of simplification in the modelling process, and therefore an ODE formulation can be reached by adding proper "parasitic" effects or elements to the circuit model [71]. One rule-of-thumb is to include parasitic capacitors to ground at each node to get a regular capacitance matrix, and thus an ODE model for the circuit. However, this approach fails, if, for example, a cutset of current source and inductor with inductance L is regularized by adding a small capacitor with capacitance C bridging the cutset [98].

The drawback is that we are confronted with a singularly-perturbed ODE system if C is too small: An additional oscillation with frequency $\omega_1 = 1/\sqrt{LC}$, the eigenfrequency of the regularized system, is invoked through regularization, which overlays the principle voltage courses, and the numerical problems even increase. One example for such an inappropriate regularization is given by the ring modulator, whose numerical problems have been discussed extensively in the literature [51, 118, 133]: Parasitic capacitances in the proposed range of some pF yield additional high-frequency oscillations in the GHz-range, which drastically slows down numerical simulation. Numerical regularization effects become visible only for capacitances thousand times larger, which are not realistic [71]. On the other hand, the DAE model without parasitic capacitors leads to physically correct results, without any numerical problems, if appropriate integration schemes are used.

Besides that, it is not trivial to make sure that a refined modelling based on including parasitic effects will always yield ODE models. Even worse, the numerical problems may increase with the refinement of the model, as will be shown in Section 9 for different levels in the refined modelling of a bipolar ring oscillator. This result can be explained easily by the fact that the DAE index, a measure for the structural properties of DAE systems, increases.

Chapter 2

DAE-index — the structural aspect

So we are faced with network equations of differential-algebraic type when simulating electrical circuits. Before attacking them numerically, we have to reveal the analytical properties of DAEs. In a first step we inspect linear systems and apply, in a second step, the results to nonlinear systems. We will see that for a rather general class of circuits the network topology determines the structural properties of the DAE network equations. However, if more general models for the network elements are incorporated, special circuit configurations apply or refined models are used to include second order and parasitic effects, one may have to cope with ill-conditioned problems.

6 The index concept for linear systems

If a circuit contains only linear elements — or if the system is linearized at an operating point $(x(t_0), \dot{x}(t_0))$, in order to investigate the system behaviour for small signal excitations from that operating point — then the corresponding network equations represent differential-algebraic equations in linear implicit form:

$$B\dot{x} + Dx = s(t), \qquad x(t_0) = x_0\,. \tag{6.1}$$

If we further assume MNA form then $x = (u, \jmath_L, \jmath_V)^\top$ and

$$B = \begin{pmatrix} A_C C A_C^\top & 0 & 0 \\ 0 & L & 0 \\ 0 & 0 & 0 \end{pmatrix},\ D = \begin{pmatrix} A_R G A_R^\top & A_L & A_V \\ -A_L^\top & 0 & 0 \\ -A_V^\top & 0 & 0 \end{pmatrix},\ s = \begin{pmatrix} -A_I \imath(t) \\ 0 \\ -v(t) \end{pmatrix}.$$

Such linear-implicit systems constitute the starting point to classify differential-algebraic equations by a structural property known as the index. In the linear case, this property depends only on the structure of B and D:

ODE-case: B regular. This holds, iff the circuit contains no voltage sources and there are no nodes which have no path to ground via capacitors. In this case, system (6.1) represents a linear-implicit system of ODEs, and can be transformed into the explicit ODE system

$$\dot{x} = B^{-1}(-D \cdot x + s(t)).$$

DAE-case: B singular. In the following we will assume D to be regular. This requirement allows for computing equilibria solutions by an operating point analysis to determine initial values, and can be assured by proper demands on the network topology [97, 101]. Thus multiplying (6.1) with D^{-1} from the left-hand side leads to

$$D^{-1}B \cdot \dot{x} + x = D^{-1} \cdot s(t). \tag{6.2}$$

By Jordan decomposition of

$$D^{-1}B = T^{-1} \begin{pmatrix} \widetilde{B} & 0 \\ 0 & N \end{pmatrix} T$$

with a regular, time independent, matrix T, equation (6.2) can be written after multiplication by T from the left-hand side as

$$\begin{pmatrix} \widetilde{B} & 0 \\ 0 & N \end{pmatrix} T\dot{x} + Tx = TD^{-1}s(t) \tag{6.3}$$

with a regular matrix $\widetilde{B}$ and a nilpotent matrix N. N belongs to the eigenvalue 0 and is of nilpotency ν, i.e. ν is the smallest number such that $N^\nu = 0$, but $N^{\nu-1} \neq 0$. The transformation

$$\begin{pmatrix} y \\ z \end{pmatrix} := Tx, \quad \text{and} \quad \begin{pmatrix} \eta(t) \\ \delta(t) \end{pmatrix} := TD^{-1}s(t)$$

with differential variables y and algebraic variables z decouples this system into an explicit ODE and a nilpotent part:

$$\dot{y} = \widetilde{B}^{-1}(\eta(t) - y), \tag{6.4}$$
$$N\dot{z} = \delta(t) - z. \tag{6.5}$$

The nilpotent part has to be investigated further:

- *Index-1 case: $\nu = 1$, i.e. $N = 0$*
  Now the nilpotent part reads
  $$z = \delta(t); \tag{6.6}$$
  the algebraic variables are explicitly given by the input signal. After one differentiation an explicit ODE system for z is obtained.
- *Higher-index case: $\nu \geq 2$*
  The algebraic variables are only given explicitly after a differentiation process: Differentiation of (6.5) and multiplication with N from the left-hand side yields
  $$N^2\ddot{z} + N\dot{z} = N\dot{\delta}(t) \Longrightarrow z = \delta(t) - N\dot{\delta}(t) + N^2\ddot{z}.$$
  If $\nu = 2$ holds, we cease the process, otherwise it has to be repeated until $N^\nu = 0$:
  $$z = \delta(t) - N\dot{\delta}(t) + N^2\ddot{\delta}(t) - \ldots + (-1)^{\nu-1}N^{\nu-1}\delta^{(\nu-1)}(t)\,. \tag{6.7}$$
  Now the solution depends not only on the input signal, but also on its derivatives! A last differentiation (i.e. the ν-th one) leads to the desired explicit ODE system
  $$\dot{z} = \dot{\delta}(t) - N\ddot{\delta}(t) + N^2\delta^{(3)}(t) - \ldots + (-1)^{\nu-1}N^{\nu-1}\delta^{(\nu)}(t). \tag{6.8}$$
  Here we have assumed that δ is $\nu - 1$-times differentiable to get a continuous solution z. On the other hand, if we allow discontinuous input signals, solutions may only exist in the sense of distributions [189].

Summing up, the solution behaviour of a linear-implicit system of differential equations differs from standard ODE theory in the following sense:

- The solution has to fulfill an algebraic constraint, since $z(t_0)$ is fixed by δ and its higher derivatives at the initial time point t_0. Especially, the solutions do not depend continuously differentiable on the initial values. For $\nu = 1$, this constraint is explicitly given by (6.6). In the higher-index case, however, the constraint is hidden: A differentiation process is necessary to obtain (6.7).
- The system is sensitive to perturbations. Take as example a signal noise, modelled by the input signal δ: Although δ may be very small, its higher derivatives may be arbitrarily large. A severe amplification of perturbations may occur for higher-index problems: We are faced with ill-posed problems.

These analytical results suggest that no severe numerical problems arise in index-1 systems: The algebraic constraint is explicitly given; hence implicit numerical integration schemes for stiff systems such as BDF [81] or ROW-type methods [194] (see Chapter III), which contain a

nonlinear equation solver, are suitable to treat these problems. Additionally, no amplification of round-off errors is to be expected since the system is not sensitive to perturbations.

However, severe numerical problems may arise for systems with nilpotency $\nu \geq 2$: There are hidden algebraic constraints, which can be resolved only by an unstable differentiation process. Regarding perturbations δ entering the right hand side due to inaccurate solutions or due to roundoff errors, terms of order $\delta/h^{\nu-1}$ will enter the solution, where h is the small time discretization parameter.

Since the value of ν defines the behaviour of the system (6.1), both in theoretical and numerical respect, ν is called the *algebraic index* of the linear implicit system (6.1). Additionally, the observations made above motivate three different point of views:

Differential index: To obtain an explicit differential system instead of the linear-implicit system (6.1), we had to differentiate the nilpotent part (6.5). Since numerical differentiation is an unstable procedure, the number of differentiation steps needed to get an explicit ODE system is a measure for the numerical problems to be expected when solving systems of type (6.1). Hence the minimum number of differentiations required is called the *differential index* ν_d of the linear-implicit linear system (6.1).

Perturbation index: We have seen that derivatives of the perturbation enter the solution of (6.1). This observation leads to a new kind of index, which measures the sensitivity of the solutions to perturbations in the equations: The linear-implicit system (6.1) has *perturbation index* ν_p, if derivatives of perturbations up to degree ν_p enter the derivative of the solution.

Tractability index: Finally it was shown that the decomposition of a DAE system into the part governed by regular ODEs, the algebraic part, and the part which can only be solved by performing a differentiation process gives much insight into the nature of the problem. This is especially helpful for analysis and construction of new methods. Griepentrog and März developed a calculus for doing this by using properly constructed chains of projectors, which led to the tractability index concept [89, 158]. We restrict here to the definition of index 1 and 2. To this end we introduce

$$N := \ker B, \qquad S := \{z : Dz \in \operatorname{im} B\}$$

and define: The system (6.1) with B being singular has tractability index 1, if $N \cap S = \{0\}$, i. e. $B_1 := B + DQ$ is nonsingular for a constant projector Q onto N.
If it is not of index 1 then we introduce

$$P := I - Q, \qquad N_1 := \ker B_1, \qquad S_1 := \{z : DPz \in \operatorname{im} B_1\}$$

and define: The system has tractability index 2, if $N_1 \cap S_1 = \{0\}$, i. e. $B_2 := B_1 + DPQ_1$ is nonsingular for a constant projector Q_1 onto N_1.
In the index-2 case, $N \cap S$ comprises just those components, which can be solved only by a differentiation process. An outcome of this index notation is an exact identification, which part of the DAE system needs which smoothness condition to be solvable.

Although the different index concepts were developed for different purposes, it turns out that in most nonpathological cases all of them yield the same number, or differ at most by one. So we are free to select one of them which suits best to our actual item of interest, or is the easiest to compute.

All definitions can be generalized in a straightforward way to nonlinear DAE systems [83, 84, 89, 110].

It remains to determine the index of the charge/flux-oriented network equations (4.1). Due to the charge and flux defining equations (4.1d–4.1e), the index is always ≥ 1 if the circuit contains energy storing elements at all.

7 Network topology and DAE-index for RLC networks

In the linear case, the two-terminal elements capacitor, inductor and resistor are described by linear functions with *positive* capacitance, inductance and resistance. Hence the matrices

$$C := \frac{\partial q_C(w)}{\partial w}, \quad L := \frac{\partial \phi_L(w)}{\partial w} \quad G := \frac{\partial r(w)}{\partial w}$$

of capacitances, inductances and resistances are symmetrical positive-definite. In other words, the elements are strictly passive.

Generalizing this property to the nonlinear case, the local strict passivity of nonlinear capacitors, inductors and resistors corresponds to the positive-definiteness (but not necessarily symmetry) of the so-called generalized capacitance, inductance and conductance matrices

$$C(w) := \frac{\partial q_C(w)}{\partial w}, \quad L(w) := \frac{\partial \phi_L(w)}{\partial w} \quad \text{and} \quad G(w,t) := \frac{\partial r(w,t)}{\partial w}.$$

already introduced in Section 4. If this property of positive-definiteness holds, the network is called an RLC-network.

Topological conditions. Let us first investigate RLC-networks with independent voltage and current sources only. To obtain the perturbation index of (4.1), we perturb the right-hand side of (4.1a–4.1c) with a slight perturbation $\delta = (\delta_C, \delta_L, \delta_V)^\top$ on the right-hand side. The corresponding solution of the perturbed system is denoted by $x^\delta := (u^\delta, j_L^\delta, j_V^\delta)^\top$. One can show that the difference $x^\delta - x$ between perturbed and unperturbed solution is bounded by the estimate

$$\begin{aligned} \|x^\delta(t) - x(t)\| \le \text{const} \cdot \Big(& \|x^\delta(0) - x(0)\| + \max_{\tau\in[0,t]} \|\delta\| + \\ & + \max_{\tau\in[0,t]} \|Q_{CRV}^\top \dot\delta_C\| + \max_{\tau\in[0,t]} \|\bar Q_{V-C}^\top \dot\delta_V\| \Big) \end{aligned} \tag{7.1}$$

using orthogonal projectors Q_C, Q_{CRV} and $\bar Q_{V-C}$ onto $\ker A_C^\top$, $\ker (A_C A_R A_V)^\top$ and $\ker Q_C^\top A_V$, respectively [237]. Since $Q_{CRV}^\top A_C = 0$ holds, the index does not raise, if also perturbations δ_q and δ_ϕ are allowed in the charge and flux defining equations (4.1d–4.1e).

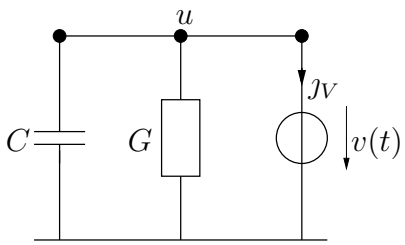
Thus the index of the network equations is one, iff the following two topological conditions hold:

- T1: There are no loops of only charge sources (capacitors) and voltage sources (no VC-loops): $\ker Q_C^\top A_V = \{0\}$ and thus $Q_{CRV} = 0$.
- T2: There are no cutsets of flux sources (inductors) and/or current sources (no LI-cutsets): $\ker (A_C A_R A_V)^\top = \{0\}$ and thus $\bar Q_{V-C} = 0$.

In this case, we are faced with well-posed problems. If however T1 or T2 is violated then we have to cope with ill-posed problems of index 2. The generic index-2 configurations for such a violation are shown in Fig. 5: A VC-loop consisting of voltage source and capacitor, and an LI-cutset of current source and inductor [3].

VC loops. Let us investigate one important case of networks with VC-loops. As we have seen in Section 5, one rule-of-thumb is to regularize a circuit by adding at each node parasitic capacitors to ground. This approach yields $\ker A_C^\top = \{0\}$, and condition T2 is fulfilled. However, T1 is violated due to $Q_C = 0$ iff the network contains voltage sources: Every voltage source leads to a loop of capacitors and this voltage source.

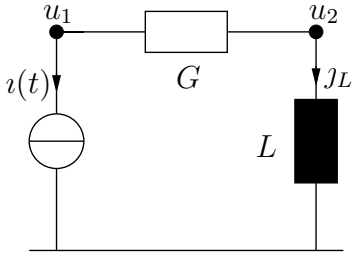
[3] The conductance is only inserted in the figures as a representative to show possible augmentations of the circuit without having an impact on the index.



$$\begin{aligned} \dot{q} + G \cdot u + \jmath_V &= 0 \\ u - v(t) &= 0 \\ q - C \cdot u &= 0 \end{aligned}$$

$$\Rightarrow \jmath_V = -C \cdot \dot{v}(t) - G \cdot v(t)$$

Loop of voltage source and capacitor



$$\begin{aligned} \imath(t) + G \cdot (u_1 - u_2) &= 0 \\ \jmath_L - G \cdot (u_1 - u_2) &= 0 \\ \dot{\phi} - u_2 &= 0 \\ \phi - L \cdot \jmath_L &= 0 \end{aligned}$$

$$\Rightarrow \begin{aligned} u_2 &= -L \cdot i(t) \\ u_1 &= -L \cdot i(t) - 1/G \cdot \imath(t) \end{aligned}$$

Cutset of current source and inductor

Figure 5: Index-2 generic configurations

We determine now the index for both cases. After differentiating the characteristic equations for charge, flux and voltage sources, we get with $\widetilde{C}(w) := A_C C(w) A_C^\top$ a system of the type

$$\begin{pmatrix} \widetilde{C}(A_C^\top u^\delta) & 0 & A_V \\ 0 & L(\jmath_L^\delta) & 0 \\ A_V^\top & 0 & 0 \end{pmatrix} \begin{pmatrix} \dot{u}^\delta \\ \dot{\jmath_L}^\delta \\ \jmath_V^\delta \end{pmatrix} + f(u, \jmath_L, t) = \begin{pmatrix} \delta_C(t) + A_C \dot{\delta}_q(t) \\ \delta_L(t) + \dot{\delta}_\phi(t) \\ \dot{\delta}_V(t) \end{pmatrix} \tag{7.2}$$

with $u^\delta, \jmath_L^\delta, \jmath_V^\delta$ being the solution of (4.1), perturbed with $\delta = (\delta_C, \delta_L, \delta_V, \delta_q, \delta_\phi)^\top$ on the right-hand side. With Kirchhoff's voltage law we have $\ker A_V = 0$, and thus the system can be resolved for $(\dot{u^\delta}, \dot{\jmath}_L^\delta, \jmath_V^\delta)$. For networks without voltage sources the index is one, otherwise two.

One notes that system (5.1) generated by subcircuit partitioning in Section 4 represents a special linear case of (7.2): $C(A_C^\top u) = C$ and $L(\jmath_L) = L$. In this case, we can derive sharper perturbation estimates than (7.1). The difference between the differential part $y := (u, \jmath_L)$ and $y^\delta := (u^\delta, \jmath_L^\delta)$ of unperturbed and perturbed solution is bounded by

$$\|y(t) - y^\delta(t)\| \le \text{const} \cdot \left(\|y(0) - y^\delta(0)\| + \max_{\tau \in [0,t]} \|\delta_1\| + \max_{\tau \in [0,t]} \left\| \int_0^\tau \delta_0(\tau) d\tau \right\| \right)$$

with $\delta_0 = (\delta_L, \delta_C)$ and $\delta_1 = (\delta_V, \delta_q, \delta_\phi)$ — no derivatives of perturbations enter the estimate for the differential variables. But for the algebraic components $\jmath_V$ one gets with the sharp estimate

$$\|\jmath_V(t) - \jmath_V^\delta(t)\| \le \text{const} \cdot \left(\|y(0) - y^\delta(0)\| + \max_{\tau \in [0,t]} \|\delta\| + \max_{\tau \in [0,t]} \|\dot{\delta}_1\| \right)$$

an index-2 behaviour, as expected. In general however, derivatives of perturbations cannot be neglected in the bounds of both differential and algebraic components [4].

Generalization. A generalization of these results for RLC-networks with independent voltage and current sources to special linear controlled sources is given in [192], where linear active networks are considered of capacitors, inductors, resistors, ideal transformers and gyrators. The results hold also for RLC-networks with a rather large class of nonlinear voltage and current sources: The index depends only on the topology; in general, the index is one, and two only for special circuit configurations [67, 98, 237]. This class of sources contains, for example, controlled current sources not being part of VC-loops that are controlled by $(A_C A_V A_R)^\top u$, $\jmath_V$ and t.

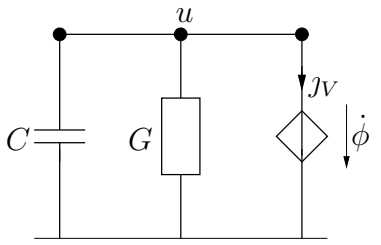
One example for such an RLC-network is given by the Schmitt trigger already introduced in Section 1. Inspecting the charge-/flux-oriented network equations (4.1) derived for the Schmitt trigger in Section 4, we see that the current sources I_B, I_C and I_E describing the bipolar transistors are only controlled by the branch voltages $A_V^\top u$ and $A_R^\top u$. Since $\ker Q_C^\top A_V = \{0\}$ due to

$$Q_C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and $\ker(A_C A_R A_V)^\top = \{0\}$ hold, the Schmitt trigger yields an index-1 problem.

These results obtained for RLC circuits rest on two different types of assumptions: Positive-definiteness of generalized capacitance, inductance and conductance matrices on the one hand, and no arbitrary controlled sources on the other hand. If one of these demands is violated, the index may depend not only on whether the topology conditions T1 and T2 hold or not, but also on circuit and model parameters and — for circuits containing nonlinear elements — on their bias conditions. In addition, the index can be larger than two.

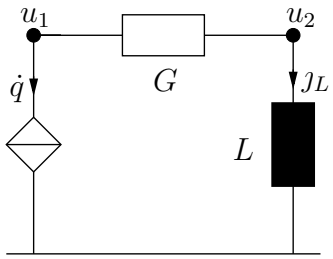
Violation of positive-definiteness. Independent charge and flux sources, which may model α-radiation or external magnetic fields in a somewhat higher level of abstraction, can destroy the positive-definiteness of generalized capacitance and inductance matrices. Generic examples for this case are the circuits of Fig. 6: ΦC-loop and QL-cutsets. We see that u_1 and u_2 (resp. $\jmath_V$) are index-3 variables in the cutset (loop) circuit. It has to be checked whether this mechanism may also lead to index-3 problems in case of MOS circuits with charge models whose derivatives vanish under certain bias conditions.



$$\begin{aligned} \dot q + G\cdot u + \jmath_V &= 0 \\ \dot\phi - u &= 0 \\ q - C\cdot u &= 0 \\ \phi - \phi_L(t) &= 0 \end{aligned}$$

$$\Rightarrow \jmath_V = -C\cdot\ddot\phi_L(t) - G\cdot\dot\phi_L(t)$$

Loop of flux source and capacitor



$$\begin{aligned} \dot q + G\cdot(u_1-u_2) &= 0 \\ \jmath_L - G\cdot(u_1-u_2) &= 0 \\ \dot\phi - u_2 &= 0 \\ q - q_C(t) &= 0 \\ \phi - L\cdot\jmath_L &= 0 \end{aligned}$$

$$\Rightarrow \begin{aligned} u_2 &= -L\cdot\ddot q_C(t) \\ u_1 &= -L\cdot\ddot q_C(t) - 1/G\cdot\dot q_C(t) \end{aligned}$$

Cutset of charge source and inductor

Figure 6: Index-3 configurations: ΦC-loop and QL-cutset.

8 Networks with controlled sources

Higher index can also be generated by controlled sources. One example is the coupling of index-2 problems via controlled sources. Another example is that although both topological conditions T1 and T2 hold, circuit parameters may have an impact on the structural properties of the network equations if a network contains controlled sources, even it is linear. First we discuss the effects of coupling circuits with controlled sources; then we analyze a *differentiator circuit* and a *Miller integrator*, for illustration of the second phenomenon.

Before doing this we should note that controlled sources are indispensable elements in circuit simulation, which are extensively used in semiconductor models as well as in macro models of a somewhat higher level of abstraction, and for modelling signal propagation on and between interconnects. An instructive example for the latter use and its effects on the index is discussed in Section 9.

Coupling of higher-index configurations via controlled sources *may* raise the index of the driven circuit part by one or two per controlled source, if the controlling network variable itself is of higher index. The question, in which cases the index gets higher, is difficult to answer. Below we will give some simple generic cases. Surprisingly, much sophisticated research in this field was done twenty years ago, although the index notion was not yet introduced at all. The motivation was to develop algorithms for setting up network equations in the *State Variable* approach [97] for general classes of networks including controlled sources, and the methods developed for this purpose aimed just to capture as many circuit configurations as possible, which in our notation are of index ≤ 2. Most of them start from properly constructed *normal trees* spanning the network graph. An overview can be found in [32, 38].

There are eight possibilities to couple cutsets of current sources/inductors (JL cutsets) and loops of voltage sources/capacitors (VC loops) via either a voltage-controlled element or a current-controlled element. It turns out that only those configurations will have an increased index where the controlling variable itself is of higher index. These configurations are listed in Table 2. Here the subscript C (D) denotes network elements and variables of the controlling (driven) circuit. The argument t characterizes the input variable, and a prime $'$ indicates the derivative with respect to the controlling variable.

Case	Controlling circuit	Input variable	Driven circuit	Controlled source	Output variable	Index
1	JL cutset	J(t)	JL cutset	$J_D(u_C)$	$u_D = L_C L_D J_D' \ddot{J}(t)$	3
2	JL cutset	J(t)	VC loop	$V_D(u_C)$	$I_D = L_C C_D V_D' \ddot{J}(t)$	3
3	VC loop	V(t)	VC loop	$V_D(I_C)$	$I_D = C_C C_D V_D' \ddot{V}(t)$	3
4	VC loop	V(t)	JL cutset	$J_D(I_C)$	$u_D = C_C L_D J_D' \ddot{V}(t)$	3

Table 2: Index-3 coupling of index-2 circuits via controlled sources

Extensions are possible by replacing in the VC loops and JL cutsets the inductor or voltage source by a flux source, and the current source or capacitor by a charge source. These extensions lead to 32 further high-index configurations, and one can get a circuit configuration of index 5 with only 4 elements [98]. Further extensions are possible by replacing the sources by norators.

The higher-index configurations described here may be recursively used, thus obtaining circuit configurations of arbitrary high index.

Differentiator circuit We must expect a higher-index problem if the circuit itself acts as a differentiator. A differentiator circuit with input source $v(t)$ and output voltage u_3 is given in Fig. 7 where the operational amplifier (see Fig. 8) with amplification factor a is a special case for a voltage-controlled voltage source.

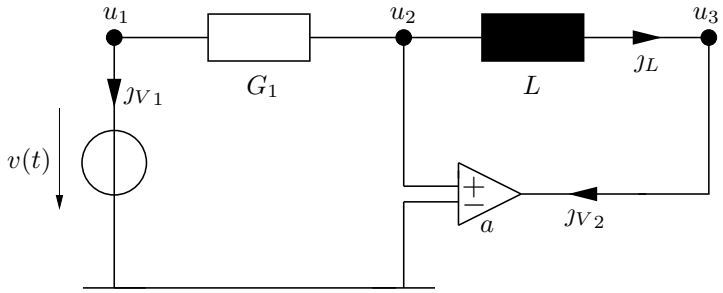
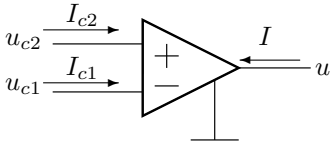


Figure 7: Differentiator circuit



$u = a \cdot U_{control}$,
$U_{control} = u_{c2} - u_{c1}$
$I_{c1} = -I_{c2} = 0$

Figure 8: Operational amplifier: Network symbol and characteristic equations

From its MNA equations

$$\begin{aligned} A_R G A_R^\top u + A_L \jmath_L + A_V \jmath_V &= 0 \\ \dot{\phi} - A_L^\top u &= 0 \\ A_V^\top u - \begin{pmatrix} v(t) \\ a u_2 \end{pmatrix} &= 0 \\ \phi - L \cdot \jmath_L &= 0 \end{aligned}$$

with $G := G_1$ and

$$A_R = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \quad A_L = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix} \quad \text{and} \quad A_V = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$$

one obtains index 1.

For the limit case of an ideal operational amplifier, i. e. $a \to \infty$, the element relation has $U_{control} = 0$ ($u_2 = 0$) as a limiting case, and neither the output voltage U (u_3) nor the output current I ($\jmath_{V2}$) are determined by the characteristic equations. So, its controlling nodes are connected by a *nullator* (i.e. an element with vanishing branch voltage and branch current), and its output nodes are connected by a *norator* (i.e. an element with arbitrary branch voltage and branch current).

In this case, the MNA structure (4.1) is destroyed, since

$$A_V^\top u - \begin{pmatrix} v(t) \\ a u_2 \end{pmatrix} = 0$$

is replaced by

$$\widetilde{A}_V^\top u - \begin{pmatrix} v(t) \\ 0 \end{pmatrix} = 0 \quad \text{with} \quad \widetilde{A}_V^\top = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \neq A_V^\top .$$

We recognize that the *static* elements, whose element equations are independent of the output variables (here: The degenerated controlled source), are responsible for the higher index,

because now the output variable is determined only via differential equations of the *dynamic* elements. We have $u_3 = -L \cdot G \cdot \dot{v}(t)$ for the differentiator circuit, i.e. the output voltage is the time derivative of the input voltage, and the problem is of index 2. This situation is typical for higher-index problems and is merely an electrical interpretation of the mathematical condition for index ≥ 2.

Any extension of the differentiator circuit, which does not shortcut the inductor in Fig. 7, will keep the index ≥ 2. By inserting LC-, LR- or RC-circuits into the feedback loop between ideal operational amplifier and inductor of the differentiator circuit, the index can be raised by 2, 1 or 1, respectively [191].

Miller integrator. When we replace the inductor of the differentiator circuit by a capacitor then the circuit turns into an integrator. Fig. 9 shows such a circuit, which is called Miller integrator. The capacitor C_2 is mandatory for the circuit, while C_1 is added as a parasitic grounded capacitance, which may vanish.

We take this circuit to illustrate that due to the use of controlled sources the circuit parameters may have an impact on the structural properties of the network equations: Index, sensitivity of the solution with respect to input signals, and degree of freedom for assigning initial values. This is true even for linear circuits.

The function of this time-continuous version of an integrator is to integrate an input signal over time. Such integrators are important parts of integrated filter circuits, since they are used to substitute inductors of arbitrary inductance L, which are expensive to obtain otherwise in integrated technologies. Hereby the inductor relation is taken in admittance form

$$\jmath = \frac{1}{L} \int u \, dt,$$

which requires the integration of the branch voltage u. For the sake of simplicity we use an ideal operational amplifier element with limited amplification a here.

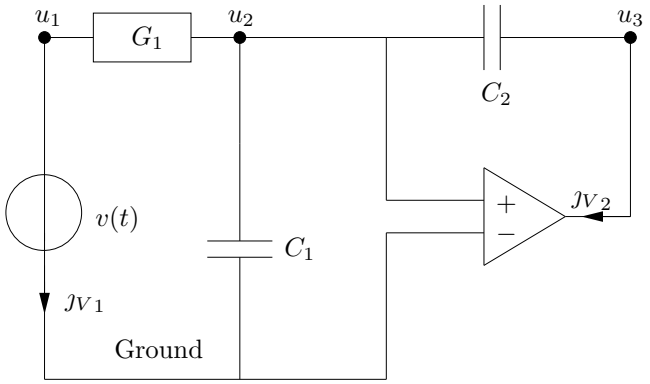


Figure 9: Miller integrator circuit

Using Modified Nodal Analysis, the network equations read

$$\begin{aligned} A_C C A_C^\top \cdot \dot{u} + A_R G A_R^\top \cdot u + A_V \jmath_V &= 0, \\ A_V^\top u - v(u,t) &= 0, \end{aligned}$$

with

$$A_C = \begin{pmatrix} 0 & 0 \\ 1 & -1 \\ 0 & 1 \end{pmatrix}, \quad A_R = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \quad A_V = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix},$$

$$C = \operatorname{diag}(C_1, C_2), \quad G = G_1, \quad v(u,t) = \begin{pmatrix} v(t) \\ a u_2 \end{pmatrix}.$$

Technical parameter	Index	Degree of freedom	Sensitivity w.r.t. $v(t)$
$\underline{C_1 > 0}$			
$a \neq 1 + C_1/C_2$	2	only u_2	only $v(t)$
$a = 1 + C_1/C_2$	3	—	$v(t)$ <u>and</u> $\dot{v}(t)$
$\underline{C_1 = 0}$			
$a \neq 1$	1	only u_2	only $v(t)$
$a = 1$	2	—	only $v(t)$

Table 3: Miller integrator circuit: Impact of technical parameters on index, degree of freedom and sensitivity with respect to input signal

If the amplification factor a tends to infinity then $u_2 = 0$, and for the capacitor current we get $C_2 \cdot \dot{u}_3 = -\jmath_{V2} = -G \cdot u_1 = -G \cdot v(t)$, from which follows the integrator function of the circuit:

$$u_3 = -\frac{G}{C_2} \int v(t)dt.$$

For $a \neq 1 + C_1/C_2$, one can solve for $\dot{u}_2$ by inserting the last equation into the second: $\dot{u}_2 = G(v(t) - u_2)/C$ with $C = C_1 + C_2(1 - a)$. All components are now fixed by u_2, the only degree of freedom:

$$u_1 = v(t),\ u_3 = au_2,\ \jmath_{V1} = G(u_2 - v(t)),\ \jmath_{V2} = \frac{C_2}{C} G(1 - a)(v(t) - u_2).$$

For $a = 1 + C_1/C_2$, however, u_2 is fixed by the hidden algebraic relation $u_1 - u_2 = 0$. Now the solution is given at every time point by the input signal and its derivatives:

$$u_1 = u_2 = v(t), u_3 = av(t), \jmath_{V1} = 0, \jmath_{V2} = C_2(1 - a)\dot{V}(t).$$

This reflects the impact of the technical parameters C_1, C_2 and a on the system w. r. t. input signals and and degree of freedom for assigning initial values, see Table 3 for an overview. It remains to discuss the influence on the index:

First case: $C_1 > 0$. The derivative of the algebraic part with respect to the algebraic components $(u_1, \jmath_L, \jmath_V)^\top$ is singular, since the element relation for the amplifier $u_3 = au_2$ does not depend on $\jmath_{V2}$. After one differentiation we get by inserting the formulae for the differential variables u_2 and u_3 the linear algebraic relation

$$\begin{pmatrix} -(1-a)G_1/C_1 \\ (1-a)G_1/C_1 \\ (1-a)/C_1 + 1/C_2 \end{pmatrix}^\top \begin{pmatrix} u_1 \\ u_2 \\ \jmath_{V2} \end{pmatrix} = 0 \tag{8.1}$$

in $u_1, u_2, \jmath_{V2}$, which can be solved for $\jmath_{V2}$ iff $a \neq 1 + C_1/C_2$; in this case, the index is two. This is not surprising, since together with C_1 and C_2 the operational amplifier forms a loop of voltage source and capacitors.

For the exceptional case $a = 1 + C_1/C_2$ the relation (8.1) reads

$$u_1 = u_2,$$

and a second differentiation is necessary to solve for $\jmath_{V2}$: The index is now three.

Second case: $C_1 = 0$. The partial derivative of the algebraic part with respect to the algebraic components $(u_1, u_2 + u_3, \jmath_{V1}, \jmath_{V2})^\top$ now reads

$$\begin{pmatrix} G & -G/2 & 1 & 0 \\ -G & G/2 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & (1-a)/2 & 0 & 0 \end{pmatrix}.$$

The matrix is regular, and correspondingly the index is 1, iff $a \neq 1$ holds. For $a = 1$, however, the matrix has only rank 3. The last algebraic relation becomes $u_2 - u_3 = 0$, which fixes the differential component, too. To get the remaining differential relation from this equation, two differentiations are necessary. Hence the index is 2.

Conclusion These results for controlled sources have an important practical consequence: It is not sufficient to rely only on structural aspects when trying to cope with higher-index problems in circuit simulation. This will be further elaborated when looking at stepwise refinements of a bipolar ringoscillator model in the following section. Possible solutions to this problem are discussed in Section 10.

9 Effects of refined modelling — a bipolar ringoscillator

The task of a ringoscillator is to generate autonomously an oscillating signal, which may be used for driving other parts of a circuit, but in many cases serves only for measuring the maximal clock rates which can be achieved with a given technology. The basic principle is to connect an odd number of inverter stages in a loop. Compared to standard MOS technologies, bipolar technologies are faster (by approximately an order of magnitude, such that frequencies of 10 GHz and higher are possible) due to a very small signal swing and high driving capabilities, but the circuits are not as compact and have a higher power consumption.

The basic model. A circuit diagram of our bipolar ringoscillator is shown in Fig. 10. Since it is simplified as far as possible it may look somewhat strange for an experienced circuit designer. On the other hand it has still its basic functionality, and can be extended in such a way that we observe the effects we want to discuss here.

Circuit description. The dashed box contains the core of the circuit and will be used as an icon in the extensions discussed later. It consists of three differential stages. The nodes between the resistors and the collector of the bipolar transistors (e.g. the nodes 1 and 2 for the left stage) are the outputs of each stage, while the nodes connected to the base of the bipolar transistors (e.g. the nodes 7 and 8 for the left stage) are its inputs. Each output of a stage is connected to the corresponding input of the next stage, thus forming a loop. Basically the circuit works in a *current mode*: The differential stages are driven by current sources, and due to the exponential characteristic of the bipolar transistor just that branch of each differential stage will take over almost all of the current, whose input node is at a higher voltage level. Since the Ohmic resistors cause a larger voltage drop for the branch carrying the larger current, its output will be at a lower voltage level, thus inverting the input signal.

In principle, one input of each differential stage may be fixed at a constant reference voltage. For speed advantages, often the complementary technique shown here is used, where both the original signal and its inverse are generated in each stage and propagated to the inputs of the next. Note that the circuit operates with negative voltages, which reduces the sensitivity of the signals with respect to perturbations of the power supply. First-order formulas for designing such an oscillator can be found in the textbooks (see e. g. [115]).

Network equations. With $u := (u_1, \ldots, u_{10})^\top$, being the vector of node voltages at nodes $1, \ldots, 10$, and $\jmath_V = J_{SS}$ being the current through the only voltage source, the charge oriented MNA network equations read:

$$\begin{aligned} A_C\dot{q} + A_R\mathrm{diag}(G_1, G_2, G_3, G_4, G_5, G_6)A_R^\top u + A_V\jmath_V + A_I\imath(A^\top u, t) &= 0, \\ A_V^\top u + V_{SS}(t) &= 0, \\ q - q_C(A_C^\top u) &= 0, \end{aligned}$$

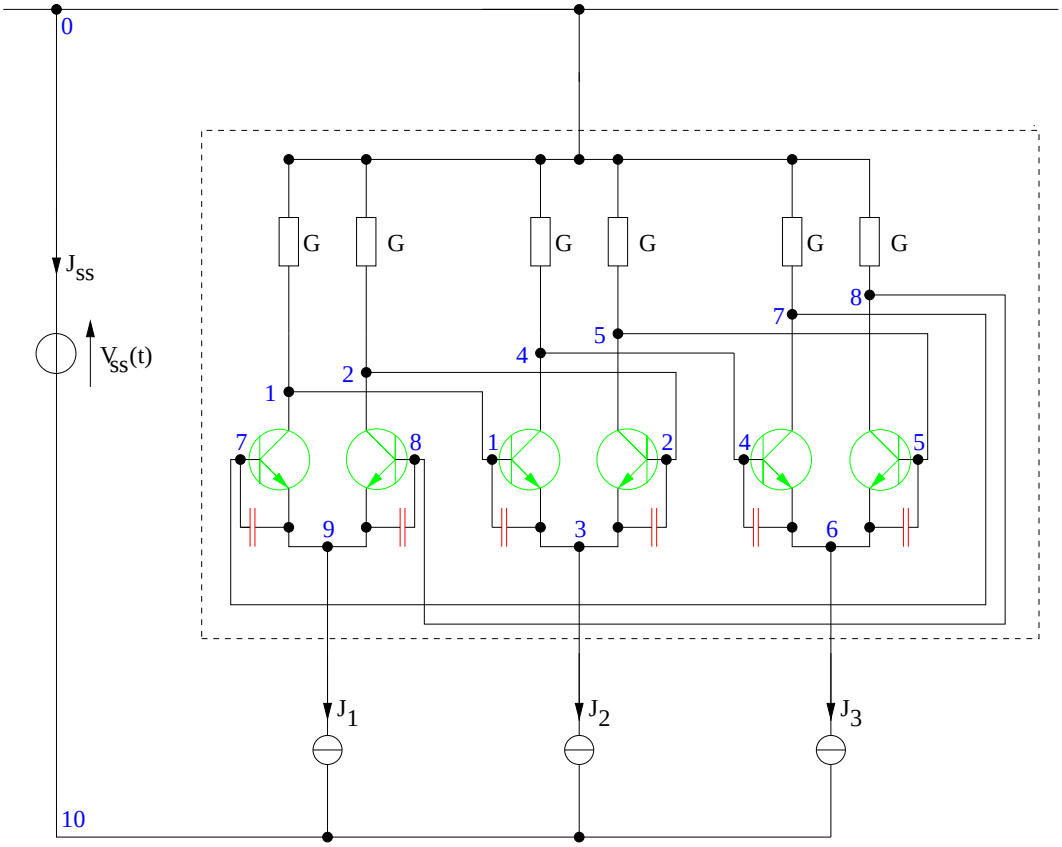


Figure 10: Bipolar ringoscillator

with

$$A_C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad A_R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad A_V = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix},$$

$$A_I = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 \end{pmatrix}, \quad \text{and}$$

$$q_C(A_C^\top u) = \mathrm{diag}(c_{13}, c_{23}, c_{46}, c_{56}, c_{79}, c_{89}) \cdot A_C^\top u,$$
$$\imath(A^\top u, t) = (I_{C1}, \ldots, I_{C6}, I_{B1}, \ldots, I_{B6}, J_1(t), J_2(t), J_3(t))^\top .$$

Here I_{Cj} and I_{Bj} are the collector and base current of the bipolar transistor $T_j (j = 1, \ldots 6)$ introduced in Section 1. The capacitances c_{ij} between nodes i and j may be linear, or modelled in a nonlinear way: $c_{ij} = c_{ij}(A_C^\top u)$ [98].

The index. With the projector

$$Q_C = \begin{pmatrix} 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

onto $\ker A_C^\top$ one shows that $\ker Q_C^\top A_V = \{0\}$ and $\ker(A_C A_R A_V)^\top = \{0\}$ hold. Since all sources are either independent or current sources that are not part of any VC loop and which are driven by branch voltages of capacitive paths, this model yields an index-1 problem. We only have to require that the charge model used for the capacitors in the nonlinear case yields a positive-definite generalized capacitance matrix.

Refined modelling. Eventually our basic circuit model has to be refined in order to get a higher degree of accuracy and to take non-ideal operating conditions into account. Basically this is achieved by

- replacing idealized network elements by real circuits. As an example we will discuss the substitution of the current sources by transistor configurations,
- a more detailed modelling with respect to parasitic effects (see Table 4 for an overview).

The impact of a model refinement on the index is not a priori clear: Regularization to lower index, no change, and even an increase of the index may happen. In [98] some circuit configurations are reviewed which may yield higher-index problems. This will be illustrated in the following with some extensions of our basic ringoscillator model. Hereby it is sufficient for our purpose to modify only the circuit frame, while the core symbolized by a dashed box (see Fig. 10) remains unchanged.

Inductance of interconnect. Since power supply and ground line conduct a significant and rapidly changing current, it may be necessary to take their inductance into account (see Table 4). For the sake of simplicity we insert only an inductor with inductance L into the ground line of Fig. 10. The inclusion of an inductor into the power supply line gives no further insight here.

The differential index is raised from one to two, since the circuit now contains a cutset of an inductor and current sources. All node voltages in the cutset depend on the first derivatives of $J_1(t), \ldots, J_3(t)$, i. e. are index-2 variables. Numerically, this may not cause problems as far as the sources $J_i(t)$ are smooth. However it becomes apparent if the $J_i(t)$ are slightly perturbed with a 'noisy' signal of small amplitude and high frequency .

Realistic model for current sources. The sources providing the current for the differential stages in our basic ringoscillator model of Fig. 10 are in practice realized by bipolar transistors of npn-type, which are biased with a positive base-emitter voltage and a negative base-collector voltage. In this case the collector current is approximately given by

$$I_C \approx \beta \cdot (e^{\frac{U_{BE}}{U_T}} - 1)$$

Effect	Important for	Example	Impact on index
Non ideal element characteristics	high performance, analog circuits	limited output conductance of transistors	eventually decreasing
Resistance of diffusions	standard designs	emitter resistance of bipolar transistors	eventually decreasing
of interconnects	long interconnects, high currents	resistance of power supply, via holes	eventually decreasing
Capacitance of diffusions	standard designs	capacitive load	usually no change
of interconnects	large interconnects	signal cross-coupling	usually no change
Inductance of interconnects, package, etc.	high currents, fast switching	inductance of power supply	eventually increasing
Temperature effects	large temperature range	temperature dependence of mobility	no change
	high power	self-heating of power transistors	eventually increasing
Distributed (non-compact) elements	very fast switching signals	delay time of transmission lines	eventually increasing
	charge sensitive circuits	non quasistationary element equations	usually no change
Parasitic semiconductor devices	compact design rules, unusual operating conditions	bipolar latchup in CMOS circuits	usually no change
External electromagnetic noise, radiation	flux or charge sensitive designs	α-radiation in dynamic memory cells	eventually increasing

Table 4: Important parasitic effects in integrated circuit designs

(see Sec. 1), which defines $v_{Bias} = U_{BE}$ in order to get the same value for I_C as was provided by the current sources $J_1 \cdots J_3$ in the basic model.

Formally, the cutset of current sources and inductors is broken, and the index is reduced to 1. Numerically however, the bipolar transistors still act as current sources, and so one has to deal with a singularly perturbed index-2 problem if the regularizing capacitances c_{ij} at the three transistors acting as real current sources are small.

Modelling of crosstalk. If the interconnects are long parallel wires in the layout, then it may become necessary to take crosstalk between them into account (see Table 4). We restrict here to the simple case of crosstalk between the interconnect nodes 9 and 10 in our circuit of Fig. 10. Usually, crosstalk is modeled by adding a coupling capacitor between the nodes. However, sometimes also controlled sources are used for this purpose, especially in higher order models. We will focus here on the latter model since it may have a negative impact on the index, while the first one has a regularizing effect.

Node 10 is split into a pair 10 and $10a$ which are connected by a voltage-controlled voltage source E_{Cross}. The controlling branch voltage is the voltage drop between nodes 9 and 10:

$$E_{Cross} = u_{10a} - u_{10} = \alpha_E \cdot (u_9 - u_{10}) \,.$$

A reasonable value for the crosstalk factor α_E is between 1% and 10%. Note that here the mutual crosstalk from node 10 to node 9 is one order of magnitude smaller and can be neglected. Now the power supply voltage of node 10 is no longer constant. In our case, this will not have an effect on the oscillating waveforms, since the current provided by the sources J_1, J_2, J_3 is independent of u_{10}. But now the parasitic capacitor c_{10} of node 10 versus ground has to be included, since it is de- and upcharged simultaneously and such causes an additional load for the power supply current J_{SS}.

With J_L and J_E being the currents through the inductor and the controlled voltage source E_{Cross}, respectively, a first order approximation yields

$$\begin{aligned} J_{SS} = -J_E &= c_{10} \cdot \dot{u}_{10} - J_1 - J_2 - J_3 \\ &\approx \Delta J_{SS} - J_1 - J_2 - J_3, \end{aligned}$$

where

$$\Delta J_{SS} = -\alpha_E c_{10} \dot{u}_9$$

is caused by the crosstalk effect. The relative additional current

$$\left| \frac{\Delta J_{SS}}{J_1 + J_2 + J_3} \right|$$

is not very significant for smooth current sources, but it may increase dramatically if the current sources J_1, J_2, J_3 are somewhat noisy. The reason is, that u_9 is of index 2 due to the cutset of current sources/inductor. Since this variable controls the voltage source E_{Cross}, which is enclosed in a loop of voltage sources/capacitor anyway, its current J_E and therefore also the current J_{SS} of the power supply source V_{SS} are of index 3 [98]. So J_{SS} depends on the second derivatives of the current sources $J_1(t), J_2(t), J_3(t)$.

Note that for the latter model the numerical integrations schemes in standard simulation packages will fail in general if a noisy signal is applied to the input sources. Not even the startup behaviour of this circuit, where the power supply and input signals are ramped up to their final value, can be analysed in general due to the nonsmooth form of the ramp-up signals.

A detailed discussion of the bipolar ringoscillator and its refinement levels, including all technical parameters and models, derivation of network equations, and waveforms, can be found in [97].

After setup and analysis of the DAE network equations modelling electrical circuits in time domain, it remains to discuss the third step in circuit simulation: Numerical integration using DAE discretization schemes, which are tailored to the structure and index of the network equations.

Chapter III

Numerical Integration Schemes

The numerical integration of the network equations defines (at least from a mathematical point of view) the kernel of simulation packages in circuit design. This chapter does not aim at an introduction into numerical integration schemes for DAE systems: Neither in theory (convergence and stability) nor in general aspects of implementation (adaptivity, solution of nonlinear and linear systems). For this, the reader may consult a bunch of excellent textbooks [7, 26, 111] or the survey article [190].

In the following we first describe the conventional approach based on implicit linear multi-step methods, discuss the basic algorithms used, and how they are implemented and tailored to the needs of circuit simulation. Special care is demanded of index-2 systems. In addition, we introduce an alternative approach based on one-step methods. This recently developed scheme is compatible to the conventional one with respect to efficiency and robustness, and shows interesting numerical damping properties.

Throughout this chapter we will assume that the network equations correspond to RLC networks, and the only allowed controlled sources are those which keep the index between 1 and 2, depending on the network structure.

10 The conventional approach: Implicit linear multi-step formulas

To simplify notation, we first rewrite the network equations (4.1) in charge/flux oriented formulation

$$
\begin{aligned}
0 &= \underbrace{\begin{pmatrix} A_C & 0 \\ 0 & I \\ 0 & 0 \end{pmatrix}}_{A :=} \cdot \underbrace{\begin{pmatrix} \dot{q} \\ \dot{\phi} \end{pmatrix}}_{\dot{y} :=} + \underbrace{\begin{pmatrix} A_R r(A_R^\top u, t) + A_L \jmath_L + A_V \jmath_V + A_I \imath(u, \jmath_L, \jmath_V, t) \\ -A_L^\top u \\ v(u, \jmath_L, \jmath_V, t) - A_V^\top u \end{pmatrix}}_{f(x,t) :=}, \\
\underbrace{\begin{pmatrix} q \\ \phi \end{pmatrix}}_{y :=} &= \underbrace{\begin{pmatrix} q_C(A_C^\top u) \\ \phi_L(\jmath_L) \end{pmatrix}}_{g(x,t) :=}
\end{aligned}
$$

in a more compact linear-implicit form:

$$
\begin{aligned}
0 &= \mathcal{F}(\dot{y}(t), x(t), t) := A \cdot \dot{y}(t) + f(x(t), t), && \text{(10.1a)} \\
0 &= y(t) - g(x(t)) && \text{(10.1b)}
\end{aligned}
$$

with $x := (u, \jmath_L, \jmath_V)^\top$ being the vector of unknown network variables.

The basic algorithm. The conventional approach can be split into three main steps: Computation of consistent initial values, numerical integration of $\dot{y}$ based on multi-step schemes, transformation of the DAE into a nonlinear system and its numerical solution by Newton's procedure. Since the third step is usually performed with methods which are not very specific for circuit simulation, we will not discuss it further here.

Let us assume for the moment that the network equations are of index 1 — the index-2 case will be discussed later.

Consistent initial values. The first step in the transient analysis is to compute consistent initial values (x_0, y_0) for the initial time point t_0. In the index-1 case, this can be done by performing a steady state (DC operating point) analysis, i.e. to solve

$$\mathcal{F}(0, x_0, t_0) = 0 \tag{10.2}$$

for x_0 and then set $y_0 := g(x_0)$. If there are no controlled sources, the Jacobian $\partial\mathcal{F}/\partial x$ of (10.2) with respect to x_0 reads

$$\frac{\partial \mathcal{F}}{\partial x} = \begin{pmatrix} \widetilde{G}(A_R^\top u_0, t_0) & A_L & A_V \\ -A_L^\top & 0 & 0 \\ -A_V^\top & 0 & 0 \end{pmatrix}$$

with the definition $\widetilde{G}(A_R^\top u, t) := A_R G(A_R^\top u, t) A_R^\top$ already introduced in Section 4. Since $\ker\,(\partial\mathcal{F}/\partial x) = \ker\,(A_R, A_L, A_V)^\top \times \ker\,(A_L, A_V)$ holds, the matrix is only regular, if there are neither loops of independent voltage sources and/or inductors, nor cutsets of independent current sources and/or capacitors. If these topological conditions are violated, no steady state solution can be computed, and so most circuit analysis programs check and refuse these circuit configurations. Additional assumptions are implied in the case of controlled sources. But note that in the nonlinear case the Jacobian matrix also may become numerically singular, e. g. due to vanishing partial derivatives or in the case of bifurcation.

An approach always feasible in the index-1 case is to extract the algebraic constraints using the projector Q_C onto $\ker A_C^\top$:

$$\begin{aligned} Q_C^\top(A_R r(A_R^\top u, t) + A_L \jmath_L + A_V \jmath_V + A_I \imath(u, \jmath_L, \jmath_V, t)) &= 0 \\ v(u, \jmath_L, \jmath_V, t) - A_V^\top u &= 0. \end{aligned}$$

If the index-1 topological conditions hold, this nonlinear system uniquely defines for $t = t_0$ the algebraic components $Q_C u_0$ and $\jmath_{V,0}$ for given (arbitrary) differential components $(I - Q_C)u_0$ and $\jmath_{L,0}$. The derivatives $\dot{y}_0$ have then to be chosen such that $A\dot{y}_0 + f(x_0, t_0) = 0$ holds.

Numerical integration. Starting from consistent initial values, the solution of the network equations is computed at discrete time points $t_1, t_2, \ldots$, by numerical integration with implicit linear multi-step formulas.

The direct approach, which is shortly described here, was first proposed by Gear [81] for *backward differentiation formulas* (BDF methods): For a timestep h_k from t_{k-1} to $t_k = t_{k-1} + h_k$ the derivative $\dot{y}(t_k)$ in (10.1) is replaced by a linear ρ-step operator ρ_k for the approximate $\dot{y}_k$, which is defined by

$$\rho_k = \frac{1}{h_k} \sum_{i=0}^{\rho} \gamma_{k,i} y_{k-i} - \sum_{i=1}^{\rho} \beta_{k,i} \dot{y}_{k-i} := \alpha_k y_k + r_k \tag{10.3}$$

with $y_{k-i} := g(x_{k-i})$, $i = 0, 1, \ldots, \rho$ and $\dot{y}_{k-i}$, $i = 1, \ldots, \rho$, already computed by previous operators ρ_{k-i}. The index k in the method coefficients $\beta_{k,i}$ and $\gamma_{k,i}$ indicate their dependence on the step size history in the case of variable step size implementations (see the paragraph about adaptivity below). The remainder r_k contains values of y and $\dot{y}$ for previous time points.

Transformation into a nonlinear system of equations. The numerical solution of the DAE system (10.1) is thus reduced to the solution of a system of nonlinear equations

$$\mathcal{F}(\alpha_k g(x_k) + r_k, x_k, t_k) = 0, \tag{10.4}$$

which is solved iteratively for x_k by applying Newtons's method in a predictor-corrector scheme. Starting with a predictor step $x_k^{(0)}$ (x_{k-1} or some kind of extrapolated value from previous timepoint may be a reasonable choice), a new Newton correction $\Delta x_k^{(l)} := x_k^{(l)} - x_k^{(l-1)}$ is computed from a system of linear equations

$$D\mathcal{F}^{(l-1)}\Delta x_k^{(l)} = -\mathcal{F}^{(l-1)}, \quad \mathcal{F}^{(l-1)} := \mathcal{F}(\alpha_k g(x_k^{(l-1)}) + r_k, x_k^{(l-1)}, t_k) \tag{10.5}$$

directly by sparse LU decomposition and forward backward substitution. Due to the structure of the nonlinear equations the Jacobian $D\mathcal{F}^{(l-1)}$ for Newton's scheme is

$$D\mathcal{F}^{(l-1)} = \alpha_k \cdot \mathcal{F}_{\dot{x}}^{(l-1)} + \mathcal{F}_x^{(l-1)} \text{ with } \mathcal{F}_{\dot{x}}^{(l-1)} = A \cdot \frac{\partial g(x_k^{(l-1)})}{\partial x}, \quad \mathcal{F}_x^{(l-1)} = \frac{\partial f(x_k^{(l-1)}, t_k)}{\partial x}.$$

If the step size h is sufficiently small, the regularity of $D\mathcal{F}^{(l-1)}$ follows from the regularity of the matrix pencil $\{A \cdot \partial g(x)/\partial x, \partial f/\partial x\}$ that is given at least for index-1 systems.

Implementation: Element stamps and cheap Jacobian. The implementation of the direct approach for one timestep into the analysis kernel of circuit simulation packages such as SPICE is outlined in Fig. 11.

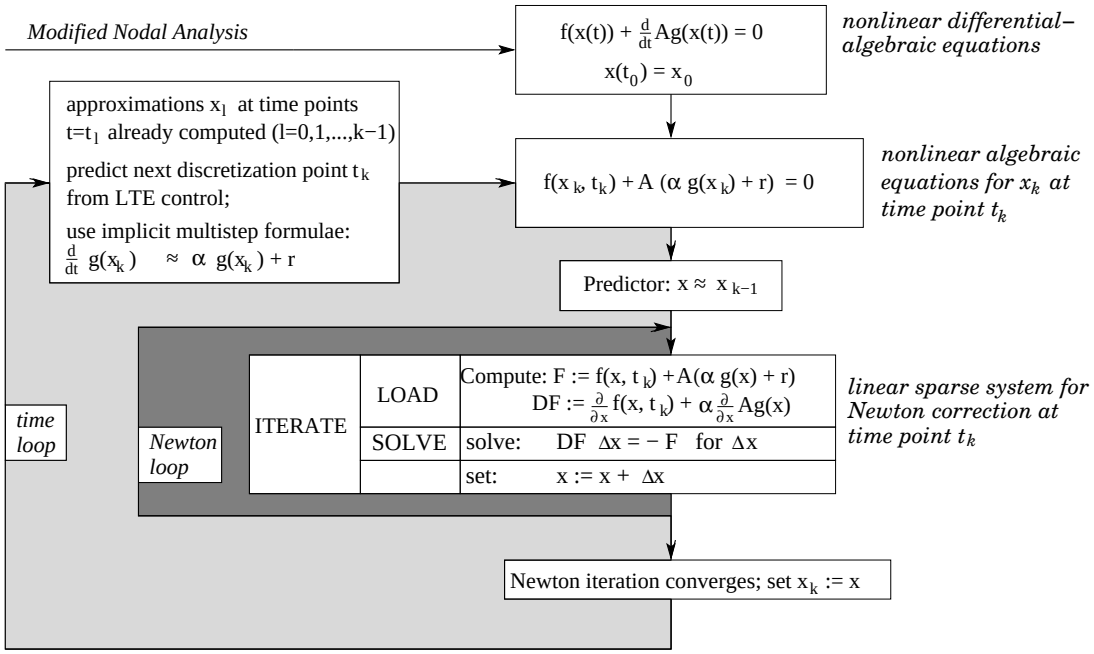


Figure 11: The direct approach in SPICE like simulators

In every Newton step (10.5), two main steps have to be performed:

- `LOAD` part: First the right-hand side $-\mathcal{F}^{(l-1)}$ of (10.5) and the Jacobian $D\mathcal{F}^{(l-1)}$ have to be computed;
- `SOLVE` part: The arising linear system is solved directly by sparse LU decomposition and forward backward substitution.

A characteristic feature of the implementation is that modelling and numerical integration (10.3) are interwoven in the `LOAD` part: First the arrays for right-hand side and Jacobian are zeroed. In a second step, these arrays are assembled by adding the contributions to $\mathcal{F}$ and $D\mathcal{F}$ element by element: So-called *element stamps* are used to evaluate the time-discretized models for network elements.

Let us consider, for example, a linear capacitor with capacitance C between the nodes $''+''$ and $''-''$ with node potentials u_+ and u_- at time point t_k. Its characteristic equation reads

$I_C(t_k) = \dot{q}_C(t_k)$, $q_C(t_k) = C \cdot (u_+ - u_-)$. After incorporating the approximation (10.3) for $\dot{q}_C$ one gets the approximate element relation

$$I_C = \alpha_k C \cdot (u_+ - u_-) + r_k,$$

which gives the following contributions to the Jacobian matrix for the rows corresponding to nodes $''+''$ and $''-''$ and columns corresponding to node potentials u_+ and u_-, and to the right-hand side (rhs) at nodes $''+''$ and $''-''$:

	u_+	u_-	rhs
$+$	$\alpha_k C$	$-\alpha_k C$	$-I_C$
$-$	$-\alpha_k C$	$\alpha_k C$	I_C

One consequence of using element stamps is the cheap availability of the Jacobian: For highly integrated circuits with a very sparse Jacobian, it is only slightly more expensive to evaluate both right-hand side and Jacobian than to evaluate only the right-hand side by its own. So, if not linear algebra aspects are dominant (which may happen for very large circuits) then the use of full rather than modified Newton may be appropriate in many cases.

BDF schemes and trapezoidal rule. It remains to answer the question which types of implicit linear multi-step formulae (10.3) are actually used. Since SPICE2 [170], most circuit simulators solve the network equations either with the *trapezoidal rule* (TR)

$$\rho_k = -\dot{y}_{k-1} + \frac{2}{h}(y_k - y_{k-1}) \qquad (\rho = 1, \beta_{k,1} = 1, \gamma_{k,0} = -\gamma_{k,1} = 2) \tag{10.6}$$

or with BDF schemes:

$$\rho_k = \frac{1}{h_k} \sum_{i=0}^{\rho} \gamma_{k,i} y_{k-i}, \qquad (\beta_{k,1} = \ldots \beta_{k,\rho} = 0) \tag{10.7}$$

For the BDF methods no derivatives of y at previous time points are needed. The first timestep is always performed by BDF1 (implicit Euler scheme) as starting procedure.

Why BDF schemes? The most appealing argument is to save function evaluations as much as possible, since they are extremely expensive in circuit simulation — see Gear's article [81] which was explicitly dedicated for solving circuit equations, and consequently had been published in an electrical engineering journal. A second one is that the use of higher order methods does not require much extra cost. And the third one is a settled convergence and stability theory for fully-implicit (and not only semi-explicit) index-1 systems. Nonlinear index-1 network equations fit into this class of problems. For such systems the following convergence result for BDF schemes can be found in any textbook on DAEs: The ρ-step BDF method of fixed size h for $\rho < 7$ is feasible and converges to $\mathcal{O}(h^\rho)$ if all initial values are correct to $\mathcal{O}(h^\rho)$ and if the Newton process at each timestep is solved to accuracy $\mathcal{O}(h^{\rho+1})$. This convergence result has also been extended to variable stepsize BDF methods, provided that they are implemented in such a way that the method is stable for standard ODEs, i.e. the ratio of two succeeding stepsizes is bounded.

It should be noted that BDF schemes with order greater 3 are rarely used in practice because of the low smoothness properties of the transistor model equations. Stability properties give an additional argument for BDF1 and BDF2 schemes anyway: They are *A-stable*, i.e. for Dahlquist's linear test equation $\dot{x} = \lambda x$ the numerical solution with arbitrary stepsize h is bounded for all $\{\lambda; \mathrm{Re}(\lambda) < 0\}$ in the left half plane $\mathbb{C}^-$. In other words, no stability problems occur for stiff systems with decaying solutions. In contrast, convergent BDF schemes with higher order ($3 \le \rho \le 6$) cannot be A-stable because of the second Dahlquist barrier. However, they are

$A(\alpha)$-stable with $0 < \alpha < \pi/2$, i.e. stable in the sectorial $\{\lambda; |\arg(-\lambda)| < \alpha, \lambda \neq 0\}$ of the left half plane; and at least for BDF3 $\alpha \approx 86 \cdot \frac{\pi}{180}$ is large enough to yield no serious stability problems in practice.

In addition to A-stability — and $A(\alpha)$-stability, respectively — the numerical solutions of BDF tend to zero (for fixed stepsize h) in the very stiff limit $\mathrm{Re}(\lambda) \to -\infty$. This *stiff decay* property (which is equivalent to the L-stability property for one-step methods) allows to skip rapidly varying solution details and still maintain a decent description of the solution on a coarse level in the very stiff case. Hence they are suitable for network equations that are generally very stiff because of the widely separated time constants in electrical circuits.

One consequence for A stable methods with stiff decay is numerical damping along the imaginary axis. This behaviour defines a serious shortcoming for BDF1 and BDF2 schemes: The solution is damped so strongly, that even for rather small timesteps oscillations may be damped out, and after some cycles a circuit seems to be quiescent even though it oscillates in reality.

A natural alternative to BDF2 is the trapezoidal rule TR, since it is the A-stable linear multi-step method of order 2 with smallest leading error coefficient. Due to its energy conserving property, it avoids the shortcoming of BDF methods: Oscillations are not damped at all — unfortunately, not even instabilities of highest frequency caused by numerical noise. This weak instability can be seen directly from (10.6): Errors of $\dot{y}_{k-1}$ are propagated to $\dot{y}_k = \rho_k$ without being damped, and errors of $\dot{y}_k$ propagate directly to the respective components of x_k.

One conclusion might be that TR would be a desirable integration rule, if it were damped sufficiently, but not as strongly as BDF. For this purpose several approaches are described to construct a combination of TR and BDF schemes [70], so-called TR-BDF schemes. This name was first used in a paper by Bank et al. [9]. The aim is to combine the advantages of both methods: Large timesteps and no loss of energy of the trapezoidal rule (TR) combined with the damping properties of BDF. An interesting interpretation of TR-BDF as a one-step method was presented in [120].

When looking for alternatives to TR-BDF, we will return in Section 12 for a more detailed discussion to the problem of preserving physical oscillations, while damping out artificial ones very efficiently.

Adaptivity: Stepsize selection and error control. Variable integration stepsizes are mandatory in circuit simulation since activity varies strongly over time. A simple criterion for timestep control can be obtained from the Newton process itself: The stepsize is reduced/increased, if the number of Newton iterations per timestep is larger/smaller than a given threshold (for example, 8 and 3); otherwise, the stepsize remains unchanged. This criterion is cheap to compute, but not very reliable: Linear problems converge with one single Newton step and hence would always be integrated with maximal stepsize.

The conventional strategy: Estimating the local truncation error in $\dot{\boldsymbol{y}}$. A more reliable and still efficient stepsize prediction is based on estimating the local truncation error $\varepsilon_{\dot{y}} = \dot{y}(t_{k+1}) - \rho_{k+1}$ of the next step to be performed, i.e. the residual of the implicit linear multi-step formulas if the exact solution is inserted ($\beta_{k+1,0} = 1$):

$$\varepsilon_{\dot{y}} \;:=\; \sum_{i=0}^{\rho} \beta_{k+1,i} \dot{y}_{(t_{k+1-i})} - \frac{1}{h_{k+1}} \sum_{i=0}^{\rho} \gamma_{k+1,i} g(x(t_{k+1-i}))\,.$$

Usually the accuracy of $\dot{y}$ is controlled rather than that of y, because y itself is no quantity of interest for the user. This implies the loss of one integration order, as we will see now. After Taylor expansion around t_k the leading error term in $\varepsilon_{\dot{y}}$ turns out to be

$$\varepsilon_{\dot{y}} \;\approx\; \begin{cases} \dfrac{1}{2} h_{k+1} \dfrac{\mathrm{d}^2}{\mathrm{dt}^2} g(x(t_k)) & \text{for BDF1} \\ \dfrac{1}{6} h_{k+1}(h_{k+1} + h_k) \dfrac{\mathrm{d}^3}{\mathrm{dt}^3} g(x(t_k)) & \text{for BDF2} \\ \dfrac{1}{6} h_{k+1}^2 \dfrac{\mathrm{d}^3}{\mathrm{dt}^3} g(x(t_k)) & \text{for TR} \end{cases}$$

The higher order time derivatives of g are usually estimated via divided differences based on $y_k, \ldots, y_{k-1-\rho}$. This is rather inaccurate, since only backward information is used to get the derivatives at the actual timepoint. So timestep control is always somewhat "behind" the actual timepoint, which makes it unstable and gives rise to overreactions, especially when the timesteps are large. This is another argument — besides that of low order smoothness of the element models — why BDF schemes of order greater than 3 are seldom used in practice. To improve the estimates for the higher order derivatives of g, it was suggested in [138] to replace the divided differences by a higher order scheme — e. g. the trapezoidal rule — and to employ $\rho_k, \ldots \rho_{k-p}$ for its evaluation. This improves accuracy, needs less backward stages, and is surely more consistent since the time derivatives entering the solution are either used for timestep control, and not any further approximations of them.

A new stepsize can be predicted by matching $\varepsilon_{\dot{y}}$ with a user defined error tolerance `TOL`. If h_{k+1} is not different from h_k, then TR allows due to its smaller error constant a timestep which is approximately 40% larger than for BDF2.

For an a-posteriori error check, the inequality $||\varepsilon_{\dot{y}}|| \leq$ `TOL` has to be evaluated with updated function evaluations for the higher order derivatives. Furthermore, an order control for variable order BDF schemes can be constructed very easily: The stepsize predictions for order $\rho-1, \rho$ and $\rho+1$ are computed, and that order is chosen which gives the maximal timestep. In practice, the difference between the converged solution at $t = t_k$ and the initial value provided by a suitable predictor polynomial is a key value for the local truncation error estimation.

Modified timestep control. The main flaw of controlling $\varepsilon_{\dot{y}}$ is that the user has no direct control on the really interesting circuit variables, i.e. node potentials u and branch currents $\jmath_L, \jmath_V$. In order to overcome this disadvantage associated with charge/flux oriented integration, Denk [50] used

$$\dot{g}(x(t))) = \frac{\partial g(x(t))}{\partial x}\dot{x}(t),$$

which means to assemble the terminal charges/branch fluxes in circuit nodes/branches and to perform classical integration on x rather than y. This method works well if Newton's procedure is started from a low order predictor. However, it requires the computation of the second derivatives of g, which are hard to get in practice or do not even exist due to poor smoothness properties of transistor models.

An alternative approach [221] is based on the idea not to transfrom the whole network equations as done by Denk, but only the local truncation error $\varepsilon_{\dot{y}}$ for $\dot{q}$ into a (cheap) estimate for the local error $\varepsilon_x := x(t_k) - x_k$ of $x(t)$. By expanding $\mathcal{F}(\dot{y}(t), x(t), t)$ at the actual time point t_k into a Taylor series around the approximate solution $(\dot{y}_k, x_k)$ and neglecting higher order terms, one obtains

$$\mathcal{F}(\dot{y}(t_k), x(t_k), t_k) \approx \mathcal{F}(\dot{y}_k, x_k, t_k) + \frac{\partial \mathcal{F}}{\partial \dot{y}}(\dot{y}(t) - \dot{y}_k) + \frac{\partial \mathcal{F}}{\partial x}(x(t) - x_k).$$

With the difference of exact and approximate value for $\dot{y}(t_k)$

$$\dot{g}(x(t_k)) - \dot{g}(x_k) = \alpha_k(g(x(t_k)) - g(x_k)) + \varepsilon_{\dot{y}} \approx \alpha_k \frac{\partial g}{\partial x}(x(t_k) - x_k) + \varepsilon_{\dot{y}}$$

follows:

$$\mathcal{F}(\dot{y}(t_k), x(t_k), t_k) \approx \mathcal{F}(\dot{y}_k, x_k, t_k) + \frac{\partial \mathcal{F}}{\partial \dot{y}}\varepsilon_{\dot{y}} + \left(\alpha_k \frac{\partial \mathcal{F}}{\partial \dot{y}}\frac{\partial g}{\partial x} + \frac{\partial \mathcal{F}}{\partial x}\right)\varepsilon_x.$$

As $\mathcal{F}$ is zero for both the exact and the approximate solution, the desired error estimate ε_x for $x(t_k)$ can be computed from the linear system

$$\left(\alpha_k A \frac{\partial g}{\partial x} + \frac{\partial f}{\partial x}\right)\varepsilon_x = -A\varepsilon_{\dot{y}} \tag{10.8}$$

of which the coefficient matrix is the Jacobian of Newton's procedure! Since the local error ε_x can be interpreted as a linear perturbation of $x(t_k)$, if $\mathcal{F}$ is perturbed with the local truncation error $\varepsilon_{\dot{y}}$, the choice of ε_x is justified as an error estimate for numerical integration. The idea

to weight the local truncation error via Newton's method was already proposed by Sacks-Davis [207] for stiff ordinary differential equations and by Gupta et al. [105] and Leimkuhler [152] for nonlinear DAEs of index 2. They key motivation pursued in the literature was to damp the impact of the stiff components on timestep control — which otherwise would yield very small timesteps. While this aspect can be found in the textbooks, a second aspects comes from the framework of charge oriented circuit simulation: Newton's matrix brings system behaviour into account of timestep control, such mapping integration errors of single variables onto those network variables, which are of particular interest for the user.

Because of $\alpha_k = \mathcal{O}(h^{-1})$, the first term in Newton's iteration matrix may become dominant if the timestep is sufficiently small. Hence for high accuracy requirements — which force the timesteps to be small — we can expect to get back one order of accuracy, which was lost by directly controlling the truncation error $\varepsilon_{\dot{y}}$. However, the a-posteriori test is more rigorous than with the conventional strategy because of the inclusion of an updated iteration matrix. This leads in principle to a loss in robustness since more timesteps are likely to be refused. In such a situation more conservative a-priori timesteps should be chosen, but overall this may degrade efficiency either.

Timestep control as an optimal control problem. How can we determine an optimal compromise between large a-priori timesteps and only a few number of a-posteriori refused timesteps? An interesting approach pursued by Söderlind et al. [106] is to look at this problem from the viewpoint of control theory, and to build a linear PI-controller for this purpose: Its P-term is proportional to the difference between the desired tolerance `TOL` and the actual a-posteriori error, and its I-term integrates (sums up) the past values of these values. An increase/decrease of them gives rise to a more conservative/relaxed a-priori choice of timesteps. This approach has for the first time opened timestep control to a rigorous mathematical analysis, and consequently has found entrance into the textbooks [54, 111]. An actual survey is given in [227]. Since practical experience shows that it is difficult to find a fixed set of parameters for the controller, which applies well to all circuit simulation problems [3], it was suggested to employ adaptive control mechanisms for this purpose [162]. Although looking very interesting and promising, this kind of timestep control still needs improvements in details, which would make it practical for standard applications in an industrial environment. One reasonable extension might be to include the number of Newton iterations per timestep into the controller ([3]).

Note. In practice, in some circuit simulators attention to the local discretization error is restricted to the voltage unknowns in x [141].

The index-2 case. Since most applications of practical interest yield network equations of index 2, numerical integration must be enabled to cope with this kind of problems. As they are not of Hessenberg type, it is not a-priori clear whether the BDF approach can be generalized to such problems. Fortunately, the fine structure of the network equations derived in Chapter II helps to answer this question. It turns out that the BDF can be used to solve such systems, provided that consistent initial values are available, a weak instability associated with an index-2 non-Hessenberg system is fixed, and some problems with timestep control are solved.
The latter item was already mentioned before: It can be solved by using Newton's iteration matrix for weighting the local truncation error $\varepsilon_{\dot{y}}$, thus getting ε_x for timestep control, see equation (10.8). The first items can be solved by using information from an index monitor, as will be shown in the following.

An index monitor has following tasks: It determines the index, identifies critical parts of the circuit and invokes special treatment for them in order to avoid failures of the numerical integration, gives hints to the user how to regularize the problem in case of trouble, and which network variables may be given initial values, and which must not. And of course the index monitor must be fast enough to cope with the large size of problems which are standard in industrial applications.

Such an index monitor has been developed by Estévez Schwarz and Tischendorf [67, 237] [4] and

[4] Based on the *generic index* concept, alternative algorithms were suggested in [191, 192]. The same approach can also be used to smooth results when restarting from discontinuities, which otherwise show some intial wiggles.

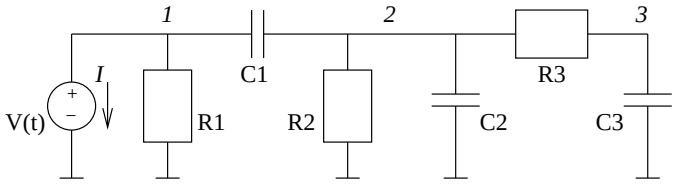


Figure 12: An index-2 circuit

successfully implemented into an industrial circuit simulator [68]. It aims at characterizing a charge oriented network model in MNA formulation to be of index $0, 1, 2$, or possibly larger than 2. This diagnosis tool consists of a graph oriented part, which checks topological criteria about position and — in case of networks with controlled sources – control of the network elements, and of a numerical part, which checks positive definiteness of element relations during analysis. The combination of *topological* and *local numerical* checks makes the monitor very efficient: A 30000 transistor circuit can be handled in a few seconds. In case of circuit configurations which may yield an index > 2, the critical circuit parts are identified, and suggestions for regularization are issued.

One essential outcome of this work is that industry has learned how to construct future device and circuit models in order to avoid numerical problems due to high DAE index as far as possible.

Computing consistent initial values. The usual way in circuit simulation to compute initial values by solving a DC steady state problem (10.2) may yield inconsistent initial values in the index-2 case, since the hidden constraints — relating parts of the solution to the time derivatives of the time dependent elements — are not observed. A simple example is the VC-loop of Fig. 5, where the current $\jmath_V$ depends on the time derivative $\dot{v}(t)$ of the input signal. This raises two questions for index-2 problems:

- How can we get consistent initial values?
- What happens when integration is started from nonconsistent initial values?

The standard method for the first problem consists of three steps ([65, 182]):

1. Select variables which can be given initial values, and initialize them;
2. setup equations for hidden constraints;
3. solve an augmented *nonlinear* system which includes the hidden constraints.

In [63] it was shown that the first and the second step can be done efficiently in circuit simulation by using the results of the previously described index monitor. However, a problem with this approach is that it is very much different from the handling of initial conditions in the lower index case. So an alternative was developed in [66], which aims at being as near as possible to the standard algorithm for low index:

1. Find a solution without hidden constraints from solving equation (10.2);
2. setup and solve a *linear* system for corrections to this solution, such that the hidden constraints are fulfilled;
3. add the corrections to the initial values found in step 1 to get consistent ones.

Again the hidden constraints can be easily derived from the information provided by the index monitor. When the algorithm is applicable then the variables to be corrected turn out to be branch currents in VC-loops and node voltages in LI-cutsets; details can be found in [66].

As an example we look at the circuit given in Fig. 12. It contains a VC-loop and is of index 2. The unknowns are the node voltages and the branch current of the voltage source, if an MNA

formulation is used: $x = (u_1, u_2, u_3, \jmath_V)^\top$. The network equations are given by:

$$\begin{aligned}
\text{KCL1:} \qquad & \jmath_V + C_1 \cdot (\dot{u}_1 - \dot{u}_2) + \frac{1}{R_1} u_1 = 0 \\
\text{KCL2:} \qquad & C_1 \cdot (\dot{u}_2 - \dot{u}_1) + \frac{1}{R_2} u_2 + C_2 \dot{u}_2 + \frac{1}{R_3} \cdot (u_2 - u_3) = 0 \\
\text{KCL3:} \qquad & \frac{1}{R_3} \cdot (u_3 - u_2) + C_3 \dot{u}_3 = 0 \\
\text{V-Source:} \qquad & u_1 = V(t)
\end{aligned}$$

The steady state DC solution

$$\begin{aligned}
\dot{u}_1 &= 0 \qquad \dot{u}_2 = 0 \qquad \dot{u}_3 = 0 \\
u_1 &= V(0) \qquad u_2 = 0 \qquad u_3 = 0 \\
\jmath_V &= -\frac{1}{R_1} \cdot V(0)
\end{aligned}$$

solves the network equations, but violates the hidden constraint

$$\dot{u}_1 = \dot{V}(t).$$

To make it consistent, we need an additional current $\triangle\jmath_V$ in the VC-loop, which we can compute from:

$$\begin{aligned}
\text{KCL1:} \qquad & \triangle\jmath_V + C_1 \cdot (\dot{u}_1 - \dot{u}_2) = 0 \\
\text{KCL2:} \qquad & C_1 \cdot (\dot{u}_2 - \dot{u}_1) + C_2 \dot{u}_2 = 0 \\
\text{V-Source:} \qquad & \dot{u}_1 = \dot{V}(0)
\end{aligned}$$

(Node 3 is not part of the VC-loop, and can be omitted here.) Its solution is added to the previous one to get the following consistent initial values:

$$\begin{aligned}
\dot{u}_1 &= \dot{V}(0) \qquad \dot{u}_2 = \frac{C_1}{C_1 + C_2} \cdot \dot{V}(0) \qquad \dot{u}_3 = 0 \\
u_1 &= V(0) \qquad u_2 = 0 \qquad u_3 = 0 \\
\jmath_V &= -\frac{1}{R_1} \cdot V(0) - \frac{C_1 \cdot C_2}{C_1 + C_2} \cdot \dot{V}(0)
\end{aligned}$$

In case of a charge/flux oriented formulation the procedure is similar.

To answer the second question, we note that transient analysis may abort or yield wrong results if it is started from an inconsistent inital value. An example is given in [66]. Fortunately, the multi-step methods are mostly started with a backward Euler step, and thanks to the special structure of the network equations this is sufficient in many cases to bring the solution back onto the right manifold, although integration was started from an inconsistent value [66] [5]. Note however that this is not true if integration is started with the trapezoidal rule; even with stiffly-accurate one-step methods it may take some timesteps to get back to the correct solution, if the initial values are not consistent.

Fixing the weak instability. The variable order, variable stepsize BDF for the index-2 network equations (10.1) reads

$$A \frac{1}{h_k} \sum_{i=0}^{\rho} \gamma_{k,i} g(x_{k-i}) + f(x_{k-i}, t_{k-i}) = \delta_k \,.$$

Here, the defect δ_k represents the perturbations in the kth step caused by the rounding errors and the defects arising when solving the nonlinear equations numerically. März and Tischendorf [159] have shown that if the ratio of two succeeding stepsizes is bounded and the defect δ_k is small enough, then the BDF approach is feasible — i.e. the nonlinear equations to be solved

[5] Some authors exploit this feature to get consistent initial values by performing some Eulersteps backward and then again forward in time [25, 242].

per integration step are locally uniquely solvable with Newton's method — and convergent. However, a weakly instable term of the type

$$\max_{k\geq 0} \frac{1}{h_k} \|\mathcal{D}_k \delta_k\|$$

arises on the right-hand side for the error estimate of $\max_{k\geq\rho} \|x_k - x(t_k)\|$. Here $\mathcal{D}_k$ denotes a projector that filters out the the higher-index components of the defect. In contrast to Hessenberg-type index-2 systems, this instability may affect all solution components, and may cause trouble for the timestep and error control. Remember, that the stepsize is decreased if the a-posteriori error check fails. For small stepsizes however, the weak instability is reflected by an error growth if the stepsize is decreased — the usual timestep and error control must fail!

Since all solution components may be affected, an appropriate error scaling — as done for Hessenberg systems — is no remedy. However, the instability can be fixed by reducing the most dangerous part of the defect δ_k, that is, those parts belonging to the range of $\mathcal{D}_k$. This defect correction can be done by generalizing the back propagation technique, since the projector can be computed very cheaply by pure graphical means with the use of an index monitor.

We finish this section with some remarks on a new BDF-based approach to integrate the network equations numerically, which shows some potential for the future: Modified Extended BDF.

In 1983, J. Cash proposed the Modified Extended BDF (MEBDF) method, which combines better stability properties and higher order of convergence than BDF, but requires more computations per step [33, 34]. 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 convergence increases with one for most circuits[30]. This implies that for convergence order 3 we normally apply the 3-step BDF method, while with the MEBDF method a 2-step method suffices.

The k-step MEBDF-methods are A-stable [109] for $k \leq 3$, while for BDF this is restricted to the case $k \leq 2$ [33].. Thus these MEBDF-methods 'break' Dahlquist's Law [109] that applies to real multistep methods: we have higher order methods with unconditional stability.

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 [75].

11 A second approach: One-step methods

Up to now, only multi-step methods have been used for the numerical discretization in professional packages. These conventional methods have achieved a high degree of maturity, and have proven to be efficient and very robust in an extremely large variety of applications. Nevertheless there is some motivation to look at alternative schemes also from an industrial point of view:

- The BDF methods are applicable to much more general classes of nonlinear DAEs; can methods be superior, which are definitely constructed for the special linear-implicit nonlinear form (10.1) of the circuit equations?
- In the charge/flux oriented form of conventional codes, timestep control is difficult, since charge/flux tolerances are not of interest for the user, and extra effort is necessary to derive charge/flux tolerances from the desirable user given node voltage or current tolerances. Are there methods with a more natural embedding of timestep control even in charge oriented formulation?
- The fully implicit methods used so far require in each timestep a nonlinear system to be solved. Can semi-implicit methods be employed, which need only linear systems to be solved?

Recently, a class of one-step methods was developed that give a positive answer to the three questions above. They are based on embedded Rosenbrock-Wanner (ROW) schemes, which have been used successfully for solving classical network equations [195], and

- are tailored to the special structure of charge/flux oriented network equations (10.1), and do not aim at solving arbitrary DAEs of non-Hessenberg type;
- enable a natural timestep control which applies directly on node potentials and branch currents;
- define linearly-implicit methods that need only linear systems to be solved.

Since these schemes turned out to be competitive with the standard multi-step methods even in an industrial environment, it seems worthwhile to introduce them in more detail here.

Charge/flux-oriented ROW schemes. In a first step, we apply a standard Rosenbrock-Wanner method to the linear-implicit DAE system (10.1) [110, 194]. To simplify notation, we assume for the moment that the network equations do not explicitly depend on time, i.e. $f(x(t),t) \equiv f(x(t))$. For this homogeneous case, the numerical approximation for one ROW step reads

$$x_1 = x_0 + b^\top k, \tag{11.1a}$$
$$y_1 = y_0 + b^\top l, \tag{11.1b}$$

with weights $b := (b_1, \ldots, b_s)^\top$ and increments $k := (k_1, \ldots, k_s)^\top$, $l := (l_1, \ldots, l_s)^\top$ defined by

$$\begin{pmatrix} A & \gamma h \dfrac{\partial f(x_0)}{\partial x} \\ -\gamma I & \gamma \dfrac{\partial g(x_0)}{\partial x} \end{pmatrix} \cdot \begin{pmatrix} l_i \\ k_i \end{pmatrix} = \begin{pmatrix} -hf(\sum_{j=1}^{i-1} \alpha_{ij} k_j) - h \dfrac{\partial f(x_0)}{\partial x} \sum_{j=1}^{i-1} \gamma_{ij} k_j \\ y_0 - g(\sum_{j=1}^{i-1} \alpha_{ij} k_j) + \sum_{j=1}^{i-1} (\alpha_{ij} + \gamma_{ij}) l_j - \dfrac{\partial g(x_0)}{\partial x} \sum_{j=1}^{i-1} \gamma_{ij} k_j \end{pmatrix} \tag{11.1c}$$

where $\alpha_{ij} = 0$ for $i \geq j$, $\gamma_{ij} = 0$ for $i > j$ and $\gamma_{ii} = \gamma \neq 0$, $i, j = 1, \ldots, s$. x_1 and y_1 are the approximations to the solution at time h with $x(0) = x_0$, $y(0) = y_0$. The increments are uniquely defined by the linear system (11.1c): The matrix

$$\begin{pmatrix} 0 & A \dfrac{\partial g(x_0)}{\partial x} + \gamma h \dfrac{\partial f(x_0)}{\partial x} \\ -\gamma I & \gamma \dfrac{\partial g(x_0)}{\partial x} \end{pmatrix}$$

obtained after one block Gaussian elimination step is nonsingular for sufficient small stepsizes h, since the matrix pencil $\{A\partial g(x)/\partial x, \partial f/\partial x\}$ is regular at least for index-1 systems.

In a second step, we use the special structure of (10.1) to eliminate the differential components y from the computation of x_1. The linear structure of the charge constraint (10.1b) allows for k_i to be computed independently from $l_1, \ldots, l_{i-1}$. To fulfill charge conservation during integration, the differential variables y are projected at each grid point t_i in the integration interval $[0, T]$ on the charge constraint:

$$y_i := g(x_i), \qquad \forall i \text{ with } t_i \in [0, T]. \tag{11.1d}$$

In the end, the computation of x_1 does only depend on x_0, and we have defined a class of charge/flux oriented ROW schemes by (11.1a,11.1c) and (11.1d).

One notes that the same Jacobian information is needed in both multi-step schemes and charge/flux oriented ROW methods, but for different reasons: As iteration matrix for the multi-step schemes in the first case, and as system matrix of the linear equations which serve for getting the stage increments in the latter case. Note that the same Jacobian is used here for all stage equations. It is however possible to construct efficient higher order methods which exploit the fact that in circuit simulation the Jacobian is rather cheap to get [95, 99]; in this case the Jacobian would be different at each stage.

Convergence and order conditions. As shown in [96], classical convergence theory for semi-explicit index-1 problems can be applied to the ROW method (11.1a,11.1c,11.1d). Owing to the projection (11.1d), the local error $g(x_1)-g(x(h))$ must be $\mathcal{O}(h^{p+1})$ to obtain convergence order p. For arbitrary charge functions, this conditions leads to the requirement $x_1-x(h)=\mathcal{O}(h^{p+1})$, and we have the following convergence result: To obtain order p for the network equations (10.1) of index-1, the coefficients of the Rosenbrock method (11.1a,11.1c,11.1d) have to fulfill all order conditions for the algebraic variables up to order p in semi-explicit index-1 systems. This result applies also for a large class of index-2 network equations of the form (10.1).

The coefficients of the method are free to fulfil order conditions for a given method and to guarantee A- and L-stability, respectively. In contrast to multi-step methods, one can construct A- and L-stable methods of arbitrary order.

CHORAL - an embedded method of order (2)3. On account of the low smoothness properties of transistor models, as well as of the low accuracy demands usually required in practice, an embedded method of order (2)3 seems to be suitable. The corresponding scheme, CHORAL, has four stages and only three function evaluations. To avoid a constant term in the error estimate due to inconsistent initial values, both methods are chosen as stiffly accurate [111], and in particular L-stable.

For the general non-homogeneous case of (10.1), the numerical approximation x_k after one timestep from t_{k-1} to $t_k=t_{k-1}+h_k$, together with an embedded approximation $\widehat{x}_k$ of lower order for error control and timestep prediction, is now given by

$$x_k = x_{k-1} + \sum_{i=1}^{s} d_i \kappa_i, \quad \widehat{x}_k = x_{k-1} + \sum_{i=1}^{s} \widehat{d}_i \kappa_i,$$

where the increments κ_i are computed from linear systems

$$\left(\frac{1/\gamma}{h_k}\mathcal{F}_{\dot{x}}^0 + \mathcal{F}_x^0\right)\kappa_i = \frac{1/\gamma}{h_k} A\left(g(x_{k-1}) - g(a_i)\right) - \sum_{j=1}^{i} \widetilde{\beta}_{ij} f(a_j)$$
$$- \sum_{j=1}^{i-1} \widetilde{\beta}_{ij} \frac{\partial f}{\partial x}(x_{k-1}, t_{k-1})\kappa_j - h\widetilde{\tau}_i \frac{\partial f}{\partial t}(x_{k-1}, t_{k-1}),$$

whose right-hand sides can be setup after evaluating the functions $f(a_i)$ and $g(a_i)$ at internal stage values

$$a_i := x_{k-1} + \sum_{j=1}^{i-1} \sigma_{ij}\kappa_j.$$

The corresponding coefficient set of CHORAL with $\widetilde{\beta}_{ij} := \beta_{ij}/\gamma$ and $\widetilde{\tau}_i := \tau_i/\gamma$ is given in Table 5. Since the usual error estimate $\|x_1 - \hat{x}_1\| = \|\kappa_4\|$ for stiffly-accurate embedded ROW methods is used, a reliable error control and stepsize selection are offered that are based on node potentials and branch currents only. This makes timestep control very elegant, especially in comparison with the techniques discussed in the previous section for multi-step methods.

Practical experience. The implementation of CHORAL in an industrial circuit simulation package opened the possibility to gain experience not only with simple standard benchmark examples like an LC oscillator and MOS ringoscillator, but also for numerous real life problems [119]. Some of them are included in Table 6: A 16 bit adder, critical path circuits of dynamic memory (DRAM) circuits, and an arithmetic logical unit ALU, which is the core of a central processing unit.

We see that CHORAL can cope even with large problems, and is competitive with BDF not only with respect to CPU times (Table 6) but also with respect to accuracy, see Fig. 13.

One reason for the efficiency of CHORAL seems to be the stepsize and error control that allow large stepsizes by only a few failures of the stepsize predictions. These results are confirmed by

$\gamma = 0.5728160624821349$	$\beta_{21} = -2.0302139317498051$
$d_1 = \widehat{d}_1 = \sigma_{21} = \sigma_{31} = \sigma_{41} = 1/\gamma$	$\beta_{31} = 0.2707896390839690$
$d_2 = \widehat{d}_2 = \sigma_{32} = \sigma_{42} = 0.0$	$\beta_{32} = 0.1563942984338961$
$d_3 = \widehat{d}_3 = \sigma_{43} = 1.0$	$\beta_{41} = 2/3$
$d_4 = 1.0$	$\beta_{42} = 0.08757666432971973$
$\alpha_2 = 1.0$	$\beta_{43} = -0.3270593934785213$
$\alpha_3 = 1.0$	$\gamma_1 = \gamma$
$\alpha_4 = 1.0$	$\gamma_2 = -2.457397870$
$\tau_1 = 0.3281182414375370$	$\gamma_3 = 0$
$\tau_2 = -2.57057612180719$	$\gamma_4 = 0$
$\tau_3 = -0.229210360916031$	
$\tau_4 = 1/6$	

Table 5: Coefficients for CHORAL

Circuit	# transistors	# equations	CPU time	
			CHORAL	BDF2
LC oscillator	0	3	0.57s	0.33s
MOS ringoscillator	134	73	30.13s	27.61s
16 bit adder	544	283	2m41.32s	2m30.1s
1 Mbit DRAM	2005	1211	10m16.18s	8m29.15s
16 Mbit DRAM	5208	3500	23m37.18s	12m5.11s
ALU	13005	32639	97m31.64	82m21.03s

Table 6: CPU times: CHORAL versus BDF2 on HP workstation C200

numerical tests reported in [96] for digital circuits, NAND gate and 2 bit adder: For non stringent accuracy demands required in network analysis, CHORAL turned out to be as powerful and efficient as DASSL [26], the latter being a standard code for BDF integration of low index DAEs.

Particularly appealing are CHORAL's damping properties: Excitations and oscillations with physical significance are tracked, but perturbations are damped. This behaviour will be discussed in more detail in the following section.

12 Oscillatory circuits and numerical damping: A comparison

Dealing with oscillatory behaviour, we have to distinguish between two types of oscillations. The first type is given by oscillations of physical significance which reflect the behaviour of the mathematical model and the circuit, and should be preserved during numerical integration. The LC oscillator shown in Fig. 14 (left side) can serve as a basic example. This linear circuit consists of one capacitance $C = 4\,\mathrm{pF}$ and inductance $L = 1\,\mathrm{nH}$ in parallel driven by an initial current source $I_0 = 6\,\mathrm{A}$. Numerical approximations obtained by CHORAL and BDF2 are given in Fig. 14 (right side) for the branch current through the inductor. The current oscillates with the amplitude given by I_0 and frequency $\omega = 1/\sqrt{LC}$, which corresponds to a period of $T = 2\pi/\omega \approx 0.4\,\mathrm{nsec}$. While an error becomes visible both in phase and amplitude for BDF2, both phase and amplitude are preserved by CHORAL.

The second type is given by high frequent numerical noise, which should be attenuated by the integrator. It may be due to failures of stepsize and error control, or due to an inappropriate

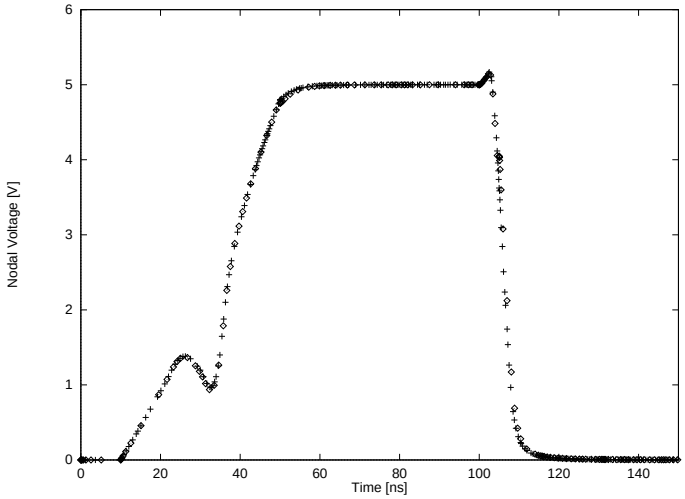


Figure 13: One output nodal voltage for the 16 bit adder: Integration steps of CHORAL ($\diamond$) vs. BDF2 (+).

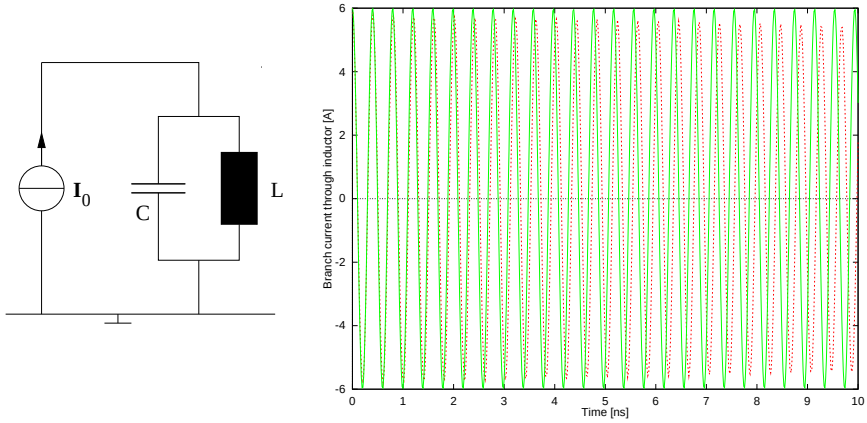


Figure 14: LC oscillator (left) and simulation results (right) for BDF2 (- -) and CHORAL (—)

semidiscretization of a PDE model with respect to space [102]. A third possible origin are discontinuities of the solution, which might be invoked by non smooth transistor models or input stimuli. In the latter case no problems should occur if integration is stopped at these points, and restarted with consistent initial values. Here the algorithms discussed in Section 10 for consistent initialization can be used efficiently for CHORAL, too. However, if integration is not stopped at these points, one may have to deal with inconsistent initial values. An example for such an effect is the current waveform of an operational amplifier circuit, for which the numerical results of the trapezoidal rule and CHORAL are shown in Fig. 15. At ≈ 1 nsec there is a sharp spike, which is invoked by traversing a discontinuity of a MOS capacitance model. Due to its energy conserving property, the trapezoidal rule maintains this perturbation, turning it into an oscillation with the actual timestep as period. This yields an impression that the circuit is unstable and oscillates.

In contrast, the perturbation is damped immediately by CHORAL, and only a few steps are necessary to get back to the smooth solution. The results with TR-BDF are not given here, but are similar to those of CHORAL.

Model equation: Harmonic oscillator. To explain these results for both physical and artificial oscillations, we investigate the model equation

$$\ddot{x} + \omega^2 x = 0 \tag{12.1}$$

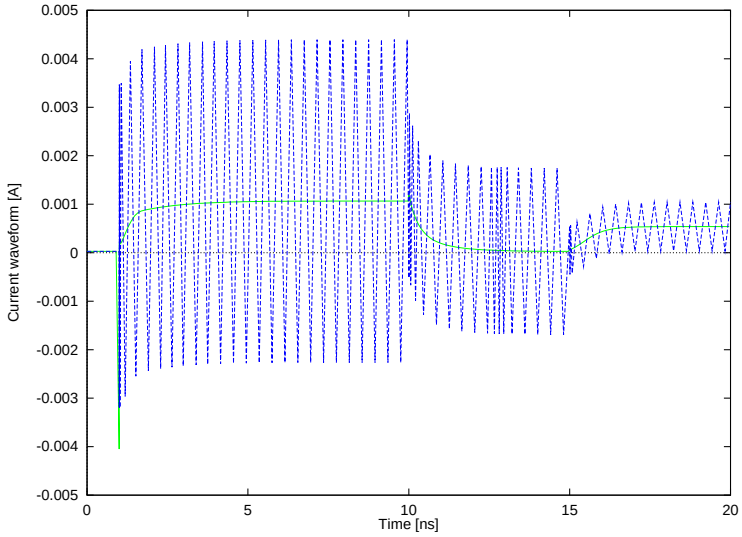


Figure 15: Operational amplifier circuit: Simulation results for trapezoidal rule (- - -) and CHORAL ($\cdots$)

of a harmonic oscillator with frequency ω over one period $[0, T := 2\pi/\omega]$. With initial values $x(0) = x_0$, $\dot{x}(0) = \dot{x}_0$, the solution reads

$$x(t) = r \cdot \mathrm{Re}\exp(i\varphi)$$

with

$$r := \sqrt{x_0^2 + \dot{x}_0^2/\omega^2}, \quad \varphi := \omega t - \arctan(\dot{x}_0/(\omega x_0)).$$

Note that the LC oscillator discussed above corresponds to a harmonic oscillator with frequency $\omega = 1/\sqrt{LC}$.

The results obtained on model equation (12.1) with initial values $(x(0),\, \dot{x}(0))^\top = (1,\, 0)^\top$ for BDF2 and the trapezoidal rule, the integration schemes TR-BDF is based on, and CHORAL are given in Figs. 16 and 17. For each method one period was resolved with *sample rate* $n = 1, 2, \ldots, 1000$ steps of equidistant stepsize $h = T/n$.

Comparing the numerical approximations with the exact solution $(x(T),\, \dot{x}(T))^\top = (1,\, 0)^\top$ after one period, we see the following: Due to its energy conserving property, the trapezoidal rule generates no magnitude error for any n; however, for small sample rates one has to deal with rather large phase errors. BDF2 acts much worse: In addition to a phase error, one has to deal with amplitude errors, if one period is sampled too roughly. CHORAL, however, has only slight amplitude and phase errors even for rather small sample rates.

These results become more visible, if we zoom into the results for $n = 10, 11, \ldots, 20$. As a rule-of-thumb in circuit simulation, one has to sample one oscillation with approximately 10–20 points to get results which are accurate enough. Thus oscillations of physical significance which are approximated numerically using sample rates in the range of 10–20 yield rather large phase errors (trapezoidal rule) or both amplitude and phase errors (BDF2). CHORAL, however, is highlighted by only slight errors in phase and amplitude.

Analysis of one-step methods. These good properties of CHORAL applied to oscillatory circuits can be explained by investigating the model equation in more detail. Besides that, this analysis can illustrate its excellent damping properties as well. As a first step, we scale and rewrite (12.1) as an ODE system of first order. With $y := [x, \dot{x}/\omega]^\top$ we have

$$\dot{y} = Jy, \quad y(0) = \begin{pmatrix} x_0 \\ \dot{x}_0/\omega \end{pmatrix}, \tag{12.2}$$

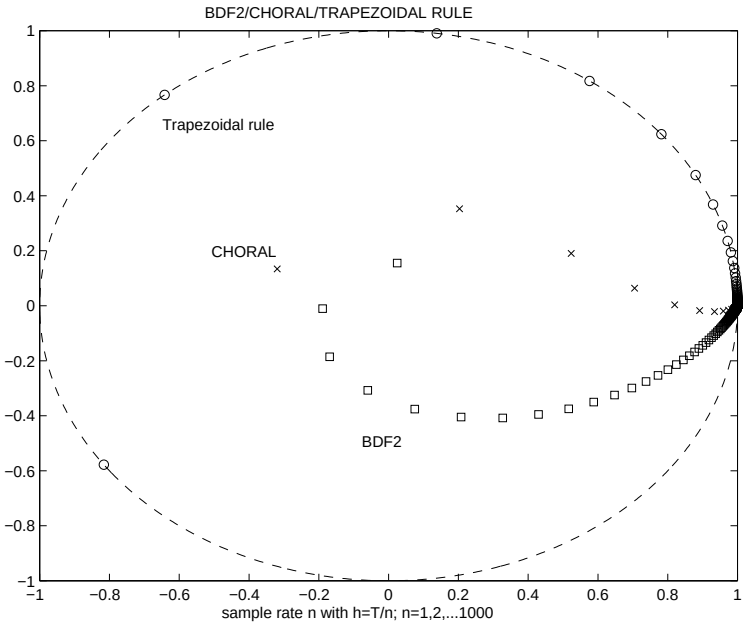


Figure 16: Numerical approximation of BDF2 (□), trapezoidal rule (◯) and CHORAL (×) for model equation (12.1) after one period $T = 2\pi/\omega$. The results are plotted for stepsizes $h = T/n$ $(n = 1, 2, \ldots, 1000)$ in phase space $x = r\exp(i\varphi)$.

where

$$J = \begin{pmatrix} 0 & \omega \\ -\omega & 0 \end{pmatrix}.$$

For one-step methods such as trapezoidal rule and CHORAL, the numerical solution y_n^h after one period with n equidistant steps of size $h = T/n$ reads

$$y_n^h = [R(hJ)]^n \begin{pmatrix} x_0 \\ \dot{x}_0/\omega \end{pmatrix}.$$

The stability matrix $R(hJ)$ has eigenvalues $R(\pm i\omega h)$ which are given by evaluating the scalar stability function $R(z)$ for imaginary arguments $z = \pm i\omega h$. Furthermore, its eigenvectors are $(1,\, i)^\top$ and $(1,\, -i)^\top$, and thus it holds

$$[R(hJ)]^n = U \begin{pmatrix} R(z)^n & 0 \\ 0 & R(-z)^n \end{pmatrix} U^{-1}, \quad U = \begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix}.$$

Therefore the numerical properties of a one-step method applied to the model equation is fixed by its stability function along the imaginary axis:

$$y_n^h = U \begin{pmatrix} R(z)^n & 0 \\ 0 & R(-z)^n \end{pmatrix} U^{-1} \begin{pmatrix} x_0 \\ \dot{x}_0/\omega \end{pmatrix},$$

see Fig. 18. Note that we have

$$\lim_{z\to 0} R(z) = 1$$

for convergent methods, and

$$\lim_{z\to\pm\infty} R(z) = 0,$$

for L-stable methods [111]. Thus there is a range of small stepsizes where $|R(z)|$ is close to one and information is almost preserved, and another range where $|R(z)|$ tends to zero and strong damping prevails.

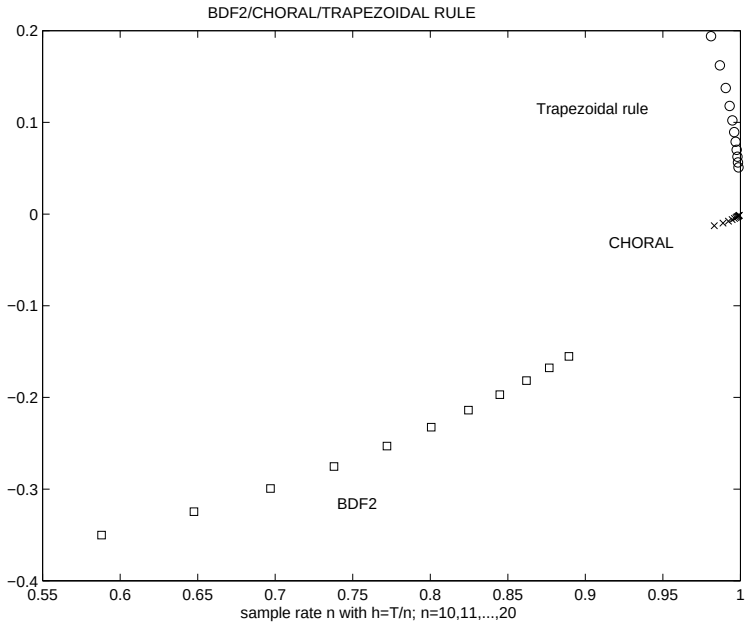


Figure 17: Zoom into numerical approximation of BDF2 (□), trapezoidal rule (◯) and CHORAL (×) on model equation (12.1) in phase space with sample rates $n = 10, 11, \ldots, 20$.

Depending on the type of oscillation, we demand different properties:

- *Oscillations of physical significance.* These should be preserved. Assuming a sample rate of 10–20 steps for oscillations of physical significance, we demand $|R(z)| \approx 1$ in the range of $|z| \in [0.1\pi, 0.2\pi]$.
  Having a look at Fig. 18, we see that this demand is fulfilled by all methods but the implicit Euler scheme.
- *Perturbations.* Such oscillations of high frequency, either numerical noise caused by timestep and error control, inconsistent initial values or by an inappropriate semidiscretization of a PDE model, should be damped as much and soon as possible. Hence a slight damping should already occur for $|z|$ larger than 0.2π, the limit for oscillations of physical significance, and $|R(z)| \approx 0$ for highly oscillatory signals, i.e. $|z| > 100$.
  Except the trapezoidal rule, which is not L-stable, all methods show good damping properties for highly oscillatory signals ($|z| > 100$). But only the implicit Euler scheme and CHORAL damp already for $|z| > 0.2\pi$.

Summing up, CHORAL shows all the desired properties of a (non-ideal) numerical low pass filter: Physical oscillations of low frequency are preserved, but highly oscillatory perturbations are efficiently damped.

The corresponding analysis for multi-step methods can be found in [104].

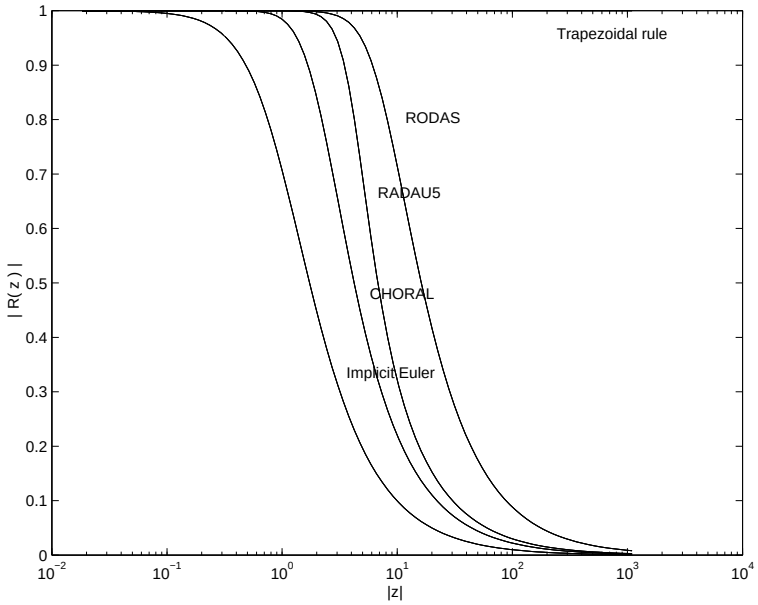


Figure 18: Decay of stability functions along the imaginary axis (from left to right: Implicit Euler, CHORAL, RADAU5, RODAS, and trapezoidal rule with $|R(z)| \equiv 1$).

Chapter IV

Numerical treatment of large problems

Due to their reliability and robustness software codes employing the standard algorithms are established as workhorses, which are inevitable when designing electronic circuits. Especially for integrated circuit design one can distinguish two different steps in the design flow, where these tools are used:

- The *electrical design* stage comprises standard applications for characterization and optimization of functional building blocks, such as gates, operational amplifiers, oscillators etc.. These analyses are run excessively in order to make sure that the functional units meet their specifications under a large variety of load and bias conditions and temperatures, and with different parameter sets representing the fluctuations of the technological processes in the fabrication lines.

- In the *verification stage* overall functionality of the circuit is checked. For this purpose the circuit parts containing the critical path inclusive parasitics — like capacitances and resistances of junctions and interconnects — are re-extracted from layout. This yields accurate but very large circuit models with many input nodes, which have to be biased with rather lengthy input stimuli in order to verify overall functionality of the circuit.

Typical data for these different kinds of application are given in Table 7. The dimension of the mathematical circuit model corresponds approximately to its number of transistors. Since the transistor models are fairly complex, most time in standard applications is spent for setting up the matrix and right hand side of the resulting linear system ('load'). Due to the overlinear increase of sparse Gaussian elimination, the computational expense for the linear solver becomes dominant for large applications. The overhead spent for timestep and convergence control etc. is usually below 5%.

Since the turnaround times for large applications are often beyond desirable limits, i.e. significantly more than $5 \cdots 8$ hours, it is of a major interest to obtain speedups without sacrificing

	standard application	large application
no of transistors	$10^1 \ldots 10^3$	$10^3 \ldots 10^5 (\ldots 10^6)$
no of equations	$10^1 \ldots 10^3$	$10^3 \ldots 10^5 (\ldots 10^6)$
no of timesteps	$10^2 \ldots 10^3$	$10^3 \ldots 10^6$
CPU times (on workstation or PC)	sec ... min	hours ... days
load	85%	$85\% \ldots < 50\%$
lin. solver	10%	$10\% \ldots > 50\%$
overhead	5%	$5\% \ldots 2\%$

Table 7: Typical data for standard and large applications in circuit simulation

Source of expense	Speedup possible by
complexity of device models	- *higher level of abstraction,* using functional modelling with languages like VHDL-AMS - use of *table models* for devices or subblocks like gates
overlinear expense for Gauss solver - typically $n^{1.2} \cdots n^{1.8}$ (n: number of circuit nodes)	*decomposition* $\Rightarrow$ decoupling into smaller blocks $\Rightarrow$ use of iterative methods
lack of adaptivity - global timestep control - global convergence control	*decomposition* $\Rightarrow$ *higher degree of adaptivity* by exploiting different activity of different circuit parts at different times
large number of devices	*parallelization*

Table 8: How to speedup circuit simulation

accuracy, universality and robustness too much.

Many attempts are pursued in the literature to overcome the computational limitations of circuit simulation. Table 8 shows the basic principles of these approaches, and which kind of problem they aim to improve.

The first row of Table 8 concerns modelling issues, which are not to be discussed here. The remaining rows of Table 8 roughly characterize the main aspects of our further discussion. First a glance at a simple MOS ringoscillator example will illustrate typical properties of the mathematical circuit models, which offer potentials for getting improvements.

13 Numerical properties of an MOS ringoscillator model

The task of a ringoscillator in bipolar technology and its basic principles were already explained in Section 9. Most of the very complex integrated circuits challenging circuit simulation however are fabricated in MOS technologies. As a typical representative we will now consider a simple ringoscillator in complementary MOS (CMOS) technology, and highlight some interesting properties of this circuit class.

Fig. 19 shows a circuit diagram of a CMOS ringoscillator consisting of 11 inverter stages, which are connected in a feedback loop. Each inverter is composed of a P-type MOS transistor — which is connected to power supply VDD — and of an N-type MOS transistor connected to ground. Furthermore a parasitic wiring capacitance to ground is added. When the input signal at the gate nodes of both transistors of an inverter is higher than a certain threshold voltage then the P-channel transistor is OFF, and the N-channel transistor is conducting, thus pulling the output signal at the common drain node to ground. Inversely, a low level input signal switches the N-channel transistor OFF and the P-channel transistor ON, such that the output node is loaded up to power supply voltage VDD. The inverted output signal drives the next inverter, and after passing all stages of the closed loop, it arrives with a certain time delay as input signal of the first one. This invokes an oscillation, and its period is usually just $2 \cdot 11$ times the average switching delay of one inverter stage. The waveforms of nodes 1, 6, and 11 are shown in Fig. 20; the other waveforms are identical — if the design is regular — but shifted in time.

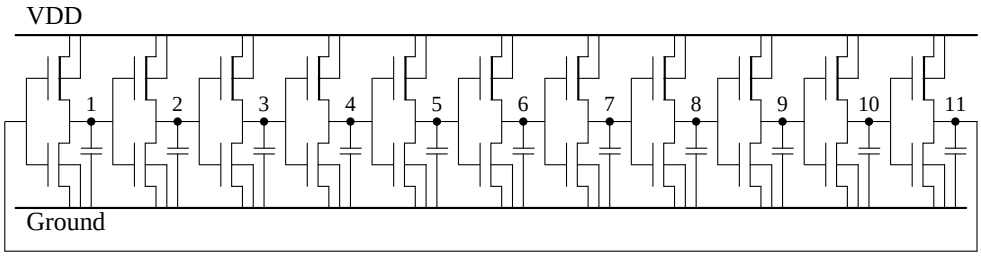


Figure 19: CMOS ringsoscillator

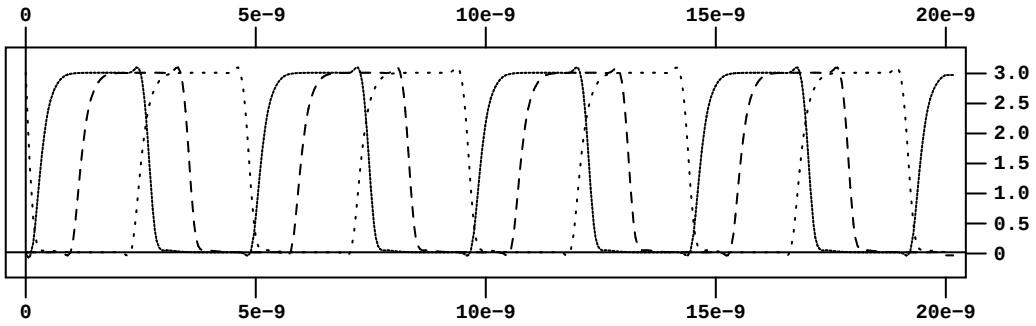


Figure 20: CMOS ringoscillator – Waveforms (in Volt) over time (in sec): Nodes 1 (——), 6 (– – –) and 11 ($\cdots$)

Multiscale. When looking at the waveforms of node 1 and 11 in Fig. 20 we recognize that the first inverter in the loop is only active in fairly small parts of an oscillation cycle, and more or less quiescent else. The same is true for all other inverters; but they are active at different time windows, since the signal is continuously propagating through the circuit. So the varying degree of activity for different circuit parts has usually no computational effect: Timestep control always has to take care about the smallest timestep in the whole circuit, unless multirate integration is being used.

In order to get an estimate for the potential benefit of multirate integration, we relate the global timestep h_{glob} to the timestep h_{loc} needed for numerical integration of the first inverter, see Fig. 21. We see that h_{loc} determines the global timestep just when the first inverter is switching, and becomes much larger else. Obviously the relations are quite similar for all inverter stages. Using a slightly modified version of a formula given in [11], we can estimate the possible speedup in this case to be:

$$speedup = \frac{n \cdot m}{n_L + n_A \cdot m} \approx \frac{1}{\text{mean value}(h_{glob}/h_{loc})}$$

where n is the number of devices, n_A is the average number of active devices, $n_L = n - n_A$ the average number of inactive devices, and m is the average number of global timesteps within a timestep in case of no activity. Measuring the mean value from Fig. 21, we get

$$speedup \approx \frac{1}{0.24} \approx 4.2,$$

or with a more practical restriction of the local timesteps to $\leq 5 \cdot h_{glob}$ still a speedup factor of approximately 3.

In reality this figure will become smaller due to inevitable overhead; on the other hand it may further increase for larger circuits. So we conclude that multirate integration seemingly offers significant speedup potential for circuit simulation.

Latency. Another effect of the varying degree of activity is the different rate of convergence for different circuit parts, when applying fully implicit integration methods like BDF. Fig. 22

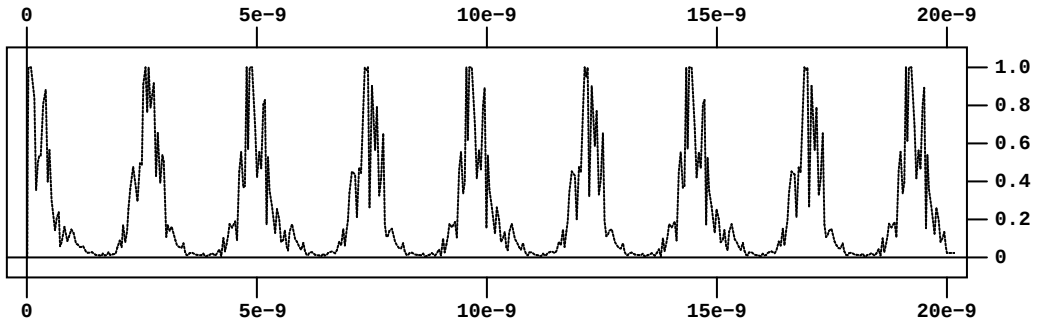


Figure 21: CMOS ringoscillator – Timestep ratio h_{glob}/h_{loc} for the first inverter, over time

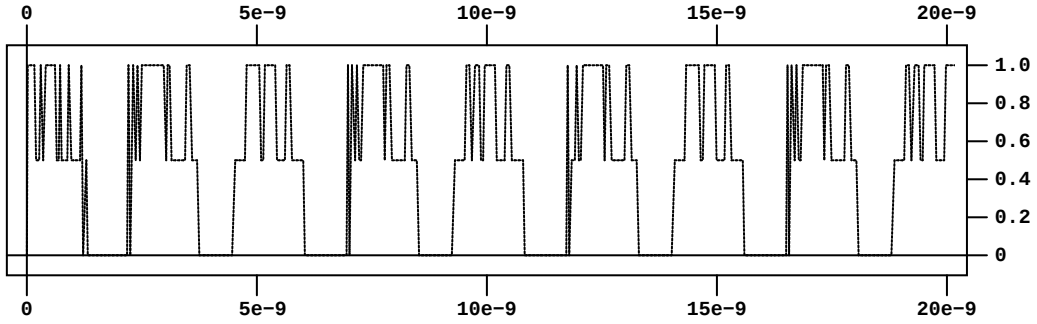


Figure 22: CMOS ringoscillator – Iteration count ratio nc_{loc}/nc_{glob} for node 1, over time

shows as an example the ratio nc_{loc}/nc_{glob} over time, where nc_{loc} counts the number of Newton iterations needed per timestep to get convergence of node 1, and nc_{glob} is the global iteration count per timestep. Similar to the multirate formula we get a rough estimate

$$speedup = \frac{n}{n_L \cdot \mu + n_A}$$

for an algorithm which exploits the different rate of convergence for different circuit parts. Here, μ is an average ratio of iteration counts for the inactive and the active circuit parts. For our ringoscillator, an approximation follows which can be directly measured from Fig. 22:

$$speedup \approx \frac{1}{\text{mean value}(nc_{loc}/nc_{glob})} \approx \frac{1}{0.48} \approx 2.1$$

A special case would be to omit re-evaluation of circuit parts which do not change from one timestep to the next ($\mu = 0 \rightarrow$ 'latency'). This gives an estimated $speedup = n/n_A$ in this case, making the exploitation of latency an interesting alternative to multirate integration.

Unidirectional signal flow. An inherent property of MOS transistors is to have — almost — no static current flow from the gate node into the device. So, when the signal flow in a circuit is passing the gate of an MOS transistor then it is mainly unidirectional in a local sense, and only dynamic effects can cause local feedback. This is illustrated in Fig. 23, where static and capacitive coupling in forward and backward direction is shown for the first inverter of the CMOS ringoscillator. Static backward coupling is negligible. Note that although capacitances are small, their coupling effect is comparable to static coupling, which is due to the high switching speed of $10^9 \ldots 10^{12}$ Volt per sec.

Unfortunately, those circuit configurations which propagate signals between source and drain node of the MOS transistors — like bus structures — do *not* exhibit unidirectional signal flow. Furthermore, *global* feedback coupling principles are extensively applied in circuit design.

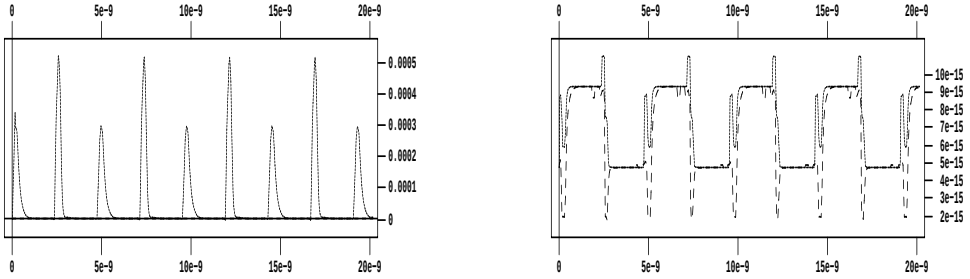


Figure 23: CMOS ringoscillator – Forward (——) and backward (- - -) coupling coefficients for the first inverter, over time. Static/capacitive coupling is shown in the left/right diagram.

Parallelism. Finally we see that the circuit schematic consists of a large number of identical primitives (here: MOS transistors and capacitors), thus offering speedup potential by handling them in parallel. Since the model equations of transistors are usually very complex and contain many if-then-else branches, their evaluation is well suited for a medium or coarse grain type of parallelization.

14 Classification of existing methods

In the literature there is a rich variety of attempts to overcome the computational limitations of standard circuit simulation. A rough overview is given in Fig. 24. We see that decomposition techniques [44, 108] are applied at almost all stages of the standard algorithms, in order to

- apply relaxation methods,
- introduce a higher degree of adaptivity, and
- improve performance on parallel computers.

Single boxes (with larger fonts) in Fig. 24 represent single but large systems, while double boxes (with smaller fonts) indicate sets of decomposed, smaller subsystems.
The three columns on the left (ROW, MLN and standard) concern algorithms which are sufficiently general to cope with any circuit of not too high DAE index. While standard TR-BDF as well as ROW integration are discussed in Chapter III, the multi-level Newton method (MLN) will be described in more detail later on. A multi-level direct linear solver (block Gauss solver) is not included in the figure, since it can be seen as a special case of the multi-level Newton solver. Furthermore, modifications of the Newton method in the standard solver — as are described in [59, 60] — are not included due to space limitations.
The right five columns (ITA, WR, WRN, PWL and Exp. Fit) describe approaches which can be efficiently used only for a restricted class of circuits, e. g., for more or less digital MOS circuits. These methods are shortly reviewed below; more details can be found in the survey papers [44, 173] and in the book of White and Sangiovanni-Vincentelli [259]. Further developments are reviewed in [213, 241], and their specific strengths and limitations are compared.

The formulas given below refer to network equations given in the compact form (10.1a). We assume that variables and equations are partitioned and reordered, such that each subblock i is characterized by just one entry in

$$x = (x_1, \dots x_i, \dots x_m)^T, \qquad f = (f_1, \dots f_i, \dots f_m)^T, \qquad A = (A_1, \dots A_i, \dots A_m)^T,$$

with m being the number of subblocks. Furthermore we assume that implicit multi-step methods

$$\dot{y} = \alpha y + \beta$$

are applied with α as leading integration coefficient and β giving the contributions of previous timepoints.

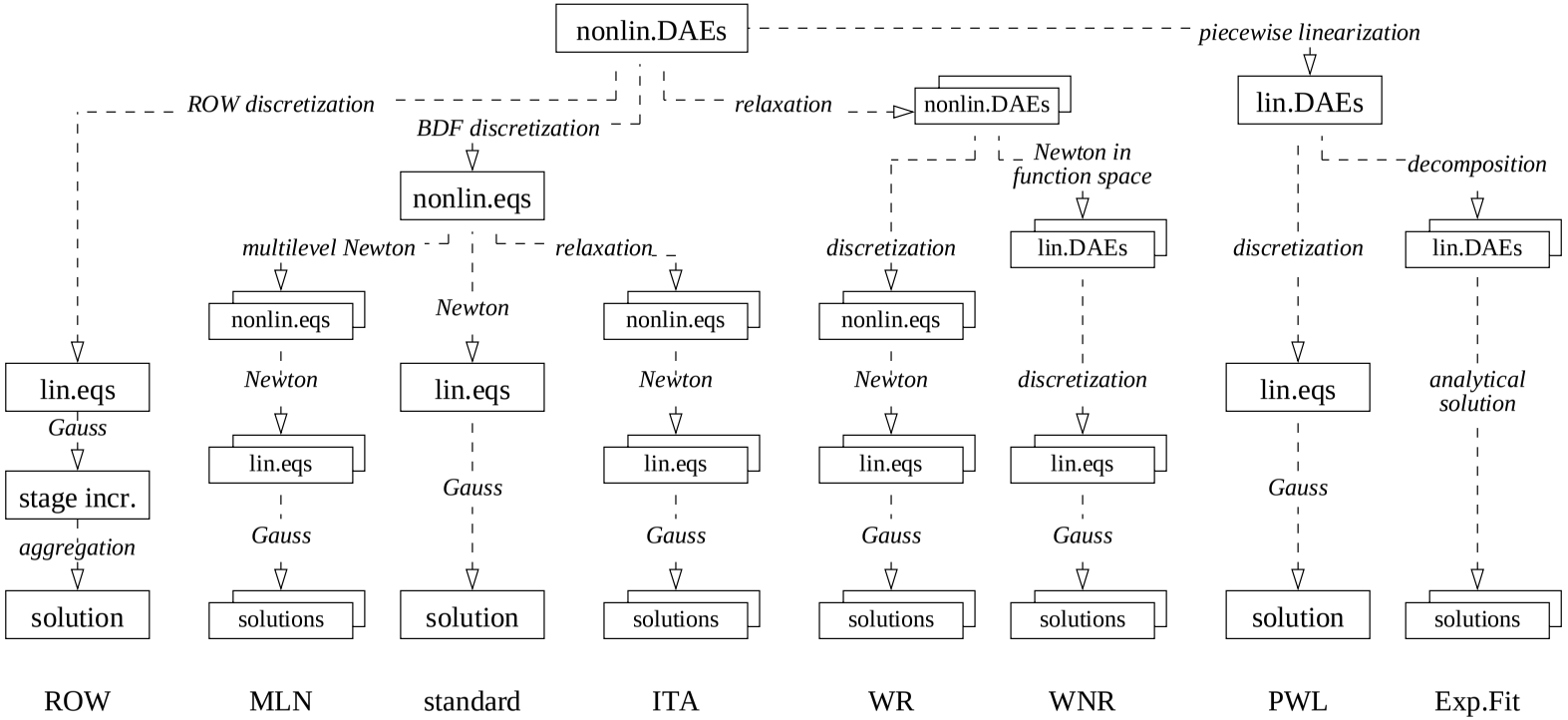


Figure 24: Existing methods for circuit analysis in the time domain.
(ROW: Rosenbrock Wanner method; MLN: multi-level Newton; ITA: iterated timing analysis; WR: waveform relaxation; WRN: waveform relaxation Newton; PWL: piecewise linear analysis; Exp.Fit: exponential fitting.)

ITA – iterated timing analysis. Historically, timing simulation was the first attempt to compute approximate waveforms for very large digital MOS circuits [36]. Here the nonlinear systems were completely decomposed into single equations, which were approximately solved by performing one single Gauss-Jacobi or Gauss-Seidel step. In iterated timing analysis ITA the applicability of the method is significantly extended by blockwise decomposition, where the subblocks are solved with conventional methods [43, 211]. Hence for each relaxation sweep j a sequence $i = 1, \dots m$ of subsystems

$$A_i \cdot (\alpha y(x^j) + \beta) + f_i(x^j, t) = 0$$

has to be solved for x_i^j with

$$x^j = \begin{cases} (x_1^{j-1}, x_2^{j-1}, \dots, x_{i-1}^{j-1}, x_i^j, x_{i+1}^{j-1}, \dots x_m^{j-1})^T & \text{for Gauss-Jacobi relaxation} \\ (x_1^j, x_2^j, \dots x_i^j, x_{i+1}^{j-1}, \dots x_m^{j-1})^T & \text{for Gauss-Seidel relaxation.} \end{cases} \tag{14.1}$$

The l-th iteration of the inner Newton process is described by

$$x_i^{j,l+1} = x_i^{j,l} + \triangle x_i^{j,l},$$

where $\triangle x_i^{j,l}$ is computed from

$$\left(\alpha A_i \cdot \left.\frac{\partial y}{\partial x}\right|_{x=x^{j,l}} + \left.\frac{\partial f}{\partial x}\right|_{x=x^{j,l}} \right) \triangle x_i^{j,l} = -A_i \cdot (\alpha y(x^{j,l}) + \beta) - f_i(x^{j,l}, t).$$

The particular Newton iterate reads, e.g., for Gauss-Seidel relaxation as

$$x^{j,l} = (x_1^j, \dots x_{i-1}^j, x_i^{j,l}, x_{i+1}^{j-1}, \dots x_m^{j-1})^T.$$

A convergence proof of ITA is given in [240] for circuits with strictly diagonal dominant capacitance matrix and Lipschitz continuous conductance matrix, provided that the timesteps are sufficiently small.

For efficiency reasons, often only one single Newton step is performed per relaxation sweep in ITA. This may cause some loss of accuracy and reliability, which is however acceptable in many cases. Adaptivity can be improved by exploiting the different activity of different circuit partitions. This is implemented in some kind of event control: Only those partitions of the system are scheduled for computation, which are activated by changing signals at their borders.

WR – waveform relaxation. This method is basically an application of Picard iteration to the network equations. The method can also be characterized as a block Gauss-Seidel-Newton or block Gauss-Jacobi-Newton method in the function space, since after decomposition of the circuit into subblocks the subsystems are solved for a whole waveform with standard methods, while their coupling is handled with relaxation methods. That means that for each relaxation sweep j a sequence of subsystems

$$A_i \cdot \dot{y}(x^j) + f_i(x^j, t) = 0 \qquad i = 1, \dots m$$

has to be solved for $x_i^j(t)$ in the time intervall $0 \le t \le t_{end}$ with x^j being defined by (14.1).

WR was first discussed in [153], and a global convergence proof was given for circuits with a grounded capacitor at each node, provided that some Lipschitz conditions hold and the time windows used are sufficiently small.

The method has found much interest in the literature, since it offers a very natural way to improve adaptivity in form of multirate by integrating each subblock with its own timestep:

$$\dot{y} = \alpha_i^j y + \beta_i^j$$

A survey including many practical aspects of WR methods is given in the book edited by Ruehli [42]. Efficient parallelization of WR is described in [176]. Recent convergence theorems given in [90] extend the class of feasible circuits and provide insight how to decompose the circuit

for getting good convergence rates. The latter aspect has turned out to be a key issue for the performance of WR. Since simple partitioning schemes based on circuit topology sometimes give no satisfactory results, information about the entries of the Jacobian is often used for this purpose. This may even require repartitions from time to time, especially in case of strongly nonlinear circuits [265].

Most of the literature published about WR deals with ordinary differential equations and therefore requires to have a grounded capacitor at each node, at least at the decoupling subblock borders. Extensions to DAEs with non differential — i.e. algebraic — coupling equations are discussed in [5]; it is shown that certain contractivity conditions additionally must hold in order to ensure convergence in these cases.

WRN – waveform relaxation Newton. If the Newton process is applied directly in the function space, before time discretization, then we get the waveform Newton method WN:
Compute

$$x^{l+1}(t) = x^l(t) + \triangle x^l(t)$$

in the time interval $0 \le t \le t_{end}$ from solving

$$A \cdot \frac{d}{dt}\left(\left.\frac{\partial y}{\partial x}\right|_{x=x^l(t)} \cdot \triangle x^l\right) + \left.\frac{\partial f}{\partial x}\right|_{x=x^l(t)} \cdot \triangle x^l = -A \cdot \dot{y}(x^l) - f(x^l, t).$$

for $\triangle x^l(t)$. With

$$C^l(t) = A \cdot \left.\frac{\partial y}{\partial x}\right|_{x=x^l(t)}, \qquad G^l(t) = \left.\frac{\partial f}{\partial x}\right|_{x=x^l(t)}$$

this reads as

$$C^l(t) \cdot \frac{d\triangle x^l}{dt} + \left(G^l(t) + \dot{C}^l(t)\right) \triangle x^l = -A \cdot \dot{y}(x^l) - f(x^l, t).$$

Note that $\triangle x^l(0) = 0$ if $x^l(0) = x_0$.

A convergence proof of WN is given in [213] for very general circuit classes.

At a first glance, this method seems not to be very attractive for circuit simulation, since one cannot expect that initial waveforms are close to the solution. Its main advantage is that it yields systems of *linear* DAEs. These can eventually be solved with discretization methods which are more efficient than standard integration [181]. The main motivation for presenting this method here is however, that it serves as a base for waveform relaxation Newton WRN.

If the nonlinear DAE subsystems of the WR method are solved with the WN method, then we get the WRN method:
Solving

$$A_i \cdot \dot{y}(x^j) + f_i(x^j, t) = 0 \qquad i = 1, \ldots m$$

with WN means to compute

$$x_i^{j,l+1}(t) = x_i^{j,l}(t) + \triangle x_i^{j,l}(t)$$

in the interval $0 \le t \le t_{end}$ from solving

$$C_i^{j,l}(t) \cdot \frac{d\triangle x_i^{j,l}}{dt} + \left(G_i^{j,l}(t) + \dot{C}_i^{j,l}(t)\right) \cdot \triangle x_i^{j,l} = -A_i \dot{y}(x^{j,l}) - f_i(x^{j,l}, t),$$

for $\triangle x_i^{j,l}(t)$, where $x^{j,l}$ is given by (14.1) and

$$C_i^{j,l}(t) = A_i \left.\frac{\partial y}{\partial x}\right|_{x=x^{j,l}}, \qquad G_i^{j,l}(t) = \left.\frac{\partial f_i}{\partial x}\right|_{x=x^{j,l}}.$$

We see again that here the DAEs are linear time variant.

A convergence proof for this method can be derived from the convergence of the WR and WN methods, from which it is composed [213, 240]. The adaptivity of this method is high due to its natural support of multirate integration. For efficiency reasons timesteps should

be coarse in early stages of the relaxation process, and become finer only when it approaches convergence. The method is reported to be superior over WR for circuits which do not have a strongly unidirectional signal flow. For efficiency often only one Newton step is performed per relaxation; the price of slightly reduced accuracy and reliability seems to be acceptable in many applications. Finally, WRN allows for an efficient parallelization [212].

PWL – piecewise linear analysis. In an alternative approach, the nonlinear device characteristics are approximated by piecewise linear models. The resulting linear DAE systems are only piecewise valid. When they are solved with conventional time discretization schemes like BDF, then timestep control has to take care that their region of validity is not left within the timestep. This may slow down efficiency, if the model resolution is fine. For finding a solution, improved versions of Katzenelson's algorithm are used in general [136, 249, 264].

If the validity regions are explicitly included as constraints for the piecewise linearized network equations, then extra "state variables" have to be introduced, which define for which particular region a certain linear relation is valid. This *piecewise linear mapping* leads to systems of the following form:

$$\begin{aligned} A \cdot \dot{y} + F \cdot x + B \cdot z + f_0(t) &= 0 && \text{piecewise linearization of f(x,t)} \\ y = G \cdot x + E \cdot z + y_0 & && \text{piecewise linearization of y(x)} \\ \overline{z} = H \cdot x + D \cdot z + z_0 & && \text{definition of region of validity} \\ z \geq 0; \qquad \overline{z} \geq 0 & && \\ z^T \cdot \overline{z} = 0 & && \text{complementarity of state variables} \end{aligned}$$

The dimension of the state variables $z, \overline{z} \in \mathbb{R}^{n_z}$ defines the maximal number of different regions of validity to be 2^{n_z}, since each component of z can be selected to be either $= 0$ or > 0, and the corresponding component of $\overline{z}$ is then defined from the complementarity condition. The crossing of a border between two regions of validity is characterized by just one component of z or $\overline{z}$ to become zero. Fortunately, this crossing can be performed by a rank one update of the system matrix; and if a hierarchical LU decomposition method is used for solving the linear system, this does not require much extra effort. A review of these techniques can be found in [243].

In [155] piecewise linear circuit equations are obtained by mapping nonlinear conductances and capacitances into time variant linear conductances and capacitances, respectively.

The most appealing aspects of piecewise linear analysis PWL are, that no Newton iterations are necessary [155], strong global convergence properties, and a uniform kind of modelling, based on tabulated data [247].

Exp.Fit – exponential fitting. If a PWL circuit model is decomposed into small subblocks then each subsystem can be solved analytically for a certain time interval, until the solution crosses the border of the particular linear model section. These techniques are known as exponential fitting methods [215]. They have shown to offer high simulation speed, especially in timing simulators, where only one relaxation step is performed [12, 177, 248].
A mathematical analysis of exponential fitting methods in circuit simulation is presented in [222]. It starts from the piecewise linearized version of (10.1a)

$$C \cdot \dot{x} + G \cdot x + f_0(t) = 0,$$

which is dicretized with an explicit exponential formula of order 1:

$$x(t_{l+1}) = x(t_l) + D^{-1}(1 - e^{-Dh})\dot{x}(t_l).$$

The solution is of the matrix exponential form

$$x(t) = e^{-Dt}x_0 - G^{-1}f_0(t),$$

where D is given by

$$D = C^{-1}G.$$

Timing simulators of this kind use for D an approximation of $C^{-1}G$, in order to decouple equations. In any way, the methods work only for regular $C = A \cdot \frac{\partial y}{\partial x}\big|_{x=x_{t_l}}$, i.e. ODEs. For the DAE case the Drazin inverse might come into play [261]. Unfortunately numerical evaluation of the matrix exponential containing the Drazin inverse turns out to be cumbersome [195]; so exponential fitting was only applicable to a restricted class of circuits up to now.

Concluding remarks:

- Surprisingly, iterative solvers for the linear equations of the standard or ROW approach are not included in Fig. 24. Although numerous attempts were made in the past to substitute direct Gaussian elimination by iterative solvers in general purpose circuit simulation programs, no one was really successful yet. Basically this is due to a lack of favourate numerical properties of the linearized circuit equations, in combination with restrictive accuracy requirements. Roughly speaking, the use of iterative linear solvers requires very good preconditioners, which are not much cheaper to get than direct LU factors. Furthermore, the widespread use of quasi Newton methods — taking Jacobians and their LU-factors from earlier iterations — alleviates the need for iterative solvers, if the circuit size is not too large, say less than 10^5 nodes. Only recently, promising approaches for iterative linear solvers were presented [14, 217]; both of them are particularly tailored to the specific structure of the network equations.

- Timing simulation has in the past always relyed on multi-step integration or exponential fitting methods. It might be interesting to explore similar techniques on the basis of an advanced one-step method — like that presented in the previous section.

- Today software codes employing ITA or WR or exponential fitting algorithms have obtained a high degree of maturity and robustness, which allows them to be successfully used even in industrial environments. Especially when exploiting hierarchical concepts on highly repetitive circuits, these codes can simulate several clocks of $10^7 \cdots 10^8$ transistor circuits on transistor level with reasonable accuracy in some hours.

  Note however that although these codes often are one or two orders of magnitude faster than standard circuit simulation packages, they cannot really substitute the latter. This is due to their limited focus of applications as well as some lack of accuracy and universality and — even worse — reliability. So we will focus in the remaining part of this chapter on methods to speed up the standard transient analysis algorithms without sacrificing their universality and robustness. One is parallelization, and another one deals with multirate integration.

15 Parallelization

At a first glance, parallel circuit simulation offers a high speedup potential due to

- a large number of devices with fairly complex, but identical characteristic equations;
- large systems of linear equations to be solved;
- a small amount of purely serial overhead.

In practice however, the speedup of parallel versus serial simulation often saturates at a low level — say 2 to 4 — even on very powerful parallel computers. Further improvements can only be obtained by carefully adapting the granularity of parallelism to the particular computer architecture, which has an impact on both algorithms and coding.

A rough classification identifies three different granularity levels of parallelism [41]:

- *Fine grain* parallelism for single instruction multiple data or pipelining architectures like vector supercomputers, which were the workhorses for large industrial circuit simulation tasks in the past. Parallelization is basically achieved by vectorization.
- *Medium grain* parallelism for multiprocessor machines with shared memory. Such systems are presently often installed in industry for complex design tasks. Parallelization here is based on thread concepts.
- *Coarse grain* parallelism on loosely coupled clusters of workstations or PCs. Here it is essential to take care for a minimum data traffic over the local network. Parallelization is based on message passing systems like PVM or MPI. This level may also be useful for shared memory multiprocessors.

Due to reasons of cost effectiveness and flexibility, vector supercomputers are no longer used for circuit simulation. So vectorization will not be further considered here; literature can be found in [58, 59, 70].

While the levels of fine and medium grain parallelism directly aim at the classical circuit simulation algorithms, the coarse grain is best realized with a multi-level Newton method for solving the network equations at a particular timepoint. This method will be described in the following subsection, before we come back to parallel circuit simulation.

Multi-level Newton method

Originally, the multi-level Newton MLN method was developed to solve large nonlinear systems by decomposition without loosing the quadratic convergence of Newton's algorithm [188]. If a proper decomposition can be found then the method offers a good speedup potential by parallel execution on clusters of fast processors, but relatively slow interconnect network.

Our further discussion restricts on a two-level Newton method; an extension to more levels is possible, but not common practice.

We assume that the nonlinear system of equations $f(x) = 0$ with $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}^n$ has a regular Jacobian $\partial f/\partial x$. Further we assume that f is decomposed into m subsystems f_i $(i = 1 \ldots m)$ and one master system f_{m+1}, and x is reordered such that

$$x^\top = (x_1, x_2, \ldots, x_m, x_{m+1}), \quad f^\top = (f_1, f_2, \ldots, f_m, f_{m+1}),$$

where x_i $(i = 1 \ldots m)$ contain the inner variables of f_i and x_{m+1} contains the outer variables. Then

$$\begin{aligned} f_i &= f_i(x_i, x_{m+1}) \qquad (i = 1 \ldots m) \\ f_{m+1} &= f_{m+1}(x_1, x_2, \ldots, x_m, x_{m+1}), \end{aligned}$$

and the Jacobian of f has bordered block diagonal form:

$$\frac{\partial f}{\partial x} = \left(\begin{array}{cccc|c} \frac{\partial f_1}{\partial x_1} & & & & \frac{\partial f_1}{\partial x_{m+1}} \\ & \frac{\partial f_2}{\partial x_2} & & & \frac{\partial f_2}{\partial x_{m+1}} \\ & & \ddots & & \vdots \\ & & & \frac{\partial f_m}{\partial x_m} & \frac{\partial f_m}{\partial x_{m+1}} \\ \hline \frac{\partial f_{m+1}}{\partial x_1} & \frac{\partial f_{m+1}}{\partial x_2} & \cdots & \frac{\partial f_{m+1}}{\partial x_m} & \frac{\partial f_{m+1}}{\partial x_{m+1}} \end{array}\right)$$

Finally it is assumed that all submatrices $\partial f_i/\partial x_i$ $(i = 1 \ldots m)$ are regular; otherwise the decomposition has to be changed. Then the two-level Newton method contains a Newton loop for the master system, where for each outer iteration k the inner systems are solved for x_i with fixed outer variables x_{m+1}^k, see Fig. 25 [6].

Fig. 26 shows an example in two dimensions, i.e. $m = 1$, with the notation $S := 1$, $M := m + 1 = 2$. The inner Newton steps starting from the point (x_S^0, x_M^0) yield a solution on the curve $f_S = 0$ with fixed outer variable x_M^0. Then a Newton step is done into x_M direction to get the point (x_S^1, x_M^1); the latter may be further improved into x_S direction by adding the tangent correction, such that the next inner Newton cycle can start from the point (x_S^{T1}, x_M^{T1}).

The two-dimensional example illustrates how the two-level Newton scheme can be derived: The subsystem $f_S = 0$ defines an implicit relation $x_S = x_S(x_M)$, and the outer Newton method solves $f_M(x_S(x_M), x_M)$ for x_M.

The MLN approach can be characterized as follows:

- Quadratic convergence of the method is shown in [188] under standard assumptions, if the inner nonlinear systems are solved with higher accuracy than the outer ones:

  $$\|\triangle x_i\| \leq \|\triangle x_{m+1}\|^2 \qquad (i = 1 \ldots m)$$

  This may become difficult to achieve, especially for MLN methods with more than two levels. Methods for reducing the number of inner iterations without affecting quadratic convergence are described in [116, 267]. A simple practical rule to get sufficiently superlinear convergence is to solve the inner systems just somewhat more accurately than the actual norm of the outer Newton process, e. g.:

  $$\|\triangle x_i\| \leq \alpha \|\triangle x_{m+1}\| \quad (i = 1 \ldots m) \quad \text{with} \quad \alpha = 10^{-1} \ldots 10^{-2}$$

- A single-level Newton method is obtained, when only one inner iteration is performed and the solution is updated with the tangent correction [266]. This is useful for combining global and multi-level Newton steps, which may improve efficiency and robustness in certain cases [116].
- Due to additional inner Newton iterations one should expect that the multi-level method is more expensive than the standard Newton process. However in practice often nonlinearity can be shifted into the smaller subsystems, thus reducing the number of outer iterations and getting even better efficiency than with the single-level algorithm [77].
- Originally, the tangent correction is not included in the MLN algorithm. Mainly it serves for getting a good start vector for the next inner iteration cycle [126, 260, 266]. In practice it turns out that the tangent correction should be omitted as long as the outer process is still far away from convergence.
- In case of sufficiently decreasing norms, quasi Newton steps may be employed for the outer iteration process by taking the Schur matrices of earlier iterations, and eventually avoiding expensive LU factorization of the outer system [77].

[6] In the linear case, the Schur complement S_i and residuum R_i of Fig. 25 can be easily explained to be the Gauss updates for eliminating x_i in f_{m+1}, i.e. to transform the system into upper triangular form.

- *Initialization:*
  - get start vectors $x_1^0, x_2^0, \ldots x_m^0, x_{m+1}^0$
  - set iteration indices $k = 0, \qquad j_i = 0 \quad (i = 1 \ldots m)$
- *Outer Newton process:*
  do until convergence
  - do for all subsystems $i = 1 \ldots m$
    - *Inner Newton process:*
      do until convergence
      - solve: $\partial f_i / \partial x_i \cdot \triangle x_i^{j_i} = -f_i$
      - add Newton correction: $x_i^{j_i+1} = x_i^{j_i} + \triangle x_i^{j_i}$
      - update inner iteration index: $j_i = j_i + 1$

      enddo
    - compute Schur complement: $S_i = \dfrac{\partial f_{m+1}}{\partial x_i} \cdot \left(\dfrac{\partial f_i}{\partial x_i} \right)^{-1} \cdot \dfrac{\partial f_i}{\partial x_{m+1}}$
    - compute residuum: $R_i = \dfrac{\partial f_{m+1}}{\partial x_i} \cdot \left(\dfrac{\partial f_i}{\partial x_i} \right)^{-1} \cdot f_i$

    enddo
  - solve: $\left(\dfrac{\partial f_{m+1}}{\partial x_{m+1}} - \sum_{i=1}^{m} S_i \right) \cdot \triangle x_{m+1}^k = -f_{m+1} + \sum_{i=1}^{m} R_i$
  - add Newton correction: $x_{m+1}^{k+1} = x_{m+1}^k + \triangle x_{m+1}^k$
  - update master iteration index: $k = k + 1$
  - *Tangent correction:*
    do for all subsystems $i = 1 \ldots m$
    - update inner variables: $x_i^{j_i+1} = x_i^{j_i} - \left(\dfrac{\partial f_i}{\partial x_i} \right)^{-1} \cdot \dfrac{\partial f_i}{\partial x_{m+1}} \cdot \triangle x_{m+1}^k$
    - update inner iteration index: $j_i = j_i + 1$

    enddo

  enddo

Figure 25: The two-level Newton method

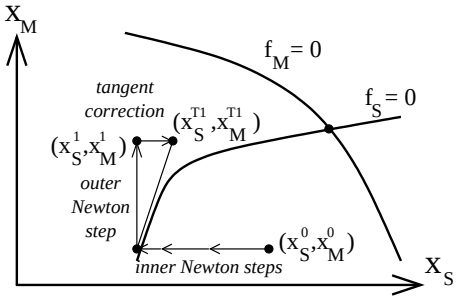


Figure 26: Two-level Newton scheme for solving $f_S(x_S, x_M) = 0$, $f_M(x_S, x_M) = 0$. Index S: inner Newton, on subsystem; index M: outer Newton, on master system

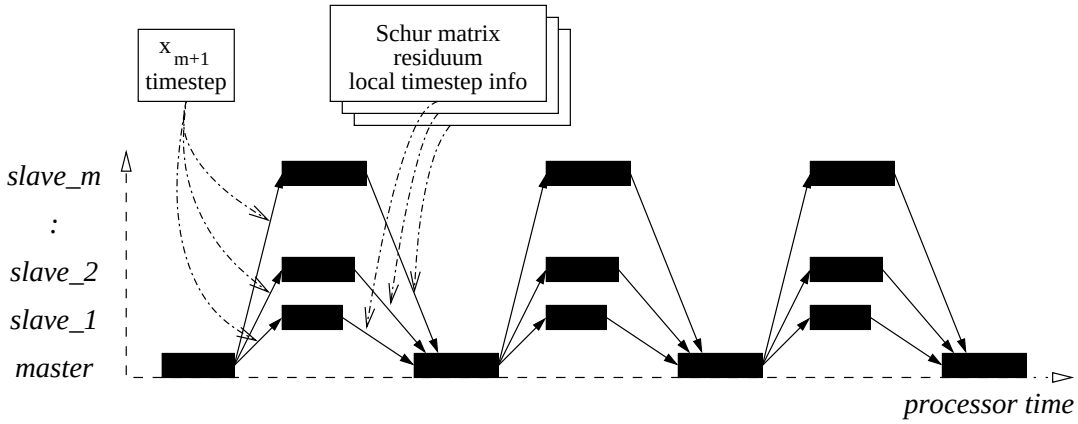


Figure 27: Ideal data flow for parallel MLN.

In [116, 117] a variant is described in which the Schur complement actions and the tangent corrective actions are replaced by introducing a simple global Newton-Raphson step as outerloop action: in fact their method is Newton-Raphson in which each iteration is solved by m-parallel Newton-Raphson sub-processes, each of at most J-iterations (with $J \leq 5$) for the subsystems.

Parallel multi-level Newton: Loop over hierarchies

If the circuit is decomposed into a set of subblocks which are interconnected via a carrier network, then the MLN method described in Fig. 25 is a natural choice for parallelization at a coarse grain level: For each timestep, each subblock is solved in an inner Newton loop by a slave process on a separate processor unit, and then the master process performs an outer iteration for getting the carrier network solution. An ideal data flow for parallel MLN is shown in Fig. 27: The master sends the values of the carrier circuit variables x_{m+1} and the actual timestep to the slaves; then each slave performs its tangent correction, solves its subblock equations, computes Schur complement S_i, residuum R_i and timestep control information, and sends all back to the master. The black boxes in Fig. 27 indicate when the particular process is busy. Of course, the relative time for data transmission should be much smaller than suggested in Fig. 27.

Decoupling. If the branch currents flowing from the slave subnets into the master carrier circuit are introduced as additional variables in the vector of unknowns, then the network equations are well decoupled, and the term $\partial f_{m+1}/\partial x_i$ arising in the Schur matrix

$$S_i = \frac{\partial f_{m+1}}{\partial x_i} \cdot \left(\frac{\partial f_i}{\partial x_i} \right)^{-1} \cdot \frac{\partial f_i}{\partial x_{m+1}}$$

and the residuum

$$R_i = \frac{\partial f_{m+1}}{\partial x_i} \cdot \left(\frac{\partial f_i}{\partial x_i} \right)^{-1} \cdot f_i$$

is simply a constant incidence matrix. Before showing more details, we make a further extension of the vector of unknowns: The pin voltages of the slave subblocks at the border to the master circuit are duplicated, and the new voltages are assigned to the slave subnets [77]. This is not necessary in principle for efficient parallelization; however there are two advantages:

- Circuits can be easily decoupled using controlled sources, as shown in Fig. 28 [262, 263]. Since the latter are standard elements in any circuit simulator, no extra programming efforts are necessary for decoupling. [7]

[7] Decoupling by imposing pin voltages to the subblocks — as illustrated in Fig. 28 — is called "node tearing" in the literature [214], and is mostly applied when simulating integrated circuits. This is surely adequate from a numerical aspect, as long as the circuit is voltage driven, i.e. its functionality is described in terms of voltage

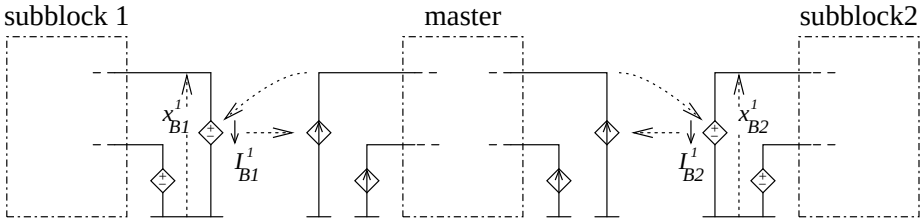


Figure 28: Circuit decoupling with controlled sources.
Pin currents I_{Bi} and pin voltages x_{Bi} are introduced for each subblock i.

- Numerical robustness of the MLN method can be improved by applying particular damping strategies for the controlled voltages sources, which drive the pins of the slave subcircuit blocks [251].

With these extensions, the vectors x_i and functions f_i, f_{m+1} of the standard MLN scheme (Fig. 25) have to be replaced by

$$x_i \Rightarrow \begin{pmatrix} x_{Si} \\ x_{Bi} \\ I_{Bi} \end{pmatrix}, \qquad f_i(x_i, x_{m+1}) \;\Rightarrow\; \begin{pmatrix} f_{Si}(x_{Si}, x_{Bi}) \\ f_{Bi}(x_{Si}, x_{Bi}) + I_{Bi} \\ x_{Bi} - A_i \cdot x_{m+1} \end{pmatrix} \quad (i = 1, \ldots, m)$$

$$f_{m+1}(x_i, x_{m+1}) \;\Rightarrow\; f_{m+1}(x_{m+1}) - \sum_{i=1}^{m} A_i^T \cdot I_{Bi},$$

where the variables $x_{Si} \in \mathbb{R}^{n_{Si}}$, $x_{Bi} \in \mathbb{R}^{n_{Bi}}$, $I_{Bi} \in \mathbb{R}^{n_{Bi}}$ and $x_{m+1} \in \mathbb{R}^{n_{m+1}}$ denote the inner network variables of subblock i, the pin voltages of subblock i, the pin currents leaving subblock i and the network variables of the master circuit, respectively. $A_i \in \{0,1\}^{n_{Bi} \times n_{m+1}}$ are incidence matrices, projecting the nodes of the master system to the pin nodes of subblock i. Hereby are m the number of subblocks (slaves), n_{Si} and n_{Bi} the number of inner network variables and pins of subblock i, and n_{m+1} the dimension of the master system.

The network equations can be characterized as follows:

— $f_{Si} : \mathbb{R}^{n_{Si}} \times \mathbb{R}^{n_{Bi}} \to \mathbb{R}^{n_{Si}}$ are the inner network equations of subblock i;

— $f_{Bi} : \mathbb{R}^{n_{Si}} \times \mathbb{R}^{n_{Bi}} \to \mathbb{R}^{n_{Bi}}$ capture the currents flowing from the inner nodes of subblock i into its pins;

— $f_{m+1} : \mathbb{R}^{n_{m+1}} \to \mathbb{R}^{n_{m+1}}$ are the network equations of the master circuit without the slave contributions;

Consequently, the Jacobians $\partial f_i/\partial x_i$, $\partial f_i/\partial x_{m+1}$, and $\partial f_{m+1}/\partial x_i$ of the MLN scheme given in Fig. 25 have to be replaced by

$$\frac{\partial f_i}{\partial x_i} \Rightarrow J_i := \begin{pmatrix} \dfrac{\partial f_{Si}}{\partial x_{Si}} & \dfrac{\partial f_{Si}}{\partial x_{Bi}} & 0 \\ \dfrac{\partial f_{Bi}}{\partial x_{Si}} & \dfrac{\partial f_{Bi}}{\partial x_{Bi}} & I \\ 0 & I & 0 \end{pmatrix}, \qquad \frac{\partial f_i}{\partial x_{m+1}} \Rightarrow \begin{pmatrix} 0 \\ 0 \\ -A_i \end{pmatrix}, \qquad \frac{\partial f_{m+1}}{\partial x_i} \Rightarrow \begin{pmatrix} 0 & 0 & -A_i^T \end{pmatrix},$$

where I is a $n_{Bi} \times n_{Bi}$ unity matrix. With this decoupling, we get the following form for the Schur complement:

$$S_i = A_i^T \cdot \left(\frac{\partial f_{Bi}}{\partial x_{Si}} \cdot \left(\frac{\partial f_{Si}}{\partial x_{Si}} \right)^{-1} \cdot \frac{\partial f_{Si}}{\partial x_{Bi}} - \frac{\partial f_{Bi}}{\partial x_{Bi}} \right) \cdot A_i \qquad (i = 1, \ldots m). \tag{15.1}$$

waveforms; an example is standard CMOS logic. For current driven circuits like some analog building blocks or power circuits with switches, "branch tearing" may be preferable [108]. In this case the subcircuit pins are driven by controlled current sources, and the pin voltages are fed back into the master circuit.

The second factor can be shown to be just $\partial I_{Bi}/\partial x_{Bi}$, if the inner system is solved exactly. Since the latter is not possible in general, we conclude that the Schur matrix is a more or less good approximation for the admittance matrix of the particular subblock:

$$S_i \approx A_i^T \cdot \frac{\partial I_{Bi}}{\partial x_{Bi}} \cdot A_i = A_i^T \cdot \frac{\partial I_{Bi}}{\partial x_{m+1}}.$$

For its numerical computation we can use equation (15.1), which requires to calculate the inverse of $\partial f_{Si}/\partial x_{Si}$. This may become expensive since n_{Si} is large in general, and so it is more economic to exploit that the Schur matrix is just the lower right part of J_i^{-1}, where J_i is the inner iteration matrix:

$$J_i^{-1} = \begin{pmatrix} \left(\frac{\partial f_{Si}}{\partial x_{Si}}\right)^{-1} & 0 & -\left(\frac{\partial f_{Si}}{\partial x_{Si}}\right)^{-1} \frac{\partial f_{Si}}{\partial x_{Bi}} \\ 0 & 0 & I \\ -\frac{\partial f_{Bi}}{\partial x_{Si}} \left(\frac{\partial f_{Si}}{\partial x_{Si}}\right)^{-1} & I & \frac{\partial f_{Bi}}{\partial x_{Si}} \left(\frac{\partial f_{Si}}{\partial x_{Si}}\right)^{-1} \frac{\partial f_{Si}}{\partial x_{Bi}} - \frac{\partial f_{Bi}}{\partial x_{Bi}} \end{pmatrix}$$

This submatrix can be computed columnwise, using the original system from the inner Newton process

$$J_i \cdot \begin{pmatrix} \triangle x_{Si} \\ \triangle x_{Bi} \\ \triangle I_{Bi} \end{pmatrix} = - \begin{pmatrix} f_{Si} \\ f_{Bi} + I_{Bi} \\ x_{Bi} - A_i \cdot x_{m+1} \end{pmatrix},$$

but taking different right hand sides: Solve

$$J_i \cdot \begin{pmatrix} \dots \\ \dots \\ s_i^k \end{pmatrix} = - \begin{pmatrix} 0 \\ 0 \\ e_i^k \end{pmatrix} \quad (k = 1, \dots n_{Bi})$$

for s_i^k, where e_i^k is the k-th column of an $n_{Bi} \times n_{Bi}$ dimensional unity matrix. This requires one LU decomposition of J_i and only n_{Bi} forward backward substitutions, which can be done locally on each slave processor. The s_i^k then form the columns of the Schur matrix, which has to be transferred to the master processor for being assembled into the outer iteration matrix.

If the inner Newton loops are truncated after some iterations then there remains a defect of the subblock equations which enters the outer Newton process in form of the residuum R_i, see Fig. 25. In the decoupled formulation we get:

$$\begin{aligned} R_i &= -A_i^T \cdot \begin{pmatrix} -\frac{\partial f_{Bi}}{\partial x_{Si}} \left(\frac{\partial f_{Si}}{\partial x_{Si}}\right)^{-1} & I & -\frac{\partial f_{Bi}}{\partial x_{Si}} \left(\frac{\partial f_{Si}}{\partial x_{Si}}\right)^{-1} \frac{\partial f_{Si}}{\partial x_{Bi}} - \frac{\partial f_{Bi}}{\partial x_{Bi}} \end{pmatrix} \cdot \\ &\quad \cdot \begin{pmatrix} f_{Si} \\ f_{Bi} + I_{Bi} \\ x_{Bi} - A_i \cdot x_{m+1} \end{pmatrix} \\ &= A_i^T \cdot \triangle I_{Bi} \qquad (i = 1, \dots m) \end{aligned}$$

Here is $\triangle I_{Bi}$ an error term for the pin currents which is induced from truncating the inner Newton loop, and which can be computed from solving

$$J_i \cdot \begin{pmatrix} \dots \\ \dots \\ \triangle I_{Bi} \end{pmatrix} = - \begin{pmatrix} f_{Si} \\ f_{Bi} + I_{Bi} \\ x_{Bi} - A_i \cdot x_{m+1} \end{pmatrix}.$$

This can be done locally on each slave processor, and after being transferred to the master processor, the $\triangle I_{Bi}$ must be assembled into the right hand side for the next outer Newton step.

Finally the tangent correction has to be computed, before the next inner Newton loop is started. It is easy to see that this can be done locally either, as soon as the actual state of the master variables x_{m+1} is available on the slave processors.

Some remarks can be given on how to improve parallel performance:

- For the master circuit the linear solver is the most time critical part. Since the Schur complements tend to be dense, a sparse block or even dense linear solver is adequate.
- We see from Fig. 27 that the master process is idle when the slaves are working, and vice versa. So, one slave process can be assigned to the master processor [8]; and on a shared memory machine a *parallel* linear solver utility should be used, which includes the slave processors for solving the large interconnect network of the master.
- A performance model for parallel MLN is described in [88]. It can be used for dynamically adopting numerical parameters of the MLN method — like the maximal number of inner iterations — to the computer- and network-configuration and its actual load.

Partitioning. In an industrial environment an automatic tool is indispensable, which — by applying tearing methods — partitions a given circuit netlist into subnets, inserts the controlled sources for proper decoupling, and assigns the subnets for being solved in a particular process (→ *static assignment*). The number of partitions is prescribed by the user. Hereby two different strategies can be applied: One makes the number of partitions just equal to the number of processors, which are actually available. In this case the partitions have to be equally balanced with respect to their computational load, and latency effects — i.e. the different rate of convergence for different partitions — cannot be exploited. The other strategy makes the number of partitions much larger than the number of processors, and assigns several subnets to one processor. This alleviates the needs to get partions of equal workload, and opens a chance to exploit latency effects. Unfortunately, in real life applications it turned out, that a large number of partitions tends to increase the interconnect network significantly, which has to be solved in the master process [52]. This is a critical issue for the performance of MLN, and so in practice a small number of partitions is often more efficient than a large one.

The most essential requirements for the partitioner are [77, 251]:

1. prescribed number of partitions;
2. small number of interconnects between each partition and master;
3. small total number of interconnects;
4. equal computational weight (workload) for each partition;
5. solvability for each partition and carrier network;
6. small runtime;
7. all nonlinear elements put into partitions;

The second requirement is for keeping the dimension n_{Bi} of the Schur complements small, which is desirable for reducing both the expense to calculate the Schur matrix and the amount of data to be sent to the master. The third requirement is extremely important for the workload of the master process since the interconnect net mainly determines the dimension of the master system, which is *not* sparse in general. The 4^{th} requirement takes care of a well balanced load of the slave processors, which is essential if each processor gets just one partition. The 5^{th} requirement concerns that unsolvable circuit structures — like loops of inductors and/or voltage sources discussed in Section 7 — may be generated due to insertion of controlled sources at the partition borders, which must be avoided. The runtime requirement is obvious, but hard to meet since partitioning problems are NP complete [80, 108]. Hence heuristics have to be found which produce near optimal solutions in a short time. Finally, the last requirement is desirable to shift nonlinearity into the slaves, thus reducing the number of expensive outer iterations.

[8] More precisely: The master can start collecting data as soon as the fastest slave has finished its task; so the slave task with the smallest workload should be assigned to the master processor.

In Section 5 it was shown that decomposition may have an impact on the DAE index of the system. This is not a critical issue as long as the index does not get greater than 2 for the decomposed system, since state of the art integrators usually can cope with index-2 problems. However, in case of DAE index > 2 most integrators may run into severe numerical problems, and to avoid that should be a further requirement for the partitioner. Unfortunately, it may be difficult to find out for a certain circuit configuration if insertion of controlled sources would raise the index beyond 2, see [67]. So this requirement is not (yet) observed in any partioning tool.

We will now shortly consider partitioning algorithms as far as they are suited for coarse grain parallelization of classical circuit simulation. Special partitioners for waveform relaxation or for placement tools, e.g., are not included here, since their objectives are different.

Since partitioning is closely related with the task to transform a given matrix into bordered block diagonal form [108], the methods suggested for the latter problem may be useful for partioning as well. One of them is nested dissection. In its original form it provides partitions of decreasing size [108]; it has to be checked if variants like that in [85] provide better balanced partitions [193].

In [214] a *local* clustering algorithm was proposed, which turned out in practice to be efficient and to generate partitions of almost equal size. It starts from a vertex of a weighted network graph, and adds that neighbour vertex to the cluster, which provides the minimal number of edges to the vertices outside the cluster. If the cluster size is somewhat beyond its "optimal" value, then a backtracking step takes care that the number of edges crossing the border becomes minimal within a certain interval. This cluster is selected as a first partition, and the cluster process is restarted. The computational complexity of this method is $\mathcal{O}(n_V^2)$, where n_V is the number of vertices [214].

A more accurate weight model for the computational cost of a partition is suggested in [251]. To this end, each circuit element is given a specific weight – depending on its computational complexity – and the cost of a partition is just the sum of the weights of its elements, plus a certain weight for each of its nodes, since the latter gives rise to one more circuit equation.

This model requires to formulate the cluster problem on hypergraphs which are weighted on both vertices and edges. It is however possible to extend the local clustering algorithm of [214] properly, such that partitions with very well balanced computational cost can be generated even for large industrial designs in a reasonable time [251].

Partitions with even smaller numbers of interconnects can be expected from algorithms with a more *global* view. These operate usually in two steps: The first step provides an initial partitioning under global aspects, in order to meet the requirements 3 and 4. For this purpose bisection methods [41], analytical placement [77], or simply the hierarchy of the network description are used. In the second step the cut cost of each partition is reduced ($\rightarrow$ requirement 2) by shifting circuit nodes or branches between partitions. This is done using Fiduccia-Mattheyses like methods [41, 180], minimizing ratio-cut [77], or with some other heuristics [132].

Global partioning methods often suffer from prohibitive runtimes when being applied to very large problems. As an alternative, a clustering algorithm was recently developed, which keeps global aspects in mind *and* aims at a very high computational efficiency [78]. Basically it forms clusters by merging adjacent vertices (circuit elements) of an edge (circuit node) in a weighted modified network graph. For clustering, the simple edge weight criterion is replaced by a more sophisticated coupling measure, which takes care that adjacent vertices with only a few edges are preferred for clustering, and that the cluster size is well balanced. For each merging step the whole circuit is inspected; this brings the global aspects into account and finally enables excellent partitioning results in a reasonable time, as is demonstrated with a large number of actual designs from industry [79]. The method has a complexity of $\mathcal{O}(n_R \cdot n_V \cdot log(n_R \cdot n_V))$, where n_R is an average number of edges per vertex, and n_V is the number of vertices.

Dynamic assignment techniques were explored in an experimental paper [41], in order to check how far latency effects can be exploited, and which maximal degree of parallelism can be obtained. To this end the number of partitions was made quite large, and by using partial

No of processors	Speedup
4	$2.5 \ldots 3$
8	$3.5 \ldots 5$
12	$5 \ldots 7$
16	> 7

Table 9: Typical speedups with parallel MLN versus serial standard Newton.

LU factorization and dynamic subtask allocation, $\approx 95\%$ of the total job could be executed in parallel, which is hard to obtain with static assignment. As a conclusion, dynamic resource allocation was recommended on shared memory machines, while static assignment fits better to clusters with distributed memory and slow interconnect network. It would be interesting to check if these results still hold for very large actual designs. Furthermore, it should be noted that dynamic allocation schemes require considerable programming efforts for the simulator itself, while partitioning requirements are less ambitious.

Results. The speedup obtainable with this kind of parallelization depends primarily on the circuit structure and on the quality of partitioning. In the best cases — with partitions providing equal workload and a small number of pins — almost linear speedups were reported, e. g. a factor 7.79 on 8 CPU's [77]. Sometimes even superlinear speedups can be observed, which is due to the shift of nonlinearity into smaller circuit blocks, or due to reduced memory needs. In the worst case of unbalanced partitions and large interconnect to the master circuit, the speedup may not be much larger than 1. Fortunately, the user can see in advance whether it makes sense to start parallel simulation with a given partitioning.

Typical speedups for real life applications are given in Table 9 [77, 78, 251, 258]. Note that it often does not make much difference if the problem is run on a shared memory multiprocessor system or on a cluster of processors with relatively slow interconnect network. The latter aspect is of commercial interest, since it allows to set up a very cheap cluster of fast PCs for running large circuit simulations efficiently.

A reasonable number of processors is actually between 4 and 16. Each processor should have a fairly large load, since otherwise there is a risk to get a poor ratio of interconnect to partition size, making parallelization inefficient. Therefore scalability is limited: Increasing speedups can only be expected for an increasing number of processors, if the problem size is increasing as well [258].

The runtime of an advanced partioning tool is $1 \ldots 10$ min for circuits containing $15k \ldots 150k$ transistors. This makes its possible to run several partitioning trials with different options, and to select the best partition found for performing the analysis.

Even more important than exact speedups is the chance to handle problems of a size which is almost one order of magnitude larger than with serial simulation. An actual example is a $500k$ transistor circuit including parasitics, which can be simulated over night on a 12 processor machine, giving full confidence in its functionality to the designer [52].

Note that the results reported here were obtained with a fully implicit integration method like BDF or TR-BDF. Semiimplicit numerical integration schemes — like the ROW method — do not require a parallel MLN method. However they can utilize the multi-level linear solver [250], which is naturally included in MLN, and so parallelization of the ROW method is achieved at almost no extra cost if a parallel MLN solver is available for fully implicit integration rules. Even partitioning is not affected by the particular choice for numerical integration.

Thread based parallelization: Loop over processes (threads)

This kind of parallelization is targeted for multiprocessor systems with a large shared memory. Hence interchange of data between processes is not of major concern. However cache effects are

of great importance, and so it is essential to take care of data locality. We restrict on systems with a limited number of processors, as are commonly used in industrial environments.

From Table 7 we see that parallelization should focus on the load and on the linear solver part of a circuit simulator. Parallelization of the rest either does not impose any difficulties, or does not make sense due to ist serial character and small runtimes.

Load. The load part consists of three tasks:

1. Evaluation of the device characteristics and their derivatives: This is an expensive, but perfectly parallelizable task, since all evaluations are independent from each other. Furthermore there are always many elements of the same kind (transistors, capacitors, ...); hence load balancing is trivial.

2. Numerical integration in case of BDF like methods; this is easy to parallelize as well.

3. Stamping, i.e. adding the element contributions to matrix and right hand side: This is some kind of protected operation, since different elements may stamp into the same entry of matrix or right hand side. Nevertheless, parallelization is necessary, since otherwise parallelization effects saturate already for $4 \ldots 6$ processors [208]. Possible solutions are [58, 208]:

- Edge colouring techniques can be applied: Circuit elements of the same kind sharing the same node are marked with different colours. Then all elements with identical colour can be stamped in parallel.
- The circuit is partitioned into equally sized subblocks with minimal number of interconnects between them. Elements of different subblocks not being at the border then can be stamped in parallel; the border elements can be stamped blockwise sequentially, partition per partition.
- The *stamp-into* operation is replaced by a *fetch-from* operation: Each entry of matrix and right hand side knows from which element it gets a contribution, and fetches it from there. All entries can act in parallel. If the number of elements connected to one node is very unbalanced, then some refinements of this method may be useful, e. g., the generation of subtasks.

In sum, parallelization of the load part can be done very efficiently, and good scalability can be expected.

Linear solver. The focus is here on parallel LU factorization, since forward backward substitution is far less expensive and easier to parallelize [74]. Two main directions are pursued:

Tearing: The first approach is to partition the circuit with tearing methods into well balanced subblocks, and LU factorize the subblocks in parallel [41, 208, 249]. This technique is closely related to the parallel multi-level Newton method described above. So we will not further discuss it here.

Clustering Gauss operations: The second approach is to cluster the Gauss operations of sparse LU factorization into sets of independent tasks, and perform all tasks of a set in parallel. These concepts are rapidly evolving at present due to their importance in a much more general framework. On overview can be found in Chapter 9 of this Handbook; an actual code is described in [216]. Note that in this framework the notation of parallel granularity (see, e. g.,[113]) is different from what we have introduced at the beginning of this section.

We end with some comments on the second approach which directly concern circuit simulation aspects.

Mixed direct / iterative linear solver: Parallelization of a mixed sparse direct / iterative linear solver was recently suggested in an interesting alternative, which directly aims at circuit simulation problems [14], see Chapter 9 for details.

Adaptive partitioning: In the course of a transient analysis the linear solver is called quite often with an identical zero/non zero pattern of the matrix. So it may be worthwhile to provide

Aspect	Thread based	Multi-level Newton
hardware	shared memory multiprocessor	— cluster of workstations or PCs — shared memory multiprocessor
hardware cost	moderate	cheap ... moderate
memory needs	large	small (eventually large for partitioner)
communication overhead	large	small
user handling	easy ... moderate	easy ... moderate ... difficult (depending on partitioner)
data flow	simple	complex on workstation cluster: — scatter partitions to slave processors — gather/merge resulting waveforms — restart and error management
scalability — small problems — large problems	 good good ... moderate	 no moderate
spatial adaptivity (exploitation of latency)	low	— static assignment: low ... moderate — dynamic load balancing: high
algorithmic overhead	small	— Schur complement — interconnect solver
algorithmic benefit	none	shift nonlinearity into subsystem
algorithmic challenge	partitioner: split Gauss operations	partitioner: split circuit
programming effort	moderate	— static assignment: low ... moderate — dynamic load balancing: high

Table 10: Aspects of parallelization.

some learning phase in the algorithms, where partioning is adapted until optimal speedups are obtained. In [74] such an adaptive partioning method is described, which does not only provide significant speedup improvements, but also requires less CPU time than a conventional partitioner.

Ordering: The sparse LU factorization needs preordering of the matrix for minimal generation of fillins. Circuit simulation codes mostly employ the Markowitz method [160], which is a variant of the minimum degree algorithms [55]. For parallel LU factorization other methods may be better suited ([14, 193]), although first experiments with nested dissection methods were not successful yet [74].

Thread based parallelization and parallel multi-level Newton MLN – a conclusion. Both schemes are realized in codes, which are used since some time in industrial environments, there is no direct comparison available at present. As a first step, we try to compare them in Table 10 with respect to the most important aspects of parallelization, without giving numbers or assessments. If run on a shared memory system, MLN may be somewhat less efficient than a dedicated thread based version. The merits of MLN are an excellent performance / cost ratio and its flexibility, which even makes distributed simulation via Internet possible. One surprising fact is, that the most critical issue in both approaches is the partitioner, even if its objectives are somewhat different.

Hierarchical Simulation

Very often the design of an electronic circuit is characterized by a hierarchical organization of models and submodels with as final leaves active and passive components. Compact transistor models (devices) are treated as building blocks in the modular design of the circuit. Submodels and devices are linked to the hierarchy by their terminal unknowns to the enclosing model. A device is a leave of the tree. The top model is the circuit-level, which has only one terminal, the ground node (which voltage is set to 0). Models and devices may have their own internal unknowns. The behaviour of the solution of of a model at all nodes is completely determined by the values at the terminals (together with the internal sources) and the non-linear interaction with its containing submodels and devices.
A hierarchical organized algorithm allows a datastructure of the circuit that is very close to the original design [187, 234, 254, 255]. A hierarchical formulation corresponds to a particular block partitioning of the problem that allows for parallelism in a natural way [16]. Even in a sequential approach particular algorithms can be pursued, starting with the observation that the overall matrix and the solution can be distributed over all hierarchical levels. Algorithms are usually defined recursively (see algorithm in Fig. 29) . Depending on actions being done before or after the recursions and passing data to or lifting data from a submodel, or device, one can speak of Top-Down and of Bottom-Up recursions.

for all $i = 0, \ldots, I-1$ (Time step iteration) **do**
  for $k = 0, \ldots, K-1$ (Newton iteration) **do**
    Recursion I: Bottom-Up Matrix Assembly $[A\mathbf{x}^{n+1} = \mathbf{b} \equiv -\mathbf{F}(\mathbf{x}^n) + A\mathbf{x}^n]$
    and Decomposition [$A = UL$, $L\mathbf{x}^{n+1} = \mathbf{c} \equiv U^{-1}\mathbf{b}$]
    Recursion II: Top-Down Linear Solution [$\mathbf{x}^{n+1} = L^{-1}\mathbf{c}$]
    Recursion III: Bottom-Up error estimation
  end for
  Recursion III: Bottom-Up discretization-error estimation
end for

Figure 29: The Main Hierarchical Recursions

In the algorithm above Gaussian elimination is used because it nicely fits a hierarchical algorithm. This is in contrast to several iterative linear solvers in which needed preconditioners disturb the assumed block structure (however, for a collection of some recent results by some hierarchical-friendly methods, see [217]).

Bypass mechanisms Each hierarchical branch normally depends continuously on the values of the terminals at some top-level, in addition to values of internal sources and values of time-derivatives. This allows for several forms of *bypassing* where we satisfy ourselves not to update results obtained previously in some part of a process.

- *Newton-level bypassing:* In a Newton-Raphson iteration one can decide to bypass a complete hierarchical branch starting from submodel S when its terminals do not change that much.

$$\mathbf{x}_{S,j}^{n+1} \quad = \quad \mathbf{x}_{S,j}^{n} \quad \text{if} \quad ||\mathbf{x}_{M,i}^{n+1} - \mathbf{x}_{M,i}^{n}|| \; < \; \varepsilon \tag{15.2}$$

  where S denotes a submodel or device, and M the encompassing model. At this highest level i ranges from 1 to n_t^M (number of terminal unknowns of S at level M). At each sublevel S, j ranges from 1 to $n_t^S + n_i^S$ (where n_t^S, n_i^S are the number of terminal and internal unknowns at level S, respectively).
  Clearly, by this one can re-use matrix-contributions and right-hand side contributions from all submodels of which the tree starts at model M. Depending on the type of linear solver one also can re-use the local LU-decompositions.

- *Transient step bypassing:* The bypass approach may be extended to a transient step, when the extrapolated values (or the result of the predictor) indicate results close to the final one at the previous time level.

- *Cross-tree bypassing:* The above bypass approaches are examples of *in-tree bypassing*: one can only bypass a remainder of the hierarchical branch by comparing it to a previous approximation to the solution of the same branch.
  A generalization to this might be called *cross-tree bypassing*. For this one identifies branches that formally have identical datastructures, for instance because each branch starts at different occurrences of a same model or device definition.
  When one has determined the solution of one branch, its results may be copied to the other branch (when needed, one might postpone this). Note that this applies to the Newton as well as to the Transient step level as well (in Transient analysis one might also apply cross-tree bypassing during the stepsize determination).
  The HSIM simulator, developed by Nassda Corporation [127], exploits this type of bypassing. It efficiently stores repeated instances of the same subcircuit, providing by this "unlimited" design capacity (compared to flat-based simulators). It takes advantage of hierarchical structures operating under the same conditions in the design to dynamically reuse computed results [252]. In mixed-signal analysis the bulk of the circuit is of a digital nature and only a minor part is a true analog part. In the digital part, a lot of branch matchings may occur, and also their boundary (terminal) values may be identical (in fact because of the digital nature, only very few different stages will be possible). Good speed up are reported when compared to conventional analog circuit simulators. In addition to bypassing, a hierarchical RC reduction algorithm compresses parasitic resistances and capacitances in the hierarchical database. Finally, a coupling decomposition algorithm efficiently models submodel couplings, such as crosstalk noises, as submodel interface currents and conductances.

16 Multirate integration

From our CMOS ringoscillator example in Section 13 we have seen that multirate integration offers significant speedup potential for circuit simulation. Waveform relaxation WR exploits the multirate behaviour in a very natural way: Subblocks are decoupled, and each of them can be integrated with its own local timestep. In standard circuit simulation however the subsystems are not decoupled. So when we solve a certain circuit part at a particular timepoint, we need information about the contribution of all other circuit parts, which is not available due to their different integration stepsize.

To get accurate, controllable, and cheap to compute estimates for these contributions has been a key problem in multirate integration.

Let us for the sake of simplicity assume that the circuit at a timepoint t can be separated into an active part x_A — which has to be integrated with a small timestep h — and a less active ("latent") part x_L, being integrated with a much larger timestep $H >> h$. Then there are two different strategies to compute a solution in the time between between t and $t+H$ [82, 225]:

- **Fastest first:** Integrate x_A with small stepsize h from t to $t+H$, using extrapolated values for x_L;
  then perform one integration step for x_L from t to $t+H$, taking interpolated values from x_A, if necessary.

- **Slowest first:** Integrate x_L with one step of size H, where $x_A(t+H)$ is extrapolated;
  then integrate x_A with small stepsizes h, taking interpolated values from x_L.

Both approaches are pursued in different implementations. While the first one seems to be straightforward — since it relies on the assumption that the slowly varying variables can be well extrapolated into future — offers the second computational advantages.

Roughly two directions can be recognized in the literature: One tries to extend standard circuit simulation techniques using multi-step integration methods; the second is oriented towards one-step methods. Both of them have in common, that for reducing overhead the circuit is partitioned into subblocks, each of which is handled with its own local timestep.

- *Initialization:*
  - partition circuit into subblocks
  - compute initial values
  - setup and initialize event list
- *Transient simulation:*
  for each entry of event list do
  - mark subblocks to be active or latent
  - *Newton loop:*
    until convergence do
    - load matrix and right hand side for active subblocks as usual
    - load matrix and right hand side for substitute circuits of latent subblocks
    - solve reduced linear system
    - add Newton correction to active part of circuit variables
    - check convergence
  - *timestep control and verification step:*
    for all active subblocks do
    - perform timestep control

    enddo
    for all latent subblocks do
    - check latency assumption

    enddo
    update event list

  enddo

Figure 30: Event controlled multi-rate integration with multi-step methods

Multi-step methods. All multi-step methods known so far employ the fastest first principle, and make use of some kind of event control, see Fig. 30: Based on conventional timestep control, each subblock computes its next timepoint and puts it into an event list. The global timestep h is determined from the next entry of this event list, and depending on their own local stepsize h_i, the subblocks are marked to be active — if h_i is not much larger than h — or to be latent else. The active subblocks are evaluated in a conventional way, but the latent subblocks are replaced by some simple substitutes, which aim at extrapolating their terminal behaviour. With these substitutes, the reduced system is solved, and after getting convergence the latency assumption has to be verified a posteriori. The latter step is important to maintain the reliability of the standard algorithm. It may give rise to roll back the simulation for several timesteps, which is very critical since it degrades performance significantly.

Alternatives for the substitute circuits are shown in Fig. 31. In a) the value of R is fixed, and V is determined such that the extrapolated values for both pin current and voltage are consistent. b) is just the Norton equivalent of the subblock at the pin node, i.e. $G = 1/R$ and C are its static and dynamic entry in the Jacobian, and I is the right hand side entry from the last iteration in active mode. Another approach suggests independent sources with extrapolated values for pin currents or voltages [209].

The speedup potential for this kind of multirate integration is mainly determined by savings of expensive device evaluations for the latent parts, and by solving a smaller system of equations. On the other hand there is slowdown due to roll back steps and due to overhead of event control. Overall speedup factors $2 \ldots 20$ have been reported [40, 70, 209], but obviously methods and

codes are not yet mature enough to be used in standard industrial environments.

One-step methods. Multirate one-step schemes so far have aimed at systems where the whole dynamics can be described by an initial value problem of ordinary differential equations

$$\dot{y} = f(t, y), \qquad y(t_0) = y_0, \qquad y \in \mathbb{R}^n. \tag{16.1}$$

For the sake of simplicity, we concentrate our investigations on autonomous initial value problems, whose state vector $y \in \mathbb{R}^n$ is partitioned into only two parts: Latent components $y_L \in \mathbb{R}^{n_L}$ and a small number of active components $y_A \in \mathbb{R}^{n_A}$ with $n_A + n_L = n$ and $n_A \ll n_L$,

$$\dot{y}_A = f_A(y_A, y_L), \qquad y_A(t_0) = y_{A0}, \tag{16.2a}$$
$$\dot{y}_L = f_L(y_A, y_L), \qquad y_L(t_0) = y_{L0}. \tag{16.2b}$$

The active components y_A are integrated with a small stepsize h, the latent components y_L with a large stepsize $H = mh$. The realisation of multirate one-step schemes now depends not only on the underlying numerical scheme, but also on which part is integrated first and, crucially, how the coupling is done.

One origin of these method are the split Runge-Kutta schemes by Rice [197]. Multirate extrapolation [61] and Runge-Kutta [62] schemes are successfully used in stellar problems by Engstler and Lubich. In Günther and Rentrop [91, 92] multirate Rosenbrock-Wanner (MROW) methods are used for VLSI applications of electrical networks. One shortcoming of all these multirate methods derived so far is the coupling between active and latent components by interpolating and extrapolating state variables, which inevitably decompose the underlying one-step method in a two-step procedure, and thus makes their implementation very difficult into existing simulation packages. Recently, a new answer on how to realize the coupling, was given by Kværnø/Rentrop [145] for explicit Runge-Kutta schemes: The internal stages are used to compute the coupling terms, too. Meanwhile, this so-called *generalized multirate* approach was extended to implicit schemes, e.g. ROW- and W-methods, to manage also stiff problems as arise in network analysis. One should note that the coefficients of all these one-step schemes can be chosen such that one gets stable methods of any prescribed order of convergence.

We start to give the outline of a somewhat generic generalized multirate one-step method: The approximate solution $y_L^H(t_0 + H)$ of y_L at time point $t_0 + H$ and $y_A^h\big(t_0 + (\lambda + 1)\, h\big)$ of y_A at time points $t_0 + (\lambda + 1)\, h$ are given by

$$y_L^H\big(t_0 + H\big) = y_{L,0} + \sum_{i=1}^{s} \tilde{b}_i k_{L,i},$$

$$y_A^h\big(t_0 + (\lambda + 1)\, h\big) = y_A^h\big(t_0 + \lambda\, h\big) + \sum_{i=1}^{i} b_i k_{A,i}^{\lambda} \qquad (\lambda = 0, \ldots, m-1),$$

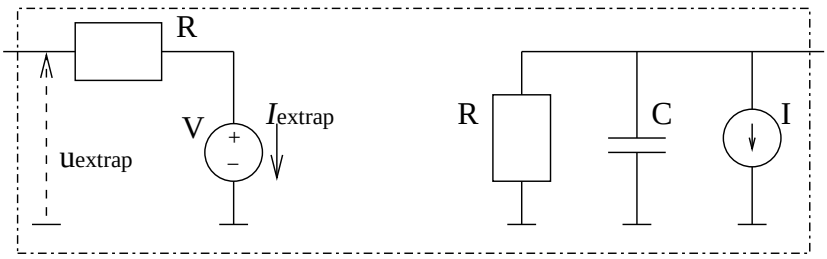


Figure 31: Substitute circuits at the pins of latent subblocks
a) Reference [70] (left), b) Reference [40] (right).

where the increments are computed via the linear systems

$$k_{L,i} = H\, f_L\Big(y_{L,0} + \sum_{j=1}^{i-1} \tilde{\alpha}_{ij} k_{L,j},\ \boxed{\bar{Y}_{A,i}}\Big) + H\ \frac{\partial f_L}{\partial y_L}\Big|_{y_0} \sum_{j=1}^{i} \tilde{\gamma}_{ij} k_{L,j} + m\, H\ \frac{\partial f_L}{\partial y_A}\Big|_{y_0} \sum_{i=1}^{i} \tilde{\nu}_{ij} k^0_{A,j},$$

$$k^\lambda_{A,i} = h\, f_A\Big(\boxed{\bar{Y}^\lambda_{L,i}}, y_{A,\lambda} + \sum_{j=1}^{i-1} \alpha_{ij} k^\lambda_{A,j}\Big) + h\ \frac{\partial f_A}{\partial y_A}\Big|_{y_{0,\lambda}} \sum_{j=1}^{i} \gamma_{ij} k^\lambda_{A,j} + \frac{h}{m}\ \frac{\partial f_A}{\partial y_L}\Big|_{y_{0,\lambda}} \sum_{j=1}^{i} \nu_{ij} k_{L,j}.$$

Still, the coupling terms need to be defined, where we aim at

- active to latent: $\bar{Y}_{A,i} \approx y_A(t_0 + \tilde{\alpha}_i \cdot H)$,
- latent to active: $\bar{Y}^\lambda_{L,i} \approx y_L(t_0 + \lambda \cdot h + \alpha_i \cdot h)$,

with $\alpha_i := \sum_{j=1}^{i-1} \alpha_{ij}$. Depending on the way how coupling terms are computed, we get different types of multirate formulae:

MROW [91]: The coupling terms are defined by the usage of rational extrapolations y_A^{extra} and y_L^{extra}, respectively:

- active to latent: $\boxed{\bar{Y}_{A,i}} = y_A^{\text{extra}}\big(t_0 + \tilde{\alpha}_i\, H\big)$;
- latent to active: $\boxed{\bar{Y}^\lambda_{L,i}} = y_L^{\text{extra}}\big(t_0 + (\lambda + \alpha_i)\, h\big)$.

Furthermore $N := (\nu_{ij})_{i,j=1,\dots,s} = 0$, $\tilde{N} := (\tilde{\nu}_{ij})_{i,j=1,\dots,s} = 0$; the coefficient matrices $A := (\alpha_{ij})_{i,j=1,\dots,s}$, $\tilde{A} := (\alpha_{ij})_{i,j=1,\dots,s}$ are strict lower triangular, and $G := (\gamma_{ij})_{i,j=1,\dots,s}$, $\tilde{G} := (\tilde{\gamma}_{ij})_{i,j=1,\dots,s}$ are lower diagonal with non-vanishing diagonals. Last, the Jacobian is evaluated at: $y_{0,\lambda} = \big(y_L^{\text{extra}}(t_0 + \lambda\, h),\ y_A^{\text{h}}(t_0 + \lambda\, h)\big)$, $\lambda = 0, \dots, m-1$. Thus the computation over each macro step is decoupled, a kind of weakened slowest first strategy [82].

Generalized Multirate [145]: The coupling terms are computed by their 'own' RK-like methods,

$$\boxed{\bar{Y}_{A,i}} = y_{A,0} + m \sum_{j=1}^{i-1} \tilde{\delta}_{ij} k^0_{A,j}, \quad \text{and}$$

$$\boxed{\bar{Y}^\lambda_{L,i}} = y_{L,0} + \frac{1}{m} \sum_{j=1}^{i-1} (\delta_{ij} + F_j(\lambda)) k_{L,j},$$

which gives us a genuine one-step method. Fixing, where to evaluate the Jacobian and some finer structure of the coefficient matrices, yields different kinds of methods:

- *explicit Runge-Kutta [145]:* $G = N = \tilde{G} = \tilde{N} = 0$ - thus no Jacobian is coupled.
- *partitioned Runge-Kutta [103]:* $G = N = \tilde{N} = 0$; $\tilde{G}$ with non vanishing diagonal.
- *W-method [10]:* G, $\tilde{G}$, N and $\tilde{N}$ have constant diagonals, which differ from zero at least for the first two matrices; in addition $y_{0,\lambda} = y_0$, i.e. the Jacobian is lagged over a single macro step in order to compute the micros.
- *ROW-method [10]:* Conditions like W-method, plus evaluation of Jacobian on the fine grid.

Generalized multirate schemes yield a compound step of macro and first micro step, and decouple all later micro steps. By the linear implicitness, we may sequentially compute the increments $k_{L,i}$ and $k^0_{A,i}$. If at least one diagonal element of $N, \tilde{N}$ vanishes, a block triangular form of the system matrix for the increments is obtained, such that the increments may be computed in an interleaved mode: $k_{L,1}$, $k^0_{A,1}$, $k_{L,2}$, $k^0_{A,2}, \dots$. Furthermore, we have ROW-type coefficients, i.e. we need just one decomposition per timestep.

Combining both coupling approaches leads to a hybrid scheme [11]: Whereas the latent and first active step are computed simultaneously in a compound step, the remaining active steps within one macro step can be computed by an arbitrary stiff method, iff dense output formulae of enough accuracy are used for evaluating the latent part. First steps have now been made

to generalize this idea of “mixed multirating” to the charge/flux oriented DAE network equations [228]. Although multirate one-step schemes show promising features to gain speed-up in circuit simulation, the reliablility and robustness of these schemes does not yet allow to use them in standard packages.

Chapter V

Periodic Steady-State Problems

Periodic Steady-State (PSS) Problems have received special attention for simulating analog circuits. The aim was to efficiently study solutions of problems where a highly oscillating signal (carrier) was modulated by another signal. Due to non-linear components the response to a single tone may give rise to higher harmonics, which in general is considered as (harmonic) distortion. When two tones are considered, intermodulation distortion may arise. Then an IC-designer is interested in detecting the (group of) components that contribute most to the distortion. The same analyses also allow study of Electromagnetic Compatibility Immunity.
The above problems were studied by techniques in the time domain, in the frequency domain, or by mixed time-frequency domain methods. In the last years, Radio Frequency simulation initiated renewed focussing on simulating PSS problems, especially in the time-domain [56, 142, 232, 233].
This Section describes several algorithms for simulating PSS problems. This will include forced problems (i.e. periodicity caused by external sources) as well as free oscillator problems. However, we will point out also that the separate algorithms are a step in a larger process: distortion analysis, immunity analysis, noise analysis. A complete algorithm shows a cascading sequence of basic simulation methods (for instance for providing initial approximative solutions). Another feature is that algorithms are favoured that exploit re-use of existing implementations.

17 RF Simulation

In the past decade there has been an exponential growth in the consumer market for wireless products. Products like pagers, cordless and cellular phones are now common products for consumers all over the world. But also computers are no longer connected to other computers and their peripherals by copper wires only: wireless computer networks are used more and more. Not yet very common but growing steadily are the wireless home systems, connecting all kinds of equipment present in peoples homes. Furthermore there are promising markets in the automotive area in vehicular navigation and inter-vehicular communication.

The change from mainly professional wireless applications (military, private mobile radio, etc.) to a consumer market has severe implications for the total design process. Where in the past there was time to build and measure several prototypes, nowadays the demands on time-to-market, time-to-quality, price, production volume, etc. are so severe that designers have to resort to simulation. In a marketing window of only a few months there clearly is no time for several iterations of these systems-on-silicon.

Although the RF part of these systems constitutes only a minor part of the total design area, it presents a major challenge in the total design cycle. This challenge is caused by the analogue/RF nature of the design but also by the lack of appropriate tools, models and design flows. Because the demand for RF simulation tools on this scale is relatively new, the developments of tools (the underlying principles and the commercial implementation there of) are lagging behind the designers needs. It is clear that we are only in the start-up phase of RF tooling and RF design flow development. Nevertheless, recently a lot of progress has been made in the research of

mathematical principles for RF simulation. A number of these new ideas are already available in industrial and commercial software.

RF circuit and signal characteristics

An RF circuit forms the link between some baseband information signal and an antenna. A transmitter modulates the baseband signal on a high frequency carrier (sinusoid) and the task of the receiver is to retrieve the baseband signal from the modulated carrier. Thus, as compared to baseband circuits, RF circuits are special in the sense that they process modulated carriers. In the frequency domain a modulated carrier is a narrow band signal where the absolute bandwidth is related to the frequency of the carrier signal and the relative bandwidth is related to the modulating baseband signal. Practically, the ratio of the two frequencies is in the order of 100 or 1000.

Another major difference is that in RF systems, noise is a major issue. Noise consists of the (usually) small unwanted signals in a system. One can think of several forms of device noise (thermal noise, shot noise, flicker noise) but also of interferers like neighbouring channels, mirror frequencies, etc. All noise sources are of major importance because they directly translate to bit-error-rates of the transmitted data. Therefore it is imperative that RF designers can predict the overall noise quickly and accurately.

When dealing with narrow band signals in a noisy environment two mechanisms are of major importance. Firstly, if a narrow band signal is passed through a non-linearity, the spectrum will be repeated about integer multiples of the carrier frequency resulting in a very wide but sparse spectrum. Secondly, the signal will interact with other signals in the circuit leading to wanted and unwanted frequency shifts. Although both mechanisms are always present and even interact, the first mechanism is less important for small signal levels (e.g. noise).

RF building blocks

RF systems are typically built from a limited number of different building blocks: oscillators, mixers, amplifiers/filters, dividers and power amplifiers. When building or discussing special RF circuit simulation functionality it is important to first determine the characteristics of each building block and the information which should be obtained during simulation:

- Oscillators are autonomous circuits which serve as a frequency reference signal often of very high accuracy. Therefore the frequency itself must be determined accurately but it is also important to be able to determine the frequency behaviour over time, i.e. the phase noise. Physically the phase noise is caused by the device noise of the oscillator's components.
- Mixers perform a frequency shift on the input spectrum. Because of unwanted non-linearities the input signal will not only be shifted but also distorted. Furthermore, the mixer will add noise to the signal, again generated by the devices in the circuit.
- Amplifiers and filters also suffer from unwanted non-linearities and add noise to the signal.
- Dividers are used to modify a frequency reference signal for example coming from an oscillator. They are strongly non-linear and they add phase noise to the signal.
- Power amplifiers are much like small signal amplifiers. However, depending on the modulation type and efficiency requirements they may be strongly non-linear. Assessing the non-linearity, especially in the frequency domain is important.

Requirements for Simulating RF circuits

As mentioned earlier, noise is of major importance in RF circuit design. Depending on the required accuracy and application area the noise can be seen as a small, independent signal in

the circuit. Much more often, however, the small noise signals interact with the large signals in the circuit resulting in frequency shifts of the noise spectra (noise folding). In a few cases the noise can not even be considered as a small signal but interacts with the other (noise) signals in a non-linear manner. The RF designer must be able to simulate all these different views on noise but the second one is considered the most important.

Non-linearity (harmonic distortion and intermodulation distortion) is mainly a measure for the behaviour of a circuit under unwanted strong disturbances which enter the system.

RF designers must be able to extract this information by simulating a design with reasonable turn-around times. This has to do with the actual computing time required for a simulation job but also addresses the robustness of the software. Equally important, however, is that the results are accurate and hence reliable.

18 The basic two-step approach

From the above it is clear that conventional SPICE-like simulators are not sufficient: transient simulation of RF circuits suffers from excessive CPU times because they have to deal with the absolute bandwidth of the signals and will therefore only be used when no alternatives are available (e.g. full non-linear noise simulation including time domain transient noise sources). AC analysis can easily deal with the high bandwidths but does neither take into account non-linearities nor frequency shifts.

The newly developed RF simulation methods all somehow exploit the 'sparsity' of the signal spectra. The basic method is that of determining the periodic steady-state (PSS) solution of a circuit. Conceptually this can be seen as a generalisation of the well-known DC operating point: for baseband circuits the spectral content around 0 Hz (the DC point) is important. For RF circuits the (narrow) spectral content around specific frequencies (of the PSS solution) is of interest. This PSS solution can be obtained in the frequency domain (f.i. by applying the harmonic balance method) or in the time domain (by methods, like shooting, based on transient simulation methods). With baseband simulation, after determining the DC point, additional simulations like AC, noise, etc. can be done to obtain more information about the circuit. Similarly, based on the PSS solution several other simulations can be done like periodic AC, periodic noise, etc. In view of the RF circuit and signal characteristics, the PSS solution determines the non-linear behaviour of the circuit while the periodic AC, etc. deals with the frequency shift.

The main difference between the time domain and frequency domain methods to obtain the PSS solution is that the former can easily deal with strongly non-linear circuits and discontinuities and have good convergence properties while the latter deal naturally with components characterised in the frequency domain. Over the years combinations of both basic methods were developed resulting in mixed time-frequency domain approaches each with their own advantages and drawbacks.

A *two-step approach* appears to be powerful as well as practical for simulating *RF mixing noise*:

- Determine the *noiseless Periodic Steady-State (PSS) solution* as large-signal solution. This can be done in the time-domain, the frequency-domain or by using mixed time-frequency methods. The time-domain representation is a time-varying solution.
  Of course, a noiseless PSS-analysis (with or without determining the oscillation frequency), has value on its own for RF simulations.
- Apply a linearisation around the PSS-solution and study noise as a *small signal perturbation*. The noise sources may have frequencies that are different from the PSS-solution.

For simulating *RF phase noise*, or *timing jitter* (i.e. shifts in zero crossings of the solution) in the case of free oscillators, to apply as second step a linearisation around the PSS-solution and study noise as small signal perturbation is of limited use [49]. In fact, the results are only usefull for small t, because the resulting perturbations may grow large with time. But it allows

that the noise sources may have frequencies that are different from the PSS-solution.
The *non-linear perturbation analysis*, proposed in [49], is an alternative to the second step. Also in this approach, the first step is necessary. The non-linear perturbation analysis results in a correct phase deviation. For the orbital deviation, again a linearisation around the PSS-solution (but including phase deviation) can be used. This implies that periodicity of the coefficients of the linear time varying differential equation can not be assumed. It also implies that, in general, the phase deviation is a time varying function.

After defining the PSS problem mathematically, we describe in the next two sections some methods for these phases in some more detail.

19 The PSS Problem

For the charge/flux oriented network equations (10.1) in compact form, the Periodic Steady-State (PSS) problem for one overall period $T > 0$ is defined as:

$$\frac{\mathrm{d}}{\mathrm{d}t}q(t,x) + j(t,x) = 0 \in \mathbb{R}^N, \tag{19.1}$$

$$x(0) - x(T) = 0 \tag{19.2}$$

with $q(t) := A{\cdot}g(x)$ denoting charges assembled at the respective nodes and fluxes, and $j(t,x) := f(x,t)$ including the static part and sources as well. This implies that for all $t \in \mathbb{R}$, $x(t) = x(t+T)$. A function $x : \mathbb{R} \to \mathbb{R}^n$ is called a *Periodic Steady-State Solution* if there is a $T > 0$ such that x satisfies (19.1)-(19.2). Note that according to this definition, a stationary solution (called the DC, direct current, solution), i.e. a solution of the form $x(t) \equiv x_0$, is also a PSS solution.

To define precisely the PSS problem, we have to introduce the concept of *limit cycles* and to define *stability* for PSS solutions and limit cycles.

The *limit cycle* $\mathcal{C}(x)$ of a PSS solution x is the range of the function $x(t)$, i.e.

$$\mathcal{C}(x) = \{x(t) \,|\, t \in \mathbb{R}\}. \tag{19.3}$$

A set $\mathcal{C}$ is called a limit cycle of (19.1) if there is a PSS solution x of (19.1) so that $\mathcal{C} = \mathcal{C}(x)$.

A PSS solution x is called *stable* (some authors prefer the term *strongly stable*) if there is a $\delta > 0$ such that for every solution x^* to (19.1) which has the property that

$$\exists_{\tau_1 > 0} ||x^*(0) - x(\tau_1)|| < \delta, \tag{19.4}$$

there exists a $\tau_2 > 0$ such that

$$\lim_{t \to \infty} ||x^*(t) - x(t + \tau_2)|| = 0 \tag{19.5}$$

A limit cycle is called stable when all of its periodic steady-states are stable.

Periodic steady-states solutions that are not stable are not interesting for the IC designer, since they do not correspond to any physical behaviour of the modelled circuit. In fact, we want to actively avoid non-stable periodic steady-states solutions for this reason.
An exception to the above might be the DC solution, which is the most well-known unstable solution. Also numerically the DC solution is of interest because it provides a way to find (approximate, initial) solutions for finding stable solutions, by perturbing the DC solution.
For forced, or driven, (i.e. non-autonomous) problems all explicity time-dependent coefficients and sources are periodic with a common (known) period T. When dealing with autonomous

circuits (also called free-running oscillator circuits) the functions q and j do not explicitly depend on time and j does not involve time-dependent external sources.

$$\frac{\mathrm{d}}{\mathrm{d}t}q(x) + j(x) = 0 \in \mathbb{R}^N. \tag{19.6}$$

$$x(0) - x(T) = 0. \tag{19.7}$$

Despite this, a time-varying periodic steady-state solution may exist for some particular value of T. When this solution is non-trivial, i.e. different from the DC-solution, we will call this solution the oscillation solution and ω_{osc} and f_{osc}, given by $\omega_{\mathrm{osc}} = 2\pi f_{\mathrm{osc}} = \frac{2\pi}{T}$, the angular and 'normal' oscillation frequency, respectively. In the autonomous case, solution and oscillation frequency have to be determined both. Mathematically, the problem is a nonlinear eigenproblem.
In the autonomous case, it is clear that when $x(t)$ is a solution of (19.6)-(19.7), another solution can simply be constructed by making a time-shift: $\tilde{x}(t) = x(t-t_0)$. To make the problem unique, in practice one gauges the solution by requiring that

$$e_i^\top x(t_0) = c \tag{19.8}$$

(for some coordinate i and constant c) [clearly c should be determined in the range of x, but not equal to a DC-value], or by imposing a condition on the time-derivative[9]

$$e_i^\top x'(t_0) = c. \tag{19.9}$$

Now the system (19.6), (19.7) and (19.8), resp., (19.9) defines a nonlinear problem with a "unique" solution for the unknowns x, T: i.e. small time shifts are excluded.

Rescaling the time by writing $t = sT$, with $s \in [0, 1]$, we have

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}q(x(t)) + j(x(t)) &= \frac{1}{T}\frac{\mathrm{d}}{\mathrm{d}s}q(x(sT)) + j(x(sT) \\ &= \frac{1}{T}\frac{\mathrm{d}}{\mathrm{d}s}q(\hat{x}(s)) + j(\hat{x}(s)) \end{aligned} \tag{19.10}$$

where $\hat{x}(s) = x(sT)$. Note that $\hat{x}(1) = x(T)$. Hence, the problem (19.6)-(19.7) can also be studied on the unit interval for the function $\hat{x}(s)$ after scaling the s-derivative by a factor $1/T$. In fact, T can be nicely added to the system as well

$$\frac{1}{T}\frac{\mathrm{d}}{\mathrm{d}s}q(\hat{x}(s)) + j(\hat{x}(s)) = 0 \tag{19.11}$$

$$\frac{\mathrm{d}}{\mathrm{d}s}T = 0 \tag{19.12}$$

$$\hat{x}(0) = \hat{x}(1) \tag{19.13}$$

$$e_i^\top \hat{x}(0) = c \tag{19.14}$$

(Clearly, T automatically fulfills the periodicity condition).

20 Perturbation analysis

Before describing algorithms for solving a PSS-problem, in this section we will consider the problem for a subsequent perturbation analysis. The PSS-solution of (19.1) will be denoted by x_{PSS}. It will also be called the noiseless time-varying large signal solution. Now we perturb the left-hand side of (19.1) by adding some small (noise) function n

$$\frac{\mathrm{d}}{\mathrm{d}t}q(x) + j(t, x) + n(t) = 0 \in \mathbb{R}^N, \tag{20.1}$$

which results in a solution

$$x(t) = x_{PSS}(t + \alpha(t)) + x_n(t), \tag{20.2}$$

in which the phase shift function $\alpha(t)$ still has to be prescribed and $x_n(t)$ is small.

[9] In the following, the prime ' will denote differentiation w.r.t. time.

Linear perturbation analysis for forced systems

Linearising (20.1) around x_{PSS} (i.e. considering the case $\alpha(t) = 0$), results in a Linear Time Varying (LTV) differential equation for x_n

$$\frac{\mathrm{d}}{\mathrm{d}t}(C(t)x_n) + G(t)x_n + n(t) = 0 \in \mathbb{R}^N, \tag{20.3}$$

$$C(t) = \frac{\partial q(x)}{\partial x}\Big|_{x_{\mathrm{PSS}}}, \quad G(t) = \frac{\partial j(t,x)}{\partial x}\Big|_{x_{\mathrm{PSS}}} \tag{20.4}$$

In practical applications, a basic noise term has the form

$$n(t) = B(t)b(t), \tag{20.5}$$

$$B(t) = B(x_{\mathrm{PSS}}(t)) \tag{20.6}$$

that consists of a normalized perturbation function $b(t)$, which is modulated by the periodical function $B(t) = B(x_{PSS}(t))$. Here $b(t)$ may be defined most conveniently in the frequency domain, while the $B(x_{PSS}(t))$ is defined by expressions in the time domain.
The validity of this approach has been discussed by [49]. For forced systems the perturbed solution $x(t)$ can be approximated by (20.2) with α being identically zero and x_n the solution of (20.3). However, when dealing with free oscillators a non-trivial choice for the phase-shift function $\alpha(t)$ has to be made too.

We note that the coefficients in (20.3) are periodic in t with period T. Thus, they can be expanded in exponentials $e^{i\omega_k t}$, in which $\omega_k = 2\pi k/T$. It is instructive to consider the case for a simple sine-wave source, i.e. when

$$n(t) = Ue^{i\nu t}, \tag{20.7}$$

in which U does not depend on time, and $\nu = 2\pi f_n$, where f_n may be interpreted as a noise frequency, that may be different from the ω_k. Introducing $y_n(t) \equiv e^{-i\nu t}x_n(t)$ results in a linear DAE of which source term and (complex) coefficients (that depend on the parameter ν) are periodical with period T

$$\frac{\mathrm{d}}{\mathrm{d}t}(C(t)y_n) + [G(t) + i\nu C(t)]y_n + U = 0 \in \mathbb{R}^N. \tag{20.8}$$

When $x_{PSS}(t) \equiv x_{DC}$, and $[G(t) + i\nu C(t)]$ is regular (and time-independent), the solution y_n is time-independent and simply equals the well-known AC-solution. For the general case, we find that y_n and x_n have expansions of the form (see also [179, 231])

$$y_n(t) = \sum_{k=-\infty}^{\infty} y_{n,k}^{(\nu)} e^{i\omega_k t}, \tag{20.9}$$

$$x_n(t) = \sum_{k=-\infty}^{\infty} y_{n,k}^{(\nu)} e^{i(\nu+\omega_k)t}. \tag{20.10}$$

Because of the periodic coefficients in (20.3) and (20.8), the determination of the $y_{n,k}^{(\nu)}$ is called Periodic AC (PAC) analysis. The expansion of $x_n(t)$ implies that

$$x_n(t+T) = \beta(\nu)x_n(t), \text{ or} \tag{20.11}$$

$$x_n(0) = \beta(-\nu)x_n(T), \text{ where} \tag{20.12}$$

$$\beta(\nu) = e^{i\nu T} \tag{20.13}$$

It is clear that, for a single input frequency ν, the solution $x_n(t)$ contains frequencies of the form $(\nu + \omega_k)$, i.e. frequency folding occurs. If we allow for several input frequencies ν_i, we can also say that a certain output frequency might originate from a large number of possible input frequencies. Hence, noise components at a certain frequency might end up in a different frequency band. This is why, for example, $1/f$ noise which has its main energy at low frequencies, still plays an important role in RF circuits.

It is important to note that we described a *linear* perturbation analysis and we will not find contributions containing for example $(\nu_1+\nu_2+\omega_k)$, $(\nu_1+2\nu_2+\omega_k)$ etc. This assumption is in general not a severe limitation when simulating noise in RF circuits.
In [179, 231] one considers the integration of (20.3) in which case the factor β easily allows adaptive re-usage of linear algebra used for solving the PSS-problem (see also [14, 15]). However, the integration of (20.8) also gives rise to elegant algorithms.

Floquet theory

When dealing with perturbed *oscillatory* systems

$$\frac{\mathrm{d}}{\mathrm{d}t}q(x)+j(x)+n(t) \quad = \quad 0 \in \mathbb{R}^N \tag{20.14}$$

it is no longer possible to assume that small perturbations $\mathbf{n}(t)$ lead to small deviations in $x_{PSS}(t)$ [An instructive example is provided by considering $y'(t)+\cos(t)y(t)-1=0$, of which the inhomogeneous solution is not periodic at all; however, note that $y(t+2\pi)$ still satisfies the differential equation]. The main reason is that the *period* of the large signal solution is influenced by $n(t)$. This can lead to large (momentary) frequency deviations such that the difference between the noiseless and noisy solution can no longer be considered to be small.
This section gives the necessary background of Floquet Theory when applied to oscillatory problems and which provides a way to a proper perturbation approach [46, 49, 147, 148]. We start by noting that $x'_{\mathrm{PSS}}(t)$ satisfies the homogeneous part of (20.3)

$$\frac{\mathrm{d}}{\mathrm{d}t}(C(t)x)+G(t)x \quad = \quad 0 \tag{20.15}$$

We assume the case of index 1 DAEs. At the end of the section the higher index case is considered.

Independent solutions. Let,

$$\mathbf{S}(t) \quad = \quad \{z \in \mathbb{R}^N \mid (G(t)+\frac{\mathrm{d}}{\mathrm{d}t}C(t))z \in \mathrm{Im}(C(t))\}, \tag{20.16}$$

$$\mathbf{N}(t) \quad = \quad \mathrm{Ker}(C(t)). \tag{20.17}$$

Then one has

$$\mathbf{S}(t) \cap \mathbf{N}(t) \quad = \quad 0, \tag{20.18}$$

$$\mathbf{S}(t) \oplus \mathbf{N}(t) \quad = \quad \mathbb{R}^N, \tag{20.19}$$

in the index-1 case. We assume that $\mathbf{S}(t)$ is m-dimensional. There are N independent solutions of the homogeneous problem: $u_1(t)e^{\mu_1 t},\ldots,u_m(t)e^{\mu_m t},u_{m+1}(t),\ldots,u_N(t)$. The first $u_1(t),\ldots,u_m(t)$ are a basis of $\mathbf{S}(t)$; the last, $u_{m+1}(t),\ldots,u_N(t)$, are a basis of $\mathbf{N}(t)$. The $\mu_1,\ldots,\mu_m$ are so-called Floquet exponents; the $e^{\mu_1 T},\ldots,e^{\mu_m T}$ are Floquet multipliers. For a stable autonomous index 1 problem we can assume that $\mu_1=0$ and that $\mathrm{Re}(\mu_i)<0$ for $i=2,\ldots,m$. In this case we can choose $u_1(t)=x'_{\mathrm{PSS}}(t)$.

Adjoint problem. The homogeneous adjoint (or dual) system corresponding to (20.3) is

$$C^\top(t)\frac{\mathrm{d}}{\mathrm{d}t}y-G^\top(t)y \quad = \quad 0 \tag{20.20}$$

Similar to the not-adjoint case we introduce

$$\mathbf{S}^\top(t) \quad = \quad \{z \in \mathbb{R}^n \mid G^\top(t)z \in \mathrm{Im}(C^\top(t))\}, \tag{20.21}$$

$$\mathbf{N}^\top(t) \quad = \quad \mathrm{Ker}(C^\top(t)), \tag{20.22}$$

which have the properties

$$\mathbf{S}^\top(t) \cap \mathbf{N}^\top(t) = 0, \tag{20.23}$$
$$\mathbf{S}^\top(t) \oplus \mathbf{N}^\top(t) = \mathbb{R}^N. \tag{20.24}$$

Also $\mathbf{S}^\top$ is m-dimensional. The adjoint problem has N independent solutions: $v_1(t)e^{-\mu_1 t}, \ldots, v_m(t)e^{-\mu_m t}, v_{m+1}(t), \ldots, v_N(t)$, where $v_1(t), \ldots, v_m(t)$ are a basis of $\mathbf{S}^\top(t)$ and the last, $v_{m+1}(t), \ldots, v_N(t)$, are a basis of $\mathbf{N}^\top(t)$.

Bi-orthogonality. It is easy to verify that if x and y are solutions of (20.15) and (20.20), respectively, the inner-products $y^\top(t)C(t)x(t)$ are constant, thus $y^\top(t)C(t)x(t) = y^\top(0)C(0)x(0)$, for all $t \geq 0$. More specifically, the bases $u_1(t), \ldots, u_N(t)$ and $v_1(t), \ldots, v_N(t)$ can be chosen such that, the $N \times N$ matrix $U(t)$ with as columns the $u_i(t)$ and the $N \times N$ matrix $v(t)$ with as rows the $v_i(t)$ satisfy a bi-orthogonality relation w.r.t. $C(t)$ and a nearly one w.r.t. $G(t)$

$$v(t)C(t)U(t) = \begin{pmatrix} I_m & 0 \\ 0 & 0 \end{pmatrix}, \tag{20.25}$$

$$v(t)G(t)U(t) = \begin{pmatrix} J_m^1 & 0 \\ J_m^2 & J_m^3 \end{pmatrix}. \tag{20.26}$$

Here I_m is a $m \times m$ identity matrix. J_m^1 is a $m \times m$ block matrix. J_m^2 and J_m^3 are just suitable block matrices.

State-transition matrix, monodromy matrix. Assuming a consistent initial condition $x(0) = x_0 \in \mathbf{S}(0)$, the solution $x_H(t)$ of (20.15) can be written as

$$x_H(t) = \sum_{i=1}^{m} u_i(t) \exp(\mu_i t) v_i^\top C(0) x_0, \tag{20.27}$$
$$= \Phi(t,0)x_0, \tag{20.28}$$
$$\Phi(t,s) = \Theta(t,s)C(s), \tag{20.29}$$
$$\Theta(t,s) = U(t)D(t-s)v(s), \tag{20.30}$$
$$D(t-s) = \mathrm{diag}(\exp(\mu_1(t-s)), \ldots, \exp(\mu_m(t-s)), 0, \ldots, 0) \tag{20.31}$$

If x_0 is not a consistent initial value, one can write $x_0 = x_0^{(S)} + x_0^{(N)}$, where $x_0^{(S)} \in \mathbf{S}(0)$ and $x_0^{(N)} \in \mathbf{N}(0)$. Clearly $C(0)x_0 = C(0)x_0^{(S)}$, and $x_H(t)$ depends on $x_0^{(S)}$, rather then on x_0. For calculating consistent initial values we refer to [19].
An inhomogeneous solution of (20.3) can be written as

$$x_P(t) = x_H(t) + \sum_{i=1}^{m} u_i(t) \int_0^t \exp(\mu_i(t-s)) v_i^\top(s) B(s) b(s) \mathrm{d}s + \Gamma(t)B(t)b(t), \tag{20.32}$$
$$= x_H(t) + \int_0^t \Theta(t,s)B(s)b(s)\mathrm{d}s + \Gamma(t)B(t)b(t) \tag{20.33}$$

Here $\Gamma(t)$ is a matrix with $\mathrm{Ker}(\Gamma(t)) = \mathrm{Span}(C(t)u_1(t), \ldots, C(t)u_m(t))$.

The monodromy matrix is the matrix $\Phi(t,0)$ after one period, i.e. $\Phi(T,0)$ (this matrix one naturally studies when one considers shooting methods or applies Floquet theory for analyzing stability of a limit cycle). Because of the periodicity of the u_i, we see that the $u_i(0)$, for $i = 1, \ldots, m$ are eigenvectors of the monodromy matrix with corresponding eigenvalues $\exp(\mu_i T)$, and that the remaining $u_i(0)$, for $i = m+1, \ldots, N$, are eigenvectors for the $(N-(m-1))$-fold eigenvalue 0.

The adjoint problem (20.20) has the state-transition matrix

$$\Psi(t,s) = v^\top(t)D(s-t)U^\top(s)C^\top(s), \tag{20.34}$$
$$= \sum_{i=1}^{m} \exp(-\mu_i(t-s)) v_i(t) u_i^\top(s) C^\top(s) \tag{20.35}$$

Similar to the not-adjoint case, the $v_i(0)$ are eigenvectors of the associated monodromy matrix $\Psi(T,0)$.

The higher index case. The index-2 case is discussed in [149] for quasilinear problems, which is sufficient here. It turns out that not only the algebraic but also the hidden constraints (see the discussion in Section 10) have to be observed when setting up the state transition and monodromy matrix. Especially they have to start from consistent initial values. The latter can either be computed with the methods sketched in Section 10, or columnwise as eigenvectors of a generalized eigenvalue problem. Of course the Floquet multipliers are only those of the independent part of the monodromy matrix. The stability criterion is again that all Floquet multipliers have magnitude < 1, except that one which has magnitude 1 in case of autonomous oscillation.

Remark. Sometimes the monodromy matrix in the higher index case is defined to comprise only the linear independent parts of $\Phi(T,0)$ i. e. the basis vectors of $\mathbf{S}(t)$ [220]. This delivers the same information as before, but may save some memory space and computational effort for its calculation.

Phase noise by non-linear perturbation analysis

Phase shift function $\alpha(t)$. We will take $u_1(t) = x'_{\rm PSS}(t)$. Let $\alpha(t)$ be a (sufficiently smooth) phase- or time-shift function and let $s = t + \alpha(t)$ be the shifted time. If $x_{\rm PSS}(t)$ is the PSS-solution of (19.6) then the phase-shifted function $y(t) \equiv x_{\rm PSS}(s) = x_{\rm PSS}(t + \alpha(t))$ satisfies

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}q(y) + j(y) &= \frac{\mathrm{d}}{\mathrm{d}s}q(x_{\rm PSS}(s)).\frac{\mathrm{d}s}{\mathrm{d}t} + j(x_{\rm PSS}(s)) \\ &= \frac{\mathrm{d}}{\mathrm{d}x_{\rm PSS}}q(x_{\rm PSS}(s))\frac{\mathrm{d}x_{\rm PSS}}{\mathrm{d}s}\alpha'(t) \\ &= C(t+\alpha(t))u_1(t+\alpha(t))\alpha'(t). \end{aligned} \tag{20.36}$$

Hence, the phase shifted function y satisfies a perturbed DAE in which the right-hand side has a particular form. Here u_1 is the tangent to the orbit.
We now consider perturbations of the form $B(x(t))b(t)$ (cf. also (20.3)) to the original DAE (19.6)

$$\frac{\mathrm{d}}{\mathrm{d}t}q(x) + j(x) + B(x(t))b(t) = 0 \tag{20.37}$$

and express $B(x(t+\alpha(t)))b(t)$ into its components using the basis $\{C(t+\alpha(t))u_1(t+\alpha(t)), \ldots, C(t+\alpha(t))u_m(t+\alpha(t)), G(t+\alpha(t))u_{m+1}(t+\alpha(t)), \ldots, G(t+\alpha(t))u_N(t+\alpha(t))\}$

$$\begin{aligned} B(x(t+\alpha(t)))b(t) &= \sum_{i=1}^{m} c_i(x,\alpha(t),t)C(t+\alpha(t))u_i(t+\alpha(t)) \\ &\quad + \sum_{i=m+1}^{N} c_i(x,\alpha(t),t)G(t+\alpha(t))u_i(t+\alpha(t)), \end{aligned} \tag{20.38}$$

$$c_i(x,\alpha(t),t) = \tilde{v}_i(t+\alpha(t))b(t) \tag{20.39}$$

$$\tilde{v}_i(t) = v_i^\top(t)B(x(t)) \tag{20.40}$$

Here the scalar functions $\tilde{v}_i(t)$ are periodical in t with period T.
The first component of $B(x(t+\alpha(t)))b(t)$ will be used to determine $\alpha(t)$. We define $\alpha(t)$ to satisfy the non-linear, scalar, differential equation

$$\alpha'(t) = -v_1^\top(t+\alpha(t))B(x_{\rm PSS}(t+\alpha(t)))b(t), \quad \alpha(0) = 0 \tag{20.41}$$

$$= -\tilde{v}_1(t+\alpha(t))b(t), \quad \alpha(0) = 0 \tag{20.42}$$

(See also already [130, 131] where a first start was made to treat the phase noise problem in the time-domain.) In [46, 49] it is argued that, in first order, (20.37) has a solution of the

form $y(t) + z(t)$, with α determined by (20.42), and where the orbital deviation $z(t)$ satisfies $||z||_\infty < \text{Const.}||b||_\infty$ (and even $z(t) \to 0$ $(t \to \infty)$). However, the phase shift function $\alpha(t)$ may increase with time (Clearly, if $N = 1, B \equiv 1, b(t) \equiv \varepsilon$, and $v_1^\top(t) \equiv \kappa$, then $\alpha(t) = \kappa\varepsilon t$).

Determination of v_1. Note that for finding α, we clearly have to know v_1. In [48] this crucial vector is called *Perturbation Projection Vector*, or *PPV*. It represents a transfer between the perturbation of the DAE and the resulting phase shift.
In [49] v_1 is determined by performing first an eigenvalue/eigenvector analysis of the monodromy matrix of the adjoint problem to obtain $v_1(0)$, and followed by time integration (backward in time). For distincting the proper initial value $v_1(0)$ from other eigenvectors that have eigenvalues close to 0, one can exploit the bi-orthogonality relation (20.25), because $v_1(0)$ must have a non-trivial C-inner-product with $u_1(0)$.
Another, direct, approach is found in [48]. It nicely fits a Finite Difference Method approach and again exploits the bi-orthogonality relation (20.25) in an elegant way.

Phase noise analysis. For deterministic perturbations one has to integrate (20.42). Because of this action in the time domain, all Fourier components of $n(t)$ are treated in a combined way. However, for stochastic noise, such a detail is not necessary. In [46, 49] expressions for the power due to the noise are derived that depend on the asymptotic behaviour (i.e. for large t) of the variance $\text{var}[\alpha(t)]$. The authors derive power spectrum expressions that depend on the Fourier components of the PSS-solution x_{PSS}, on the DC-component of $v_1(t)$, and on the power spectrum of b. The power of the j-th harmonic of x_{PSS} is preserved in the power of the 'asymptotic' j-th harmonic of y (i.e. the shifted x_{PSS}). Consequently, by summing over j, we see that also the total power is preserved.

Orbital deviation. In fact, the orbital deviation function z can be analysed by a proper linear perturbation analysis (but with linearised equations which now have non periodic coefficients!). Because n also affects the phase shifted function, around which one linearises for studying the orbital devations, there is no simple summation formula known for cumulative noise contributions.

Other approaches. Finally, we briefly mention some alternative approaches for determining phase noise. In [53] a technique based on careful sampling is described to find phase noise effects due to specific noise sources. In [**?**] phase noise is considered from a parameter dependency point of view and an averaging technique is described that works well on (but is also restricted to) finite time intervals and is of interest in behavioural modeling. In [112] a less accurate, but faster phase noise model is described that neglects the occurrence of α at the right-hand side in (20.42).

21 Algorithms for the PSS problem

In this section we describe some algorithms for solving PSS problems (i.e. for solving the noiseless, time varying, large signal). A general overview of numerical methods for highly oscillating problems can be found in [185].
As time integrator we restrict ourselves to a θ-method ($0 \le \theta \le 1$): for the explicit ODE system $\dot{y}(t) = f(x(t), t)$, one step to compute the approximate y_{n+1} at time point $t_n + h$ from the previous approximate y_n at t_n reads

$$\frac{y_{n+1} - y_n}{h} = \theta f(y_{n+1}, t_n + h) + (1 - \theta) f(y_n, t_n)\,.$$

This class of methods includes the explicit Euler-forward method ($\theta = 0$), the Trapezoidal Rule ($\theta = 0.5$) and the implicit Euler-backward scheme ($\theta = 1$). For other methods, f.i. BDF-like ones, see [256, 257].

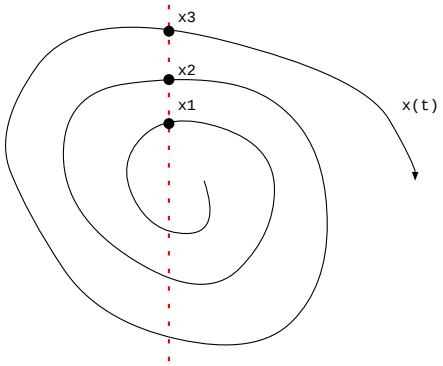


Figure 32: The trajectory of a solution, cut with a hyperplane.

Direct time integration methods

Ordinary time integration usually starts from the DC solution. For forced (non-autonomous) problems the time integration usually is very slow, because the step-size will be determined by the highest oscillating component of the solution. For these problems, the Finite Difference Method (FDM), the Shooting Method (SM), the Harmonic Balance (HB), or the Envelope approach provide much more efficient alternatives. However, in analysing autonomous, free oscillating, problems, time integration also shows a nice property in securing to find stable limit cycles. For this reason, in this subsection, we will concentrate on finding a free oscillating solution, by exploiting time integration. With extrapolation techniques one can speed up convergence. In the past this approach has been applied by [224] (even already for circuit problems) and [184]. In [226] extrapolation techniques were generalized to sequences of vectors and has resulted in methods like Minimal Polynomial Extrapolation (MPE) and Reduced Rank Extrapolation (RRE). It is worth noting that all these methods can be implemented very elegantly within existing circuit simulators.

The basic Poincaré Method. The basic method for solving (19.6)-(19.7) is called the Poincaré-map method. First we note that the length of the period can be estimated by looking for periodic recurring features in the computed circuit behaviour. A possible recurring feature is the point at which a specific condition is satisfied. This is equivalent to carrying out a Poincaré-map iteration, see [109], section I.16. The idea is to cut the transient solution $x(t)$ by a hyperplane. The hyperplane is defined by an affine equation of the form $x^\top(t)n = \alpha$, for some vector n and scalar α. This equation is called the *switch equation*. The situation is visualised in Figure 32. The basic Poincaré-map method can now be described as follows. Let an approximate solution x_0 and a required accuracy tolerance $\varepsilon > 0$ be given. The approximated solution $\tilde{x}$ and period $\tilde{T}$ is computed by:

$i := 0$, $t_0 := 0$, $x_0 :=$ some initial guess for x
repeat
  Starting with $t = t_i$, $x(t_i) = x_i$, integrate (19.6) until $(x(t), n) = \alpha$ and $\mathrm{d}(x(t), n)/\mathrm{d}t > 0$.
  $x_{i+1} := x(t)$, $t_{i+1} := t$
  $\delta := ||x_{i+1} - x_i||$
  $i := i + 1$
until $\delta \leq \varepsilon$
$\tilde{T} := t_i - t_{i-1}$, $\tilde{x} := x_i$

The MPE accelerated Poincaré-map method. Let $x(t)$ be the solution of (19.6) with $x(0) = x_0$, and T_0 is the smallest $t > 0$ such that $(x(t), n) = \alpha$ and $\mathrm{d}(x(t), n)/\mathrm{d}t > 0$. Thus T_0 depends on x_0 as well.

Now we can define a function $F : \mathbb{R}^n \to \mathbb{R}^n$ by

$$F(x_0) := x(T_0). \tag{21.1}$$

The successive approximations of the Poincaré-map method satisfy the recursion relation

$$x_{n+1} = F(x_n). \tag{21.2}$$

This recursion is only in terms of the circuit state x; the period T does not appear explicitly in this iteration. Suppose that the sequence (21.2) converges linearly to some fixed point $\tilde{x}$ of F. A vector-extrapolation method to accelerate the basic method operates on the first k vectors of a sequence $\{x_n\}$, and produces an approximation y to the limit of $\{x_n\}$. This approximation is then used to restart (21.2) with $y_0 = y$ and the basic method generates a new sequence $y_0, y_1, y_2, \ldots$. Again, the acceleration method can be applied to this new sequence, resulting in a new approximation z of the limit. The sequence $x_0, y, z, \ldots$ converges much faster to the limit of $\{x_n\}$ than the sequence $\{x_n\}$ itself. Typically, if $\{x_n\}$ converges linearly, then $\{x_0, y, z, \ldots\}$ converges super-linearly.
A well-known acceleration method is minimal polynomial extrapolation (MPE). Rather than describing MPE here in detail, the reader is referred to [226]. For results with this approach we refer to [122–124].

Finite Difference Method

The Finite Difference Method (FDM) solves the problem on a fixed time grid. Given is a number M, a series of M stepsizes $\{\Delta t_i\}$, implying a set of intermediate time-levels $\{t_i\}$ $(0 \le i \le M-1)$, where each t_i is the end point of the interval with length $\{\Delta t_i\}$. We assume that all t_i are contained in an interval of length T, that starts at $A = kT$ (for some $k \ge 0$). Thus

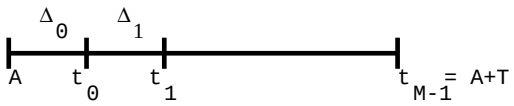


Figure 33: Discretization of interval of length T, starting at A.

$$\begin{aligned}
t_0 &= A + \Delta t_0, \quad A = kT, && (21.3)\\
t_{M-1} &= A + T, && (21.4)\\
t_i &= A + \sum_{k=0}^{i} \Delta t_k, \quad i = 1, \ldots, M-1 && (21.5)\\
\Delta_k = \Delta t_k &= t_k - t_{k-1}, \quad k = 1, \ldots, M-1 && (21.6)\\
\Delta_0 = \Delta t_0 &= t_0 - (t_{M-1} - T) && (21.7)
\end{aligned}$$

Note that in general M will be available just at the start of the PSS-analysis. The periodicity is reflected in the definition of Δ_0.
We will write $\Delta_i = \Delta_i^s T$, for $\Delta_i^s \in [0, 1]$. Then $\sum_{k=0}^{M-1} \Delta_k^s = 1$. Clearly, with $t_i = s_i T$, solutions $\hat{x}(s)$ of the rescaled problem satisfy $\hat{x}(s_i) = x(t_i)$. Thus we will drop the $\hat{}$ and simply include the factor $1/T$ in the expressions when needed.
In the Finite Difference Method, M and the $\{\Delta_i^s\}$ will remain fixed during a complete (PSS-)Newton iteration.
For the next subsections we define

$$C(x) := \partial q(x)/\partial x, \qquad G(t,x) := \partial j(t,x)/\partial x \tag{21.8}$$

and we will write

$$\begin{aligned}
C_i &= C(x(t_i)), \qquad C_i^{(m)} = C(x^{(m)}(t_i)), && (21.9)\\
G_i &= G(t_i, x(t_i)), \qquad G_i^{(m)} = G(t_i, x^{(m)}(t_i)). && (21.10)
\end{aligned}$$

The Basic FD-Method

The resulting discrete system of equations can be written as

$$\begin{aligned} F^D(x,T) &= 0, && (21.11)\\ p^\top x - c &= 0, && (21.12)\end{aligned}$$

where $F^D : \mathbb{R}^{MN} \times R \to \mathbb{R}^{MN}$ is given by

$$\begin{aligned} {F^D}_0(x,T) &= \frac{q(x(t_0)) - q(x(t_{M-1}))}{\Delta_0} + [\theta j(x(t_0)) + (1-\theta) j(x(t_{M-1}))],\\ &= \frac{1}{T}\frac{q(x(t_0)) - q(x(t_{M-1}))}{\Delta_0^s} + [\theta j(x(t_0)) + (1-\theta) j(x(t_{M-1}))], && (21.13)\\ {F^D}_i(x,T) &= \frac{q(x(t_i)) - q(x(t_{i-1}))}{\Delta_i} + [\theta j(x(t_i)) + (1-\theta) j(x(t_{i-1}))],\\ &= \frac{1}{T}\frac{q(x(t_i)) - q(x(t_{i-1}))}{\Delta_i^s} + [\theta j(x(t_i)) + (1-\theta) j(x(t_{i-1}))],\\ & 1 \le i \le M-1. && (21.14)\end{aligned}$$

In (21.12), $p \in \mathbb{R}^{MN}$ is some given vector, usually a unit vector, in which case (21.12) only affects one time level. This last equation will only be imposed when dealing with free oscillators. The above also offers the option to consider the frequency, $f = 1/T$, as natural unknown, rather than T. In that case

$$\begin{aligned} F^D(x,f) &= 0, && (21.15)\\ p^\top x - c &= 0, && (21.16)\end{aligned}$$

where $F^D : \mathbb{R}^{MN} \to \mathbb{R}^{MN}$ is given by

$$\begin{aligned} {F^D}_0(x,f) &= f\ \frac{q(x(t_0)) - q(x(t_{M-1}))}{\Delta_0^s} + [\theta j(x(t_0)) + (1-\theta) j(x(t_{M-1}))], && (21.17)\\ {F^D}_i(x,f) &= f\ \frac{q(x(t_i)) - q(x(t_{i-1}))}{\Delta_i^s} + [\theta j(x(t_i)) + (1-\theta) j(x(t_{i-1}))],\\ & 1 \le i \le M-1. && (21.18)\end{aligned}$$

In the RF-case, one has $f \gg 1$ and thus $\frac{1}{T^2} \gg 1$. Hence, the f-variant behaves better scaled and we will restrict ourselve to that formulation.
Applying Newton-Raphson yields

$$\begin{pmatrix} Y^{(k)} & F^{(k)} \\ p^\top & 0 \end{pmatrix} \begin{pmatrix} x^{k+1} - x^k \\ f^{k+1} - f^k \end{pmatrix} = -\begin{pmatrix} F^D(x^k, f^k) \\ p^\top x^k - c \end{pmatrix} \qquad (21.19)$$

in which

$$\begin{aligned} F^{(k)} &= \frac{\partial}{\partial f} F^D(x^k, f^k), && (21.20)\\ Y^{(k)} &= \frac{\partial}{\partial x} F^D(x^k, f^k) = L + B. && (21.21)\end{aligned}$$

Here L and B are given by

$$\begin{aligned} L &= \begin{bmatrix} \frac{C_0^{(k)}}{\Delta t_0} + \theta {G^{(k)}}_0 & & & \\ -\frac{C_0^{(k)}}{\Delta t_1} + (1-\theta) G_0^{(k)} & \frac{C_1^{(k)}}{\Delta t_1} + \theta G_1^{(k)} & & \\ & \ddots & \ddots & \\ & & -\frac{C_{M-2}^{(k)}}{\Delta t_{M-1}} + (1-\theta) G_{M-2}^{(k)} & \frac{C_{M-1}^{(k)}}{\Delta t_{M-1}} + \theta G_{M-1}^{(k)}, \end{bmatrix},\\ B &= \begin{bmatrix} 0 & \dots & 0 & -\frac{C_{M-1}^{(k)}}{\Delta t_0} + (1-\theta) G_{M-1}^{(k)} \\ 0 & 0 & & \\ & \ddots & \ddots & \\ & & 0 & 0 \end{bmatrix}. && (21.22)\end{aligned}$$

For the fixed period problem, i.e. the non-autonomous case, we just drop the row for $p^\top$ and the column for F and come to

$$Y^{(k)}(x^{k+1} - x^k) \quad = \quad -F^D(x^k). \tag{21.23}$$

Some simple remarks apply:

- $C \equiv 0$ and $\theta = 1$: Then $B = 0$ and L is a block-diagonal matrix. The subsystems for each time level are decoupled and the solutions are the solutions obtained at an ordinary time integration, where the time dependent sources are evaluated at the proper time level. It is clear that when $C \equiv 0$, no oscillation can occur.

- In ordinary transient analysis the DAE character implies the necessary requirement $\theta \neq 0$. However, for the PSS-problem with the Finite Difference Method, $\theta = 0$ is a valid choice (because it is quite similar to $\theta = 1$, but viewed from the opposite time direction). For example: choose $M = 2, \Delta_i = T/2, C_i = \Delta_i C, G_i = G$, then the matrix $L + B$ looks like

$$L + B \quad = \quad \begin{bmatrix} C & -C + G \\ -C + G & C \end{bmatrix} \tag{21.24}$$

  For commuting C, G (for instance $C = \mathrm{Diag}(1, 0), G = \mathrm{Diag}(1, 1)$), the matrix $L + B$ has (non-zero) eigenvalues $\lambda_C + i\lambda_{-C+G}$.

- However, the DAE nature forbids to choose $\theta = 0.5$, because it makes the linear system singular! For seeing this, assume $C_i \equiv C$, $G_i \equiv G$ (constant), and an equidistant discretization with stepsize Δ (and with M odd). Define $C' = \frac{C}{\Delta}$, $G' = 0.5G$. Then

$$L \quad = \quad \begin{bmatrix} C' + G' & & & \\ -C' + G' & C' + G' & & \\ & \ddots & \ddots & \\ & & -C' + G' & C' + G', \end{bmatrix}, \tag{21.25}$$

$$B \quad = \quad \begin{bmatrix} 0 & \dots & 0 & -C' + G' \\ 0 & 0 & & \\ & \ddots & \ddots & \\ & & 0 & 0 \end{bmatrix}, \tag{21.26}$$

  When C' is singular there is a non-trivial vector v such that $C'v = 0$. Then also the large system is singular because $(L + B)w = 0$ for

$$w \quad = \quad (v,\, -v,\, v - v,\, \dots,\, v,\, -v)^\top \tag{21.27}$$

  The trapezoidal rule looks to the mean of two subsequent function values and for this reason one can always add a zig-zag solution to such a "mean" value.
  Hence in practice one will have to take $\theta > 0.5$ and the choice is a trade-off between a better time-integration, but a nearly singular matrix, and more damping (and less order of time-integration), but with a better conditioned matrix.

We can rewrite the system (21.23) as

$$(L + \beta B)x \quad = \quad y, \tag{21.28}$$

in which $\beta = 1$. When studying linearizations around a PSS-solution responses to Fourier source terms give rise to linear systems in which β is complex, but satisfies $|\beta| = 1$ (see Section 20).
Block-Gaussian elimination allows to re-use direct solver modules from a circuit simulator. This way of decomposing the matrix meets a requirement that only memory for a limited number

of full block matrices is used. In this way it is a sparse method. However, it may not be the most optimal LU-decomposition from the point of view of numerical stability, because not the most optimal pivots may be used. For several ideas we refer to [6] (Chapter 7), and [14, 15] (for parallelizable algorithms).
The (block) lower-triangular matrix L is non-singular and can be used as preconditioner for the matrix $(L + B)$ when using a Krylov space method [206]. For this case one needs to be able to determine $L^{-1}Bp$ for some vector p. For an iterative Krylov space method, the Krylov space can be extended by re-using the LU-decompositions of $\frac{C_i}{\Delta t} + G_i$ at each time-level.
For flat matrix circuit simulators, an efficient parallelizable GMRES-algorithm is described in [15].

FD for oscillator problem

For the oscillator problem, the sub-matrix Y in (21.21) is the same that one also encounters when applying the Finite-Difference Method to a forced Periodic Steady-State problem (with a fixed period T). From a software design point of view one would like to re-use software as much as possible. Indeed, when solving (21.19) a Block-Gaussian elimination procedure that uses Y^{-1} is attractive. Note that the complete Newton-matrix is non-singular. In the limit, however, the sub-matrix Y in (21.21) becomes singular and one really needs (21.12) to gauge the complete problem.
In [19, 87, 256] this problem was solved (in the frequency-domain) by introducing an artificial element in the circuit, a voltage source, of which the applied voltage E_{osc} had to be determined in such a way that the current through this source became 0. In that case the artificial element can be eliminated from the circuit and the solution on the remaining circuit gives the oscillator solution.
It is clear that such a voltage source can only be applied sucessfully at specific locations of the circuit. It is a requirement that for each value $E \neq E_{\text{osc}}$ a unique circuit solution results. When $E \longrightarrow E_{\text{osc}}$, this unique circuit solution has to converge to the oscillator solution. In practice the user has to indicate where the oscillation will be perceived. This is not a drawback, because an IC-designer knows very well to choose a node where the oscillation occurs (as second node one can always use the ground node).
The approaches in [19, 256] were considered more closely in [23, 150]. Here also recommendations for increasing robustness were derived. In [121], a similar approach was followed in the time-domain. We will consider these approaches more closely in the next subsections.

Artificial voltage source in the time-domain The additional voltage source will be put between the nodes a and b. We assume the circuit unknowns to be ordered in such a way that at each time level $x(t) = (\ldots, x^a(t), x^b(t), i(E)(t))^\top$, where $x^a(t), x^b(t)$ are the voltage values at time level t at the nodes a and b respectively, and $i(E)(t)$ is the current through the artificial element $E(a, b)$ [Thus $i(E)$ is the $(N + 1)$-th unknown]. Let $E(a, b)(t_i) = \varepsilon_i$. The ε_i have to be determined in such a way that when the time profile of the voltage difference between the nodes a and b is identical to the time-varying voltage difference of the oscillator solution, the time profile of the current through the element is identically 0. In that case $x(t) = (\ldots, x^a(t), x^b(t), 0)^\top$, in which the part with the first N coordinates is identical to the oscillator solution at time level t. Because the artificial source E is added to the circuit, $i(E)$ does not occur as a controlling electrical variable in the user defined expressions. This implies that on each time level t_i

$$G_i = \begin{pmatrix} & & & \vdots \\ & 0 & & 1 \\ & & & -1 \\ \ldots & 1 & -1 & 0 \end{pmatrix}, \qquad C_i = \begin{pmatrix} & & & \vdots \\ & 0 & & 0 \\ & & & 0 \\ \ldots & 0 & 0 & 0 \end{pmatrix} \tag{21.29}$$

and

$$\frac{\partial j(x(t_i), \varepsilon_i)}{\partial \varepsilon_i} = -1, \qquad \frac{\partial q(x(t_i), \varepsilon_i)}{\partial \varepsilon_i} = 0. \tag{21.30}$$

In addition, the added equation on time level t_i is

$$i(E) \quad = \quad 0 \tag{21.31}$$

When the complete set of unknowns is written as $(x^\top(t_0), \ldots, x^\top(t_{M-1}), f, E^\top)^\top$, in which $E = (\varepsilon_0, \ldots, \varepsilon_{M-1})^\top$, Newton-Raphson can be formulated as

$$\begin{pmatrix} Y^{(k)} & F^{(k)} & \tilde{\mathcal{E}} \\ p^\top & 0 & 0 \\ \mathcal{E} & 0 & 0 \end{pmatrix} \begin{pmatrix} x^{k+1} - x^k \\ f^{k+1} - f^k \\ E^{k+1} - E^k \end{pmatrix} \quad = \quad - \begin{pmatrix} F^D(x^k, f^k, \mathcal{E}^k) \\ p^\top x^k - c \\ \mathcal{E} x^k \end{pmatrix} \tag{21.32}$$

Here

$$F^{(k)} \quad = \quad ((\frac{\partial}{\partial f} F_0^D(x^k, f^k))^\top, \ldots, (\frac{\partial}{\partial f} F_{M-1}^D(x^k, f^k))^\top)^\top \tag{21.33}$$

$$\tilde{\mathcal{E}} \quad = \quad \begin{pmatrix} -\theta e_{N+1} & 0 & \ldots & -(1-\theta) e_{N+1} \\ -(1-\theta) e_{N+1} & -\theta e_{N+1} & & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & & -(1-\theta) e_{N+1} & -\theta e_{N+1} \end{pmatrix} \tag{21.34}$$

$$\mathcal{E} \quad = \quad \mathrm{Diag}(e_{N+1}^\top, \ldots, e_{N+1}^\top) \tag{21.35}$$

Here x and p have length $M(N+1)$. Furthermore Y is a non-singular $M(N+1) \times M(N+1)$ matrix that has a structure like in (21.21)-(21.22), but now based on the matrices G_i and C_i in (21.29), respectively. F is a vector of length $M(N+1)$, $\tilde{\mathcal{E}}$ is a rectangular matrix of size $M(N+1) \times M$, and $\mathcal{E}$ is a rectangular matrix of size $M \times M(N+1)$ [M columns and M row-blocks of length $N+1$ each]. Note that, for $\theta = 1$, one has $\tilde{\mathcal{E}} = -\mathcal{E}^\top$.
Similar to (21.19), the linear system (21.32) can be solved using Block-Gaussian elimination that exploits the LU-decomposition of Y.

It seems natural to add the artificial voltage source to the same node as used for the gauge condition (19.8). This might indicate a possible source for conflicting requirements, because the gauge equation in its most simple form is a voltage or current condition (at a specific time level).

- Let us first consider the situation of a voltage condition. The basic point is that the (artificial) voltages ε_i are part of the Newton process and they will be tuned automatically such that in the limit they will not violate the gauge equation. More detailedly, the $p^\top$ in (21.32) might appear as a row m in Y. Then the corresponding entry F_m is zero. However, in $\tilde{\mathcal{E}}$ we will find a (minus) one in the same row. This causes both rows to be independent. The corresponding ε_m will converge in one iteration, because of the linear dependency.
- Considering the situation of a current condition for the gauge equation, we remark that now $p^\top$ can not occur as row in Y. There is also no conflict with $\tilde{\mathcal{E}}$, because $p^\top$ addresses real circuit unknowns known by the user, while $\mathcal{E}$ addresses the additional (artificial) circuit unknown $i(E)$. Hence $p^\top$ is also independent from the rows of $\mathcal{E}$.

In practice, one will put a resistor R in series with the artificial source E: the complete element $\hat{E}(a, b)$ will act like a (linear) resistor $R(a, a')$ (of value R) and $E(a', b)$. Because of the linearity of $R(a, a')$, we can easily eliminate the unknown $x^{a'}(t)$ from the system. The effective Kirchhoff Voltage Law at time level t_i yields

$$x^a - x^b - Ri(E) - \varepsilon_i \quad = \quad 0 \tag{21.36}$$

The effect is that G_i in (21.29) simply changes into

$$G_i \quad = \quad \begin{pmatrix} & & & \vdots \\ & 0 & & 1 \\ & & & -1 \\ \ldots & 1 & -1 & -R \end{pmatrix} \tag{21.37}$$

The series resistance assures that no artificial voltage-shorts-inductor-loops are generated in the circuit. Note that the equation for $i(E)$ always has a non-zero diagonal element as pivot. In practice, $R = 1$.

Two-step approach. The two-step approach [19, 87, 256] assumes that for given parameters f, E^k, the driven non-linear problem $F^D(x(f, E^k)) = 0$ is solved. For updating f, E^k, Newton-Raphson can be used ("outer loop") in which one can exploit the Jacobian-matrix of the inner Newton-Raphson process for solving $F^D(x(f, E^k)) = 0$

$$\begin{pmatrix} p^\top \frac{\partial x}{\partial f} & p^\top \frac{\partial x}{\partial E} \\ \mathcal{E}\frac{\partial x}{\partial f} & \mathcal{E}\frac{\partial x}{\partial E} \end{pmatrix} \begin{pmatrix} f^{k+1} - f^k \\ E^{k+1} - E^k \end{pmatrix} = -\begin{pmatrix} p^\top x - c \\ \mathcal{E}x \end{pmatrix} \tag{21.38}$$

in which $\partial x/\partial f$ and $\partial x/\partial E$ are obtained by applying an ordinary sensitivity analysis to the inner, driven, problem. Here the Jacobian-matrix $Y = \partial F^D/\partial x$ of the inner Newton-Raphson process is re-used in solving the systems

$$\frac{\partial F^D}{\partial f} + Y\frac{\partial x}{\partial f} = 0, \tag{21.39}$$

$$\frac{\partial F^D}{\partial E} + Y\frac{\partial x}{\partial E} = 0. \tag{21.40}$$

Normal projection of the Newton correction

In [27] the idea is recalled to project the Newton correction for the circuit solution to become perpendicular to the orbit of the solution in some point t_0. Because the time derivative is the tangential derivative, this means that

$$\Delta x(t_0) \quad \perp \quad x'(t_0) \tag{21.41}$$

More generally, we may require that the overall inner product of Δx and $x'(t)$ is zero. This is similar to requiring

$$\sum_i (\Delta x(t_i), x'(t_i)) = 0. \tag{21.42}$$

In [27] (21.41) is used to gauge the free oscillator problem rather than (21.12). Clearly, the algorithm has to find a t_0 where $x'(t_0) \neq 0$ (which will have to be approximated in practice). Near the limit solution this gauge excludes (small) time shifts. However, the algorithm does not exclude the DC-solution.

Initialization

For the driven problem, an initial timeprofile for the Finite Difference Method can be found by applying ordinary transient integration over several periods and collecting results at specific timepoints.

For the oscillator problem, the (Accelerated) Poincaré Method can be used to determine approximations for f and for a circuit solution as well - from these also initial values for the voltages as well as for the gauging value can be used. Note that FD uses a gauge value that may be different from the one used as switch value in Poincaré.
Alternatives can be found from pole-zero analysis and determining the eigenvector solutions of the dominant complex poles.
In the following, we will describe additional options when using Harmonic Balance.

Shooting method

For the Shooting Method (SM), we define $F^S : \mathbb{R}^N \to \mathbb{R}^N$ by

$$F^S(x_0) = x(T),$$

with $x : [0,T] \to \mathbb{R}^N$ the solution of

$$\frac{\mathrm{d}}{\mathrm{d}t}q(x) + j(t,x) = 0, \quad \text{for } 0 \le t \le T, \tag{21.43}$$

$$x(0) = x_0. \tag{21.44}$$

For (single) shooting ('shooting-Newton' in [233]) one has to solve

$$F^S(x_0) - x_0 = 0.$$

Using the Newton method (PSS Newton Process), one needs to evaluate $F^S(x_0)$ and the (monodromy) matrix $\Phi(x_0)$, defined by

$$\Phi(x_0) := \frac{\mathrm{d}F^S(x_0)}{\mathrm{d}x_0} \in \mathbb{R}^{N\times N}.$$

The calculation of $F^S(x_0)$ is in fact the result of a time integration. For instance, applying Euler-backward yields discrete equations at each time level, that are solved by an internal Newton method (*Time-Level Newton Process*):

$$\left(\frac{1}{\Delta t}C_{i+1}^{(m-1)} + G_{i+1}^{(m-1)}\right)(x_{i+1}^{(m)} - x_{i+1}^{(m-1)}) =$$
$$-\{\frac{q(t_{i+1}, x_{i+1}^{(m-1)}) - q(t_i, x_i)}{\Delta t} - j(t_{i+1}, x_{i+1}^{(m-1)})\}. \tag{21.45}$$

Hence, this requires the solution of a system of linear equations with coefficient matrix $\frac{1}{\Delta t}C+G$. In fact this is a familiar process that is available in each conventional analog circuit simulator. The Newton matrix $\Phi(x_0)$ for the PSS Newton Process can be determined using a recursive procedure for the (matrix) quantities $\partial x_i/\partial x_0$

$$\left(\frac{1}{\Delta t}C(t_{i+1}, x_{i+1}) + G(t_{i+1}, x_{i+1})\right)\frac{\partial x_{i+1}}{\partial x_0} = \frac{1}{\Delta t}C(t_i, x_i)\frac{\partial x_i}{\partial x_0}. \tag{21.46}$$

$$\Phi(x_0) = \frac{\partial x(T)}{\partial x_0} = \frac{\partial x_M}{\partial x_0}. \tag{21.47}$$

The matrices C and G are rather sparse in contrast to the matrix $\Phi(x_0)$ that is rather full. In [142, 231–233] the linear equations for the PSS-Newton are solved by means of a matrix-free method, by exploiting a Krylov-space method (GMRES or CGS). Here one needs to determine the result of $\Phi(x_0)p$, for some vector p, in order to extend the Krylov space. This can elegantly be done by a similar recursive procedure as above in (21.46), but now for a sequence of vectors. The charm of this recursion is that it re-uses the existing LU-decompositions of the Time-Level Newton Process; in addition the matrices C are needed. For GMRES a final least squares problem has to be solved. In fact, this has to be done in some flat-matrix structure. Assuming a k-dimensional Krylov space, the least squares problem is of order kN, where N is the number of unknowns in a flat circuit.

We collect some differences between the Shooting Method and the Finite Difference Method.

- In contrast to the Finite Difference Method, for the Shooting Method the time discretization can be chosen adaptively in a natural way, using the ordinary transient integration.
- The Shooting Method only needs an initial value to start from. But it may diverge rather fast in case of poles that cause instabilities.
- The FDM always assures periodicity for each iterand; in the limit also the discretized equations are satisfied. To contrast: each iterand of the Shooting Method satisfies the discretized equations, while reaching periodicity is the target of the method.
- In practice, the FDM is more stable than the Shooting Method, but - per iterand - it is much slower. The stability properties of shooting methods can be increased by applying multiple shooting that can be applied to the free oscillator problem as well [256].

The higher index case. From the remark concerning the higher index case at the end of Section 20, it follows that the shooting matrix should start from consistent initial values. Fortunately it can be shown [172], that it is sufficient to start the calculation of $F^S(x_0)$ with 1(2) Backward Euler steps in case of DAE index 1(2). Alternatively, if the independent eigenvectors of the shooting matrix are known in the beginning, it is sufficient to calculate only the rectangular part comprising them [8, 172, 220].

Improving global convergence. Since the region of attraction for the PSS Newton Process is usually fairly small (see at the remarks concerning the differences between SM and FDM above), it is desirable to apply continuation methods which ensure global convergence under not too restrictive assumptions. A key issue here is that along the continuation path no bifurcations occur, which would make it difficult to track the "proper" solution branch. In [8] it is argued that this is best possible with an artificial homotopy

$$\rho(x, \lambda, a) := \lambda \cdot (F^S(x) - x) + (1 - \lambda) \cdot (x - a),$$

where λ, $0 \leq \lambda \leq 1$, is the homotopy parameter, and a is a start vector for the homotopy.
Using a theorem of Sard (see e. g. [37]) it is shown in [8] that under some reasonable assumptions for circuit models up to DAE index 2 the continuation path is smooth for almost any initial value a. So it can be traced with some kind of predictor-corrector techniques, starting from $\lambda = 0$ until the desired fixpoint $F^S(x) = x$ is obtained for $\lambda = 1$. The start vector a is obtained here from running a standard transient analysis over one cycle. For autonomous systems a gauging phase condition is added, while the frequency is an additional unknown.

Waveform Newton

In [137] the Waveform Newton Method has been described (for solving forced problems). Here one linearizes each time around a previously calculated periodic waveform $x^{(i)}$. This results in a linear DAE for the correction, in which the coefficients are periodic and depend on the last calculated waveform. From this we derive, that the next iterand $x^{(i+1)}$ satisfies

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}[C(x^{(i)})x^{(i+1)}] + G(t, x^{(i)})x^{(i+1)} \quad &= \\ -\{\frac{\mathrm{d}}{\mathrm{d}t}[q(x^{(i)}) - C(x^{(i)})x^{(i)}] + [j(t, x^{(i)}) - G(t, x^{(i)})x^{(i)}]\} \end{aligned} \tag{21.48}$$

Similar to the Shooting Method case one can solve this linear DAE easily for an initial value of $x^{(i+1)}$ such that we have periodic solution. Note that we can start with a non-periodic waveform. All next iterands will automatically be periodic.
One can show, that, on a fixed grid, the Finite Difference Method and the above approach can generate the same solutions. However, the above approach, using the Shooting Method, allows to use adaptive integration. In this way both nice features of FDM (always periodic iterands) and of SM (adaptivity) are combined. As in FDM, each iterand is periodic, but only the limit satisfies the differential equations.
A nice feature is that the algorithm very elegantly extends to a Periodic AC analysis (see Section 20).

Harmonic Balance

Harmonic Balance (HB) is a non-linear frequency-domain method for determining a periodic steady-state solution. The Fourier coefficients of the PSS are the solution of a non-linear algebraic system of equations, that is usually solved by applying Newton's method. In the next we describe the method in some detail.
We assume d independent fundamental (angular) frequencies λ_j. Let (.,.) denote the complex inner-product and Z be the set of integers. We write x (and similarly j and q) in an expansion

of complex exponentials

$$x = \sum_{\omega_k \in \Lambda} X_k e^{\iota \omega_k t}, \text{ with } \omega_k \in \Lambda \equiv \{\omega |\, \omega = (k, \lambda)\}, \tag{21.49}$$

$$k \equiv (k_1, k_2, \ldots, k_d)^\top \in K \subset Z^d, \ \lambda \equiv (\lambda_1, \lambda_2, \ldots, \lambda_d)^\top, \ \lambda_i > 0, \tag{21.50}$$

where the (complex) X_k satisfies $X_{-k} = \overline{X}_k$.
Here λ and k are uniform for each component of x. The set K, containing integer tuples, is symmetrical about 0, while also $0 \in K$. K is assumed to be finite. With K we denote the number of non-negative (angular) frequencies (i.e. ω_k with $\omega_k \geq 0$). We also assume that all ω_k are different and $\omega_0 = 0$.
The choice of the fundamental frequencies λ_j will depend on the kinds of (modified) sine-wave sources used. We note that Λ should contain a sufficiently rich set of interdistortion frequencies ω_k like $2\lambda_1$, $\lambda_1 \pm \lambda_2$, that are required in a distortion analysis. In practice, too restrictive a choice of the finite set of K may give rise to aliasing problems when compared with the analytical problem. For some sine-wave sources, a 1-D set of frequencies will be sufficient.
Let $X = (X^1, X^2, \ldots, X^N)^\top$ be the Fourier transform of x. More specifically, using a real notation, $X^j = (X_0^{j,R}, [X_1^{j,R}, X_1^{j,I}], \ldots, [X_{K-1}^{j,R}, X_{K-1}^{j,I}])^\top$, in which $X_k^j \equiv X_k^{j,R} + \iota X_k^{j,I}$ represents the k-th Fourier coefficient of x^j.
With $\mathcal{F}$, we denote the mapping of the Fourier transform, thus $X = \mathcal{F}x$ and $x = \mathcal{F}^{-1}X$. The $\mathcal{F}$-transform of j (and similarly for q) is defined by $J(X) = \mathcal{F}j(\mathcal{F}^{-1}X) = \mathcal{F}j(x)$. By this Galerkin approach, the frequency-domain equivalent of (19.1) simply becomes

$$J(X) + \Omega Q(X) = 0, \quad \text{in which} \tag{21.51}$$

$$\Omega = \text{Block\_Diag}(\Omega_K, \ldots, \Omega_K), \tag{21.52}$$

$$\Omega_K = \text{Block\_Diag}(0, \begin{vmatrix} 0 & -\omega_1 \\ \omega_1 & 0 \end{vmatrix}, \ldots, \begin{vmatrix} 0 & -\omega_{K-1} \\ \omega_{K-1} & 0 \end{vmatrix}). \tag{21.53}$$

In the terminology of circuit analysis, the method of solving (19.1) by solving (21.51) is called the Harmonic Balance method. It is clear that the system given by (21.51) is a non-linear algebraic set of equations in the frequency-domain.
In general, the system is solved by performing a Newton-Raphson iteration. A DC-analysis provides an initialization for the basic harmonic. For the other harmonics one can solve a set of AC-problems in parallel, each being linearised around the same DC-solution. Because each AC-problem is linear, this is very efficient. Note that this approach may be interpreted as the first iteration of a non-linear block Gauss-Jacobi approach, using partitions between the components of different harmonics.
Clearly, in HB, a Newton-Raphson matrix is much larger than in ordinary DC or Transient Analysis. However, it still has a similar sparse structure as in the last two cases. Hence it is not surprisingly that quite some attention is made in literature concerning iterative methods applied to the linear system of equations arising in Harmonic Balance [18, 19, 166, 199, 200, 202].

Sources. The choice of the fundamental frequencies λ_j depends on the kinds of sources used. For standard amplitude, frequency or phase modulated sources, a 2-D block of frequencies will usually be necessary, as explained below.
We assume voltage and current sources. The DC-sources are time-independent, the AC-sources may involve a simple sum of (co)sine-waves (SW-source). For Harmonic Balance the sources may also show amplitude modulation (SWAM), frequency modulation (SWFM) or phase modulation (SWPM) behaviour. Denoting a source by $s(t)$ and the carrier frequency and the signal frequency by ω_c and ω_s, respectively, the following cases can be distinguished (here θ simply denotes a phase shift).

Modulation	$x(t) = A(t)\cos(\psi(t) + \theta)$		$K_{\min} = \text{block}[n, m]$
	$A(t)$	$\psi(t)$	
AM	$a + b\sin(\omega_s t)$	$\omega_c t$	[1,1]
FM	a	$\int_0^t \omega_c + c\cos(\omega_s t)\ \mathrm{d}t$	[1,m], $m \geq c/\omega_s$
PM	a	$\omega_c t + d\sin(\omega_s t)$	[1,m], $m \geq d$

In the last column we have added the minimum rectangular subset of K in order to avoid obvious errors due to aliasing (assuming $\lambda_1 = \omega_c$, $\lambda_2 = \omega_s$). In general $\omega_c \gg \omega_s$. The following result, of which the proof is elementary, will be used in the sequel: *SWAM, SWFM and SWPM sources have a Fourier expansion with respect to the exponentials* $e^{\iota(n\omega_c + m\omega_s)t}$. *For SWAM and SWPM the coefficients are independent of* ω_c *and* ω_s. *For SWFM the coefficients depend on* c/ω_s.

A more careful evaluation of the coefficients reveals that the only non-zero harmonics are for $(k_1, k_2) = (1, m)$. For SWAM m is also restricted to $m \leq 1$, showing that a finite expansion is obtained. For SWFM and SWPM the dominant part of the infinite expansions depends on c/ω_s and d, respectively.

Discrete Fourier Transform. Let $\mathcal{F}_{\eta,\mu}$ denote the Fourier transform using fundamental frequencies η, μ. We observe that in general, by the non-linearity of i and q, $I(V)$ and $Q(V)$ depend on the specific λ_1, λ_2 mentioned previously. However, in practice, $I(V)$ and $Q(V)$ appear to be rather independent on λ_1, λ_2. This surprising phenomenon allows an efficient evaluation of $I(v)$ and $q(v)$. To be more specific, let λ_3, λ_4 be two other fundamental frequencies that satisfy the same assumptions as imposed on λ_1, λ_2, i.e. the corresponding set of frequencies generated by K and λ_3, λ_4 should not contain multiple values. In practice the non-linearity in i (and similarly in q) with respect to v is only 'algebraic' in the following sense

$$I(V)_k = (\mathcal{F}_{\lambda_1,\lambda_2} i([\mathcal{F}_{\lambda_1,\lambda_2}]^{-1} V))_k, \tag{21.54}$$

$$= (\mathcal{F}_{\lambda_3,\lambda_4} i([\mathcal{F}_{\lambda_3,\lambda_4}]^{-1} V))_k, \tag{21.55}$$

for all $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, which means that the Fourier coefficients are frequency independent. Non-linear resistors are algebraic in the above sense when using the variables i and v; non-linear capacitors when dealing with q and v (note that $i = \mathrm{d}q/\mathrm{d}t$); and non-linear inductors when dealing with ϕ and i (note that $v = \mathrm{d}\phi/\mathrm{d}t$). The expansions in $e^{\iota(n\omega_c + m\omega_s)t}$ of the SWAM, SWFM, SWPM sources show that they also exhibit this algebraic behaviour.
The algebraic non-linearity offers a way to exploit λ_3, λ_4 which are different from the fundamental analysis frequencies λ_1, λ_2, in determining $I(V)$ and its partial derivatives in an efficient and stable way using the Discrete Fourier Transform. For details we refer to [18, 139, 199, 234]. In [219] several bijective mappings between (enveloping sets of) higher dimensional spectral sets K and a 1-dimensional equivalent (with no gaps) are considered that allow for proper usage of the DFT.

Numerical aspects of Harmonic Balance Although HB is being successfully used for a wide range of applications, there are still some mathematical issues which have to be solved. One of them is error control and adaptivity, another one concerns DAE aspects.

- Accuracy of the HB solution is mainly determined by the sets k and λ in (21.50), which have to be provided by the user, or are determined from the type of sources, as is decribed above. In case of too few — or a not adequate set of — frequencies, alias effects may occur, or HB does not even converge. Harmonic Balance is useful only for mildly non-linear problems, i.e. when all quantities have a Fourier expansion that can be well approximated by some finite one of limited length. Aliasing can be reduced by applying oversampling [234]. A rigorous mathematical adaptivity concept is not implemented, in general. A proposal is given in [76]; unfortunately, adaptivity here involves to reorganize the system matrix from time to time, which does not fit well into existing implementations.
- In practice, HB has been applied successfully even for index-2 problems, and no severe drawbacks or errors have been reported yet. There are however no theoretical investigations about the feasibility of this usage, and which impact may have a higher index on numerics.

HB Oscillator Algorithm. In [19, 87, 150, 256] oscillator algorithms are given for Harmonic Balance that resemble the approach described in Section 21 for the time domain. However, there are some modifications:

- The gauge condition is replaced by the requirement that the imaginary part of the first harmonic at some predefined node has to be zero. Note that this allows the DC solution to be solution of the system. Indeed, the DC solution appears to be a strong attractor for the Newton process. Hence algorithms apply some additional deflation technique to exclude this solution.
- In practice the artificial element is defined directly in the frequency domain. For all harmonics but the first one the element acts as an 'open', i.e. the harmonic of the current is set to 0. For the first harmonic it acts as a voltage source in series with a resistor. For all analyses other than Harmonic Balance (that might be used for initialization), the current through the element is set to zero too.
- For initialization the equations are linearized around the DC-solution like in AC analysis. Kurokawa's method [143] calculates the response solution for an ordinary sinusoidal source with unit amplitude that replaces the artificial element and considers the admittance for the source element. The result is considered as a function of the frequency f. Where the imaginary part of the admittance becomes zero while the real part remains positive a good approximation for the oscillator can be found (the equation itself can be solved by applying for instance Newton-Raphson). For the circuit solution one uses the DC-solution plus the AC solution for the first harmonic. All other harmonics are set to 0.
  Alternatively one can solve a generalized eigenvalue problem (in practice one will consider the inverse eigenvalue problem) for the linearized equations [23]. Because an autonomous circuit can only start up oscillating when the DC-solution is unstable (Andronov-Hopf bifurcation theorem), one looks for eigenvalues $\lambda = \delta \pm j\omega$, where $\delta > 0$. The associated eigenvector also indicates where the artificial element may be attached. In addition it provides an estimate for the initial circuit solution. However, in practice, the f estimated by Kurokawa's method appears to be more accurate.
- The applied value of the artificial element is initialized by optimization techniques [87, 129, 256]. In [23, 150] techniques using affine damping improved convergence. Initial Global Optimization techniques improved robustness even more by providing much better initial estimates [151]. Note that the algorithm can be formulated as a full Newton process, but also as a two-step process. In the latter case for each applied value as internal step a driven Harmonic Balance process is executed until convergence. For updating the applied value and the frequency, the Jacobian matrix of the Harmonic Balance process can be reused for determining the sensitivities of the solution with respect to variations of the applied value and the frequency. In fact, this very elegantly reuses options for parameter sensitivity analysis.

Global convergence – a three stage approach. For getting global convergence properties, the application of a continuation method is adequate. In case of non-autonomous (forced) systems it is natural for this purpose to track a parameter dependent path in the frequency domain with some path-following methods, as is done in the DC domain by performing a DC transfer analysis. The parameter here may be a circuit parameter or the bias value of an independent source, either. In case of local parametrization along the solution path, even turning points in the parameter space can be tracked; and by watching the sign and magnitude of Floquet multipliers, circuit stability properties along the solution path can be analyzed [220].

For autonomous oscillators the problem is more difficult since a good estimate for the frequency is important. So it is suggested in [171] for this case to start path-following from Hopf's bifurcation point. The latter is computed from a path-following procedure in the DC-domain, such that there is a three-stage approach for solving the whole problem:

1. Follow a path of DC steady states over a parameter λ — λ being a circuit parameter or the value of an independent source — until a Hopf bifurcation point is found. The latter is characterized by a sign change of the real part of a complex conjugate pair of generalized eigenvalues.
2. These eigenvalues and the corresponding eigenvectors provide first order information about frequency and Fourier coefficients of the oscillatory branch emanating from the

DC path in Hopf's point. Since this information is not very accurate, an alternative method for getting the latter was developed [269].

3. From this start point, follow the path of periodic steady states over λ, until the final value of λ is obtained.

Again it is worth to note that all these additional algorithmic steps can be implemented elegantly using Schur complement techniques, once the basic types of analysis are available.
Recently, homotopy approaches were considered in [22, 157].

Envelope methods

In [239] an envelope method is described in some detail. The envelope method is a generalisation of the Harmonic Balance method. It allows for multitone treatments, but in fact one of the periods may be infinite. The method mixes time-domain and frequency-domain approaches.
We write (21.49) as

$$x(t) \quad = \quad \sum_{\omega_k \in \Lambda} X_k(t) e^{\iota \omega_k t}, \text{ with } \omega_k \in \Lambda \quad \equiv \quad \{\omega |\, \omega = (k, \lambda)\}, \tag{21.56}$$

where the enveloping Fourier coefficients $X_k(t)$ represent modulation on top of the carrier sinusoids at frequencies ω_k. In order to describe the effect of $q(x)$ (and similarly of $j(x)$) we introduce

$$\tilde{x}(\tau_1, \tau_2) \quad = \quad \sum_{\omega_k \in \Lambda} X_k(\tau_1) e^{\iota \omega_k \tau_2}, \tag{21.57}$$

$$= \quad \mathcal{F}_{\tau_2}^{-1} X(\tau_1) \tag{21.58}$$

(which in fact is a multivariate formulation; see also the next subsection). Here $X(\tau_1)$ is the Fourier transform in τ_2 of $\tilde{x}(\tau_1, \tau_2)$. Thus $x(t) = \tilde{x}(t, t)$. For fixed τ_1, the Fourier coefficients $Q_k(x(\tau_1))$ of q, when applied to $\tilde{x}(\tau_1, \tau_2)$, can be determined using the Fourier Transform in the τ_2 variable

$$q(\tilde{x}(\tau_1, \tau_2)) \quad = \quad \sum_{\omega_k \in \Lambda} Q_k(X(\tau_1)) e^{\iota \omega_k \tau_2}, \tag{21.59}$$

$$Q_k(X) \quad = \quad \mathcal{F}_{\tau_2} q(\mathcal{F}_{\tau_2}^{-1} X). \tag{21.60}$$

Clearly, $q(x(t)) = q(\tilde{x}(t, t))$. We will collect all $Q_k(X)$ in $q(x)$ (and similarly for $J(X)$). If we put the expansions of q and j in (19.1) we find a DAE for the $X(t)$

$$J(X(t)) + \frac{\mathrm{d}}{\mathrm{d}t} Q(X(t)) + \Omega Q(X)(t) \quad = \quad 0, \tag{21.61}$$

which can be solved using ordinary time integration methods. Stepping forward in time at each time level a non-linear set of (complex-valued) algebraic equations has to be solved, that has the size of a Harmonic Balance problem. In RF applications, the envelope solution of the DAE (21.61) behaves much less oscillating (or is not oscillating at all) than that of (19.1). Hence, despite the larger non-linear system of equations that has to be solved at each time level for (21.61), much larger time steps can be used than for (19.1).
Because of the separation of modes in τ_1 and τ_2 variables, different scaling effects can be separated. This is also the subject of the next subsection.

Analysis of high-quality oscillator circuits. Another kind of envelope following methods has been suggested for analysis of oscillatory circuits whose quality factor Q is so high that conventional methods like those described in [184, 224] failed. These circuits exchange a very small amount of energy per cycle between the oscillator core and the driven, energy supplying circuit part, which makes the problem extremely stiff. In a state space diagram, the trajectories of one cycle are almost closed, even though the circuit is not yet in a steady state. So it seems

reasonable to approximate the trajectory for this cycle by a really closed one, which can be computed by solving a PSS problem. Once this approximative trajectory is found, a transient analysis over a few cycles would correct this one into the real solution. From its dynamics a new estimate for a later cycle can be extrapolated, and the next step of this "envelope" method can be started [268]. In fact this method is again a mixed time-frequency approach.

A successful application of this idea is the startup analysis of quartz driven circuits [163, 218]. Since the quartz resonator oscillates very much like a harmonic oscillator, it can be substituted for one PSS step by a sinusoidal current source of a certain magnitude. Its phase can be arbitrarily set to zero, and a first guess for the frequency is just the resonator frequency of the quartz crystal. Once the PSS solution is found, and is "corrected" by a subsequent transient analysis over a few — 2, ..., 4, say, — cycles, the dynamic behaviour can be extrapolated over several hundred to thousand cycles, yielding a new value for the magnitude and the frequency of the substitute current source. So this method cannot only be seen as some kind of continuation method for the PSS problem, but also yiels reasonable timing information about the startup process of the circuit.

Multivariate extension

In [20, 24, 201, 203–205] multivariate extensions are described that apply to multitone situations. In fact, one introduces two or more independent time parameters, τ_1,τ_2 say. Then (19.1) is rewritten to

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}\tau_1} q(\hat{x}) + \frac{\mathrm{d}}{\mathrm{d}\tau_2} q(\hat{x}) + j(\hat{x}) &= b(\tau_1, \tau_2) \in \mathbb{R}^N. && (21.62) \\ \hat{x}(0, \tau_2) &= \hat{x}(T_1, \tau_2) && (21.63) \\ \hat{x}(\tau_1, 0) &= \hat{x}(\tau_1, T_2). && (21.64) \end{aligned}$$

After solving this partial differential problem (21.62) (hyperbolic for $\mathrm{d}q/\mathrm{d}x$ regular) on $[0, T_1] * [0, T_2]$ for $\hat{x}$, the solution $x(t)$ is found by $x(t) = \hat{x}(t \pmod{T_1}, t \pmod{T_2})$.
It is clear that the above separation in two or more independent time parameters restricts one in formulating expressions. The aim is that on $[0, T_1] * [0, T_2]$ the solution $\hat{x}$ behaves smoothly and that only one period is met in each direction. In [205] the problem of frequency modulation (FM) is considered more closely for the case of an oscillatory DAE

$$\begin{aligned} \omega(\tau_2) \frac{\mathrm{d}}{\mathrm{d}\tau_1} q(\hat{x}) + \frac{\mathrm{d}}{\mathrm{d}\tau_2} q(\hat{x}) + j(\hat{x}) &= b(\tau_2). && (21.65) \\ \phi(t) &= \int_0^t \omega(\tau_2) \mathrm{d}\tau_2 && (21.66) \\ x(t) &= \hat{x}(\phi(t), t) && (21.67) \end{aligned}$$

When (21.65) is solved, also the local frequency $\omega(\tau_2)$ is obtained (see also [24]). The derivative, $\omega(\tau_2)$, of the 'warping' function ϕ, gives the extend of the stretch of the timescale in τ_2.
For time integration methods of characteristics were studied recently [24, 186].

Optimal sweep following. An open question remains how $\omega(\tau_2)$ in (21.65) should be determined. One way to proceed is to observe that the differential equation (21.65) defines a two-dimensional manifold (called the *sweep*) in the state space $\mathbb{R}^N$. The choice of ω does not influence the sweep; however, it does influence the parametrisation of the sweep in terms of the coordinates τ_1 and τ_2.
In [**?**], it is suggested to choose ω in such a way that

$$\int_0^T \| \frac{\mathrm{d}}{\mathrm{d}\tau_2} q(\hat{x}) \|^2 \, \mathrm{d}\tau_1 \qquad (21.68)$$

becomes as small as possible. The rationale is that this will allow the largest stepsizes in the (slowly varying) τ_2-direction, thereby reducing computation time. It is shown in [**?**] that this

is the case for

$$\dot{\phi}(\tau_2) \quad = \quad \omega(\tau_2) \quad = \quad \frac{\int_0^T \left(b(\tau_2) - j(\hat{x}), \frac{\mathrm{d}}{\mathrm{d}\tau_2} q(\hat{x}) \right) \, \mathrm{d}\tau_1}{\int_0^T \| \frac{\mathrm{d}}{\mathrm{d}\tau_2} q(\hat{x}) \|^2 \, \mathrm{d}\tau_1}. \tag{21.69}$$

Since this choice of ω is optimal with respect to the minimization of (21.68), the resulting method is called *Optimal Sweep Following*.

Bibliography

[1] ANTOGNETTI, P.; MASSOBRIO, G. [1987]: Semiconductor Device Modeling with SPICE, McGraw-Hill, New York.

[2] ANZILL, W.; KÄRTNER, F.X.; RUSSER, P. [1994]: Simulation of the phase noise of oscillators in the frequency domain, Int. J. Electron. Commun. (AEÜ) 48, 45–50.

[3] APPEL, T. (2000): A new timestep control in the circuit simulation package TITAN, Bachelor thesis, Technical University München, München.

[4] ARNOLD, M. [1997]: Zur Theorie und zur numerischen Lösung von Anfangswertproblemen für differentiell-algebraische Systeme von höherem Index, VDI-Verlag, Düsseldorf.

[5] ARNOLD, M.; GÜNTHER, M. [2001]: Preconditioned dynamic iteration for coupled differential-algebraic systems, BIT 41, 1–25.

[6] ASCHER, U.M.; MATTHEIJ, R.M.M.; RUSSELL, R.D. [1998]: Numerical Solution of Boundary Value Problems for Ordinary Differential Equations, Prentice Hall, Englewood Cliffs.

[7] ASCHER, U.M.; PETZOLD, L.R. [1998]: Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations, SIAM, Philadelphia.

[8] BAIZ, A. (2003): Effiziente Lösung periodischer differential-algebraischer Gleichungssysteme in der Schaltungssimulation, PhD thesis, TU Darmstadt, Darmstadt.

[9] BANK, R.E.; COUGHRAN, W.M.; FICHTNER, W; GROSSE, E.H.; ROSE, D.; SMITH, R.K. [1985]: Transient simulation of silicon devices and circuits, IEEE Trans. Comp. Aided Des. CAD 4, 436–451.

[10] BARTEL, A. [2000]: Generalised multirate — Two ROW-type versions for circuit simulation, Unclassified NatLab Report 804/2000, Philips Research Laboratories, Eindhoven.

[11] BARTEL, A.; GÜNTHER, M.; KVÆRNØ, A. [2001]: Multirate methods in electrical circuit simulation, Numerics Report 2/2001, Norwegian University of Science and Technology, Trondheim.

[12] BAUER, R.; FANG, A.; BRAYTON, R. [1988]: XPSim: A MOS VLSI simulator, Proc. ICCAD'88, Santa Clara, 66–69.

[13] BOMHOF, W.; VAN DER VORST, H.A. [2000]: A parallel linear system solver for circuit simulation problems, Num. Lin. Algebra Appl. 7, 649–665.

[14] BOMHOF, W. (2001): Iterative and parallel methods for linear systems with applications in circuit simulation, PhD thesis, Utrecht University, Utrecht.

[15] BOMHOF, W.; VAN DER VORST, H.A. [2001]: A parallelizable GMRES-type method for p-cyclic matrices with applications in circuit simulation, in Proc. SCEE-2000, Warnemünde, Lecture Notes Comp. Sci. 18 (Editors: U. VAN RIENEN et al.), pp. 293–300, Springer, Berlin.

[16] BORCHARDT, J.; GRUND, F.; HORN, D. [1997]: Parallelized numerical methods for large systems of differential-algebraic equations in industrial applications, Preprint 382, Weierstrass Institut für Angewandte Analysis und Stochastik, Berlin.

[17] BOWERS, J.C.; SEDORE, S.R. [1971]: SCEPTRE: A Computer Program for Circuit and System Analysis, Prentice-Hall, Englewood Cliffs.

[18] BRACHTENDORF, H.G. [1994]: Simulation des eingeschwungenen Verhaltens elektronischer Schaltungen, PhD thesis, Universität Bremen, Bremen (ISBN 3-8265-0226-4, Shaker, Aachen).

[19] BRACHTENDORF, H.G.; WELSCH, G.; LAUR, R. [1995]: A simulation tool for the analysis and verification of the steady state of circuit designs, Int. J. Circ. Theory Appl. 23, 311–323.

[20] BRACHTENDORF, H.G.; WELSCH, G.; LAUR, R.; BUNSE-GERSTNER, A. [1996]: Numerical steady state analysis of electronic circuits driven by multi-tone signals, Electrical Engineering 79, 103–112.

[21] BRACHTENDORF, H.G.; WELSCH, G.; LAUR, R. [1998]: A time-frequency algorithm for the simulation of the initial transient response of oscillators, Proc. ISCAS'98, Monterey, Vol. I, 236–239. ???? stimmt Vol I ????

[22] BRACHTENDORF, H.G.; MELVILLE, R.; FELDMANN, P.; LAMPE, S. [2002]: Steady state calculation of oscillators using continuation methods, Proc. Design Automation and Test in Europe (DATE 2002), Paris, France, 1139.

[23] BRACHTENDORF, H.G.; LAMPE, S.; LAUR, R. [2000]: Comparison of the Philips and Bremen algorithm for calculating the steady state of autonomous oscillators, ITEM Report, Universität Bremen, Bremen.

[24] BRACHTENDORF, H.G.; LAUR, R. [2000]: Analyse des transienten Verhaltens von Oszillatoren durch eine inverse Charakteristikenmethode, Proc. ITG Workshop Mikroelektronik für die Informationstechnik, Darmstadt, 29–34, ISBN 3-8007-2586-X, VDE, Berlin.

[25] BRACHTENDORF, H.G.; LAUR, R. [2001]: On consistent initial conditions for circuit's DAEs with higher index, IEEE Trans. Circ. Syst. CAS I 48, 606–612.

[26] BRENAN, K.E.; CAMPBELL, S.L.; PETZOLD, L.R. [1996]: Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations, SIAM, Philadelphia.

[27] BRAMBILLA, A.; D'AMORE, D.; SANTOMAURO, M. [1995]: Simulation of autonomous circuits in the time domain, Proc. ECCTD'95, Istanbul, 399–402.

[28] BRAMBILLA, A.; MAFFEZONI, P. [2000]: Envelope following method for the transient analysis of electrical circuits, IEEE Trans. Circ. Syst. CAS I 47, 999–1008.

[29] BRUGNANO, L.; TRIGIANTE, D. [1998]: Solving differential problems by multistep initial and boundary value methods, Gordon and Breach Science Publishers

[30] BRUIN, S.M.A. [2001]: Modified extended BDF applied to circuit equations, MSc-Thesis Free Univ. of Amsterdam, NatLab. Unclassified Report 2001/826, Philips Research Laboratories, Eindhoven.

[31] CALAHAN, D.A. [1968]: A stable, accurate method of numerical integration for nonlinear systems, Proceedings of the IEEE 56, 744–????.

[32] CALAHAN, D.A. [1972]: Computer Aided Network Design, McGraw-Hill, New York.

[33] CASH, J.R. [1983]: The integration of stiff initial value problems in ODEs using Modified Extended Backward Differentiation Formulae, Comp. & Maths. with Appls, Vol 9-5, 645–657.

[34] CASH, J.R. [2000]: Modified extended backward differentiation formulae for the numerical solution of stiff initial value problems in ODEs and DAEs, J. Comput. and Appl. Mathematics, Vol. 125, 117–130.

[35] CHANG, C.R.; STEER, M.B.; MARTIN, S.; REESE JR., E. [1991]: Computer-aided analysis of free-running microwave oscillators, IEEE Trans. Microwave Theory Techniques 39, 1735–1745.

[36] CHAWLA, B.R.; GUMMEL, H.K.; KOZAK, P. [1975]: MOTIS — An MOS timing simulator, IEEE Trans. Circ. Syst. CAS 22, 901–910.

[37] CHOW, S.; MALLET-PARET, J.; YORKE, J. [1978]: Finding zeros of maps: Homotopy methods that are constructive with probability one, Math. of Computation 32, 887–889.

[38] CHUA, L.O.; LIN, P.-M. [1975]: Computer-Aided Analysis of Electronic Circuits: Algorithms & Computational Techniques, Prentice-Hall, Englewood Cliffs.

[39] CHUA, L.O. [1984]: Nonlinear circuits, IEEE Trans. Circ. Syst. CAS 31, 69–87.

[40] COX, P.F.; BURCH, R.G.; YANG, P.; HOCEVAR, D.E. [1989]: New implicit integration method for efficient latency exploitation in circuit simulation, IEEE Trans. Comp. Aided Des. CAD 8, 1051–1064.

[41] COX, P.F.; BURCH, R.G.; HOCEVAR, D.E.; YANG, P.; EPLER, B.D. [1991]: Direct circuit simulation algorithms for parallel processing, IEEE Trans. Comp. Aided Des. CAD 10, 714–725.

[42] DEBEVFE, P.; ODEH, F.; RUEHLI, A.E. [1985]: Waveform techniques, in Circuit Analysis, Simulation and Design, Part 2 (Editor: A.E. RUEHLI), pp. 41–127, North Holland, Amsterdam.

[43] DE MAN, H.; ARNOUT, G.; REYNAERT, P. [1981]: Mixed mode circuit simulation techniques and their implementation in DIANA, in Computer Design Aids for VLSI Circuits, NATO Advanced Study Institute series E Vol. 48 (Editors: P. ANTOGNETTI et al.), pp. 113–174, Martinus Nijhoff, The Hague.

[44] DE MICHELI, G.; HSIEH, H.Y.; HAJJ, I.N. [1987]: Decomposition techniques for large scale circuit analysis and simulation, in Circuit Analysis, Simulation and Design, Part II (Editor: A.E. RUEHLI), pp. 1–39, North Holland, Amsterdam.

[45] DEMIR, A.; LIU, E.W.Y.; SANGIOVANNI-VINCENTELLI, A. [1996]: Time-domain non-Monte Carlo noise simulations for nonlinear dynamic circuits with arbitrary excitations, IEEE Trans. Comp. Aided Des. CAD 15, 493–505.

[46] DEMIR, A. [1998]: Phase noise in oscillators: DAEs and coloured noise sources, Proc. ICCAD'98, San Jose, 170–177.

[47] DEMIR, A.; SANGIOVANNI-VINCENTELLI, A. [1998]: Analysis and Simulation of Noise in Nonlinear Electronic Circuits and Systems, Kluwer, Boston.

[48] DEMIR, A.; LONG, D.; ROYCHOWDHURY, J. [2000]: Computing phase noise eigenfunctions directly from Harmonic Balance/shooting matrices, Proc. ICCAD'2000, San Jose, 283–288.

[49] DEMIR, A.; MEHROTRA, A.; ROYCHOWDHURY, J. [2000]: Phase noise in oscillators: A unifying theory and numerical methods for characterization, Proc. DAC'98, San Francisco, 26–31 (extended version: IEEE Trans. Circ. Syst. CAS I 47, 655–674).

[50] DENK, G. [1990]: An improved numerical integration method in the circuit simulator SPICE2-S, in Proc. Oberwolfach Conf., Int. Ser. Num. Math. 93 (Editors: R.E. BANK et al.), pp. 85–99, Birkhäuser, Basel.

[51] DENK, G.; RENTROP, P. [1991]: Mathematical models in electric circuit simulation and their numerical treatment, in Proc. NUMDIFF5 (Editor: K. STREHMEL), pp. 305–316, Teubner, Leipzig.

[52] DENK, G. [2002]: Private communication.

[53] DE SMEDT, B.; GIELEN, G. [1997]: Accurate simulation of phase noise in oscillators, Proc. ESSCIRC'97, Southampton, 208–211.

[54] DEUFLHARD, P; BORNEMANN, F. [2002]: Numerische Mathematik II De Gruyter, Berlin.

[55] DIRKS, H.K.; FISCHER, M.; RÜDIGER, J. [2001]: Parallel algorithms for solving linear equations in VLSI circuit simulation, in Proc. SCEE-2000, Warnemünde, Lecture Notes Comp. Sci. Eng. 18 (Editors: U. VAN RIENEN et al.), pp. 301–308, Springer, Berlin.

[56] DUNLOP, A.; DEMIR, A.; FELDMANN, P.; KAPUR, S.; Long, D.; Melville, R.; Roychowdhury, J. [1998]: Tools and methodology for RF IC design, Proc. DAC'98, San Francisco, 414–420.

[57] DUTTON, R.W.; TROYANOVSKY, B.; YU, Z.; ARNBORG, T.; ROTELLA, F.; MA, G.; SATO-IWANAGA, J. [1997]: Device simulation for RF applications, Proc. IEDM'97, Washington D.C.

[58] EICKHOFF K.M. (1991): Effiziente Methoden zur Simulation grosser MOS-Schaltungen, PhD thesis, Rheinisch-Westfälische Technische Hochschule, Aachen.

[59] EICKHOFF, K.M.; ENGL, W. [1995]: Levelized incomplete LU-factorization and its application to large-scale circuit simulation, IEEE Trans. Comp. Aided Des. CAD 14, 720–727.

[60] ENGL, W.L.; LAUR, R.; DIRKS, H.K. [1982]: MEDUSA — A simulator for modular circuits, IEEE Trans. Comp. Aided Des. CAD 1, 85–93.

[61] ENGSTLER, C.; LUBICH, C. [1997a]: Multirate extrapolation methods for differential equations with different time scales, Computing 58, 173–185.

[62] ENGSTLER, C.; LUBICH, C. [1997b]: MUR8: A multirate extension of the eighth-order Dormand-Prince method, Appl. Num. Math. 25, 185–192.

[63] ESTÉVEZ SCHWARZ, D. [1999a]: Topological analysis for consistent initialization in circuit simulation, Preprint 99–3, Institut für Mathematik, Humboldt-Universität zu Berlin, Berlin.

[64] ESTÉVEZ SCHWARZ, D. [1999b]: Consistent initialization for index-2 differential-algebraic equations and its application to circuit simulation, Preprint 99–5, Institut für Mathematik, Humboldt-Universität zu Berlin, Berlin.

[65] ESTÉVEZ SCHWARZ, D.; LAMOUR, R. [1999]: The computation of consistent initial values for nonlinear index-2 differential-algebraic equations, Preprint 99–13, Humboldt-Universität zu Berlin, Berlin.

[66] ESTÉVEZ SCHWARZ, D. (2000): Consistent initialization for differential-algebraic equations and its application to circuit simulation, PhD thesis, Humboldt Universität zu Berlin, Berlin (available at: http://dochost.rz.hu-berlin.de/dissertationen).

[67] ESTÉVEZ SCHWARZ, D.; TISCHENDORF, C. [2000]: Structural analysis for electrical circuits and consequences for MNA, Int. J. Circ. Theory Appl. 28, 131–162.

[68] ESTÉVEZ SCHWARZ, D.; FELDMANN, U.; MÄRZ, R.; STURTZEL, S.; TISCHENDORF, C. [2003]: Finding benificial DAE structures in circuit simulation, to appear in Mathematics – Key Technology for the Future (Editors: W. JÄGER; H.-J. KREBS), Springer, Berlin.

[69] FELDMANN, P.; MELVILLE, B.; LONG, D. [1996]: Efficient frequency domain analysis of large nonlinear analog circuits, Proc. CICC'96, San Diego, 461–464.

[70] FELDMANN, U.; WEVER, U.; ZHENG, Q.; SCHULTZ, R.; WRIEDT, H. [1992]: Algorithms for modern circuit simulation, AEÜ 46, 274–285.

[71] FELDMANN, U.; GÜNTHER, M. [1999]: Some remarks on regularization of circuit equations, Proc. ISTET'99, Magdeburg, 343–348.

[72] FENG, D,; PHILLIPS, J.; NABORS, K.; KUNDERT, K.; WHITE, J. [1999]: Efficient computation of quasi-periodic circuit operating conditions via a mixed frequency/time approach, Proc. DAC'99, New Orleans, 635–640.

[73] FILSETH, E.S.; KUNDERT, K.S. [1996]: Simulations strategies for RF design, Electr. Eng. 68-832, 43–50.

[74] FISCHER, M. (2001): Multigranulare parallele Algorithmen zur Lösung von Gleichungssystemen der VLSI-Netzwerksimulation, PhD thesis, Christian-Albrechts-Universität zu Kiel, Kiel (ISBN 3-8265-8565-8, Shaker, Aachen).

[75] FRANK, J.E.; VAN DER HOUWEN, P.J. [2000]: Diagonalizable extended backward differential formulas, BIT, Vol 40-3, 497–512.

[76] FRIESE, T. [1996]: Eine adaptive Spektralmethode zur Berechnung periodischer Orbits, in Proc. 2nd ITG Workshop 1995, Berlin (Editors: W. MATHIS; P. NOLL), pp. 21–26, ISBN 3-8007-2190-2, VDE-Verlag, Berlin.

[77] FRÖHLICH, N.; RIESS, B.M.; WEVER, U.A.; ZHENG, Q. [1998]: A new approach for parallel simulation of VLSI circuits on a transistor level, IEEE Trans. Circ. Syst. CAS I 45, 601–613.

[78] FRÖHLICH, N.; GLÖCKEL, V.; FLEISCHMANN, J. [2000]: A new partitioning method for parallel simulation of VLSI circuits on transistor level, Proc. DATE'2000, Paris, 679–684.

[79] FRÖHLICH, N. (2002): Verfahren zum Schaltungspartitionieren für die parallele Simulation auf Transistorebene, PhD thesis, TU München, München.

[80] GAREY, M.R.; JOHNSON, D.S. [1979]: Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, New York.

[81] GEAR, C.W. [1971]: Simultaneous numerical solution of differential-algebraic equations, IEEE Trans. Circ. Theory CT 18, 89–95.

[82] GEAR, C.W.; WELLS, D.R. [1984]: Multirate linear multistep methods, BIT 24, 484–502.

[83] GEAR, C.W. [1988]: Differential-algebraic equation index transformations, SIAM J. Sci. Stat. Comp. 9, 39–47.

[84] GEAR, C.W. [1990]: Differential-algebraic equations, indices, and integral algebraic equations, SIAM J. Num. Anal. 27, 1527–1534.

[85] GEORGE, J.A. [1974]: On block elimination for sparse linear systems, SIAM J. Num. Anal. 585–603.

[86] GILMORE, R.J.; STEER, M.B. [1991]: Nonlinear circuit analysis using the method of Harmonic Balance — A review of the art: I. Introductory concepts, II. Advanced concepts, Int. J. Microwave Millimeter-Wave Comp.-Aided Eng. 1–1, 22–37 (Part I), 2–1, 159–180 (Part II).

[87] GOURARY, M.M.; ULYANOV, S.L.; ZHAROV, M.M.; RUSAKOV, S.G.; GULLAPALLI, K.K.; MULVANEY, B.J. [1998]: Simulation of high-Q oscillators, Proc. ICCAD'98, San Jose, 162–169.

[88] GRÄB, R.; GÜNTHER, M.; WEVER, U.; ZHENG, Q. [1996]: Optimization of parallel multilevel Newton algorithms on workstation clusters, in Proc. Euro-Par96, Lecture Notes Comp. Sci. 1124 (Editors: L. BOUGE et al.), pp. 91–96, Springer, Berlin.

[89] GRIEPENTROG, E.; MÄRZ, R. [1986]: Differential-Algebraic Equations and their Numerical Treatment, Teubner, Leipzig.

[90] GRISTEDE, G.D.; ZUKOWSKI, C.A.; RUEHLI, A.E. [1999]: Measuring error propagation in waveform relaxation algorithms, IEEE Trans. Circ. Syst. CAS I 46, 337–348.

[91] GÜNTHER, M.; RENTROP, P. [1993]: Multirate ROW methods and latency of electric circuits, Appl. Num. Math. 13, 83–102.

[92] GÜNTHER, M.; RENTROP, P. [1994]: Partitioning and multirate strategies in latent electric circuits, in Proc. Oberwolfach Conf., Int. Ser. Num. Math. 117 (Editors: R.E. BANK et al.), pp. 33–60, Birkhäuser, Basel.

[93] GÜNTHER, M. (1995): Ladungsorientierte Rosenbrock-Wanner-Methoden zur numerischen Simulation digitaler Schaltungen, PhD thesis, TU München, München (ISBN 3-18-316820-0, VDI-Verlag, Düsseldorf).

[94] GÜNTHER, M.; RENTROP, P. [1996]: The NAND-gate — A benchmark for the numerical simulation of digital circuits, in Proc. 2nd ITG Workshop 1995, Berlin (Editors: W. MATHIS; P. NOLL), pp. 27–33, ISBN 3-8007-2190-2, VDE-Verlag, Berlin.

[95] GÜNTHER, M.; HOSCHEK, M. [1997]: ROW methods adapted to electric circuit simulation packages, Comp. Appl. Math. 82, 159–170.

[96] GÜNTHER, M. [1998]: Simulating digital circuits numerically — A charge-oriented ROW approach, Num. Math. 79, 203–212.

[97] GÜNTHER, M.; FELDMANN, U. [1999a]: CAD based electric circuit modeling in industry I: Mathematical structure and index of network equations, Surv. Math. Ind. 8, 97–129.

[98] GÜNTHER, M.; FELDMANN, U. [1999b]: CAD based electric circuit modeling in industry II: Impact of circuit configurations and parameters, Surv. Math. Ind. 8, 131–157.

[99] GÜNTHER, M.; HOSCHEK, M.; WEINER, R. [1999]: ROW methods adapted to a cheap Jacobian, Appl. Num. Math. 37, 231–240.

[100] GÜNTHER, M.; RENTROP, P. [1999]: PDAE-Netzwerkmodelle in der elektrischen Schaltungssimulation, Proc. Analog'99, München, 31–38.

[101] GÜNTHER, M.; HOSCHEK, M.; RENTROP, P. [2000]: Differential-algebraic equations in electric circuit simulation, Int. J. Electron. Commun. (AEÜ) 54, 101–107.

[102] GÜNTHER, M. [2001]: Partielle differential-algebraische Systeme in der numerischen Zeitbereichsanalyse elektrischer Schaltungen, ISBN 3-18-334320-7, VDI-Verlag, Düsseldorf.

[103] GÜNTHER, M.; KVÆRNØ, A.; RENTROP, P. [2001]: Multirate partitioned Runge-Kutta methods, BIT 41, 504–515.

[104] GÜNTHER, M.; RENTROP, P.; FELDMANN, U. [2001]: CHORAL — A one step method as numerical low pass filter in electrical network analysis, in Proc. SCEE-2000, Warnemünde, Lecture Notes Comp. Sci. Eng. 18 (Editors: U. VAN RIENEN et al.), pp. 199–215, Springer, Berlin.

[105] GUPTA, G.K.; GEAR, C.W.; LEIMKUHLER, B.J. [1985]: Implementing linear multistep formulas for solving DAEs, Rep. No. UIUCDCS-R-85-1205, University of Illinois, Urbana.

[106] GUSTAFSSON, K.; LUNDH, M.; SÖDERLIND, G. [1988]: A PI stepsize control for the numerical solution of ordinary differential equations, BIT 28, 270–287.

[107] HACHTEL, G.D.; BRAYTON, R.K; GUSTAVSON, F.G. [1971]: The sparse tableau approach to network analysis and design, IEEE Trans. Circ. Theory CT 18, 101–113.

[108] HACHTEL, G.D.; SANGIOVANNI-VINCENTELLI, A. [1981]: A survey of third-generation simulation techniques, Proc. IEEE 69, 1264–1280.

[109] HAIRER, E.; NØRSETT, S.P.; WANNER, G. [1987]: Solving Ordinary Differential Equations I, Springer, Berlin.

[110] HAIRER, E.; LUBICH, C.; ROCHE, M. [1989]: The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods, Lecture Notes in Mathematics 1409, Springer, Berlin.

[111] HAIRER, E.; WANNER, G. [1996]: Solving Ordinary Differential Equations II, 2nd Ed., Springer, Berlin.

[112] HAJIMIRI, A.; LEE, T.H. [1998]: A general theory of phase noise in electrical oscillators, IEEE J. Solid-State Circ., Vol. 33-2, 179–194.

[113] HEATH, M.; NG, E.; PEYTON, B. [1990]: Parallel algorithms for sparse linear systems, in Parallel Algorithms for Matrix Computations (Editors: K.A. GALLIVAN et al.), pp. 83–124, SIAM, Philadelphia.

[114] HO, C.W.; RUEHLI, A.E.; BRENNAN, P.A. [1975]: The modified nodal approach to network analysis, IEEE Trans. Circ. Syst. CAS 22, 505–509.

[115] HOFFMANN, K. [1996]: VLSI-Entwurf, Oldenbourg, München.

[116] HONKALA, M.; ROOS, J.; VALTONEN, M. [2001]: New multilevel Newton-Raphson method for parallel circuit simulation, Proc. ECCTD'01, Helsinki, Vol. II, 113–116.

[117] HONKALA, M.; KARANKO, V.; ROOS, J. [2002]: Improving the convergence of combined Newton-Raphson and Gauss-Newton multilevel iteration method, Proc. of ISCAS'02, Scottsdale, Arizona, May 26-29.

[118] HORNEBER, E.-H. [1985]: Simulation elektrischer Schaltungen auf dem Rechner, Springer, Berlin.

[119] HOSCHEK, M. (1999): Einschrittverfahren zur numerischen Simulation elektrischer Schaltungen, PhD thesis, TU Darmstadt, Darmstadt (ISBN 3-18-329320-X, VDI-Verlag, Düsseldorf).

[120] HOSEA, M.E.; SHAMPINE, L.F. [1996]: Analysis and implementation of TR-BDF2, Appl. Num. Math. 20, 21–37.

[121] HOUBEN, S.H.M.J. [1999]: Algorithms for periodic steady state analysis on electric circuits, Unclassified NatLab Report 804/99, Philips Research Laboratories, Eindhoven.

[122] HOUBEN, S.H.M.J.; MAUBACH, J.M. [2000]: An accelerated Poincaré-map method for autonomous oscillators, Preprint TU Eindhoven, Eindhoven
(available at: http://www.win.tue.nl/~anwww/preprints/2000.html).

[123] HOUBEN, S.H.M.J.; MAUBACH, J.M. [2001]: Periodic steady-state analysis of free-running oscillators, in Proc. SCEE-2000, Warnemünde, Lecture Notes Comp. Sci. Eng. 18 (Editors: U. VAN RIENEN et al.), pp. 217–224, Springer, Berlin.

[124] HOUBEN, S.H.M.J.; TER MATEN, E.J.W.; MAUBACH, J.M.; PETERS, J.M.F. [2001]: Novel time-domain methods for free-running oscillators, Proc. ECCTD'01, Helsinki, Vol. III, 393–396.

[125] HOWARD, A. [1995]: Circuit envelope simulator analyzes high-frequency modulated signals, RF Design 18–9, 36–45.

[126] HOYER, W.; SCHMIDT, J.W. [1984]: Newton-type decomposition methods for equations arising in network analysis, ZAMM 64, 397–405.

[127] HSIM Datasheet [2002]: The full-chip hierarchical circuit simulation and analysis tool, NASSDA, http://www.nassda.com.

[128] IAVERNARO, F.; MAZZIA, F. [2002]: Generalization of backward differentiation formulas for parallel computers, preprint Univ. Bari.

[129] JOKUNEN, H. (1997): Computation of the steady-state solution of nonlinear circuits with time-domain and large-signal-small-signal analysis methods, PhD thesis Helsinki University of Technology, Espoo (Acta Polytechnica Scandinavica, Electrical Engineering Series 87, 1–75).

[130] KÄRTNER, F.X. [1989]: Untersuchung des Rauschverhaltens von Oszillatoren, PhD-Thesis TU München.

[131] KÄRTNER, F.X. [1990]: Analysis of white and $f^{-\alpha}$ noise in electrical oscillators, Int. J. Circuit Theory Appl., Vol. 18, 485–519.

[132] KAGE, T.; KAWAFUJI, F.; NIITSUMA, J. [1994]: A circuit partitioning approach for parallel simulation, IEICE Trans. Fundamentals E77–A 3, 461–466.

[133] KAMPOWSKY, W.; RENTROP, P.; SCHMIDT, W. [1992]: Classification and numerical simulation of electric circuits, Surv. Math. Ind. 2, 23–65.

[134] KANAGLEKAR, N.; SIFRI, J. [1997]: Integrate CAD methods for accurate RF IC simulation, Microwaves & RF, 88–101.

[135] KATO, T.; TACHIBANA, W. [1998]: Periodic steady-state analysis of an autonomous power electronic system by a modified shooting method, IEE Trans. Power Electr. 13–3, 522–527.

[136] KATZENELSON, J. [1965]: An algorithm for solving nonlinear resistor networks, Bell Syst. Tech. J. 44, 1605–1620.

[137] KEVENAAR, T.A.M. [1994]: Periodic steady state analysis using shooting and waveform-Newton, Int. J. Circ. Theory Appl. 22, 51–60.

[138] KLAASSEN, B.; PAAP, K.L. [1987]: Correction formula and LTE-estimation for trapezoidal rule, in Proc. VLSI and Computers, CompEuro87 (Editors: W.E. PROEBSTER; H. REINER), pp. 238–241, ISBN 0-8186-0773-4, IEEE Computer Society, Washington D.C..

[139] KUNDERT, K.S.; WHITE, J.K.; SANGIOVANNI-VINCENTELLI, A. [1990]: Steady-state methods for simulating analog and microwave circuits, Kluwer, Boston.

[140] KUNDERT, K.S. [1994]: Accurate Fourier analysis for circuit simulators, Proc. CICC'94, San Diego, 25–28.

[141] KUNDERT, K.S. [1995]: The designer's guide to Spice & Spectre, Kluwer Academic Publishers, Boston.

[142] KUNDERT, K.S. [1997]: Simulation methods for RF integrated circuits, Proc. ICCAD'97, San Jose.

[143] KUROKAWA, K. [1969]: Some basic characteristics of broadband negative resistance oscillator circuits, Bell System. Techn. J. 48, 1937–1955.

[144] KUROKAWA, K. [1973]: Injection locking of microwave solid-state oscillators, Proc. of the IEEE 61–10, 1386–1410.

[145] KVÆRNØ, A.; RENTROP, P. [1999]: Low order multirate Runge-Kutta methods in electric circuit simulation, Preprint 99/1, IWRMM, University of Karlsruhe, Karlsruhe.

[146] LAMOUR, R. [1997]: A shooting method for fully implicit index-2 differential-algebraic equations, SIAM J. Sci. Stat. Comp. 18, 94–114.

[147] LAMOUR, R. [1998]: Floquet theory for differential-algebraic equations (DAE), ZAMM 78–3, S989–S990.

[148] LAMOUR, R.; MÄRZ, R.; WINKLER, R. [1998a]: How Floquet theory applies to index 1 differential-algebraic equations, J. Math. Anal. Appl. 217, 372–394.

[149] LAMOUR, R.; MÄRZ, R.; WINKLER, R. [1998b]: Stability of periodic solutions of index-2 differential-algebraic systems, Preprint 98–23, Humboldt Universität zu Berlin, Berlin.

[150] LAMPE, S.; BRACHTENDORF, H.G.; TER MATEN, E.J.W.; ONNEWEER, S.P.; LAUR, R. [2001]: Robust limit cycle calculations of oscillators, in Proc. SCEE-2000, Warnemünde, Lecture Notes Comp. Sci. Eng. 18 (Editors: U. VAN RIENEN et al.), pp. 233–240, Springer, Berlin.

[151] LAMPE, S.; LAUR, R. [2002]: Initialisierungsprozedur für die Oszillatorsimulation basierend auf globalen Optimierungstechniken, ANALOG 2002, Bremen, 81–86.

[152] LEIMKUHLER, B.J. [1986]: Error estimates for differential-algebraic equations, Report UIUCDCS-R-86-1287, University of Illinois, Urbana.

[153] LELARASMEE, E.; RUEHLI, A.E.; SANGIOVANNI-VINCENTELLI, A. [1982]: The waveform relaxation method for time-domain analysis of large scale integrated circuits, IEEE Trans. Comp. Aided Des. CAD 1, 131–145.

[154] LI, D,; TYMERSKI, R. [2000]: Comparison of simulation algorithms for accelerated determination of periodic steady state of switched networks, IEEE Trans. Ind. Electr. 47–6, 1278–1285.

[155] LIN, S.; KUH, E.S.; MAREK-SADOWSKA, M. [1993]: Stepwise equivalent conductance circuit simulation technique, IEEE Trans. Comp. Aided Des. CAD 12, 672–683.

[156] LIOEN, W.M.; DE SWART, J.J.B. [2001]: Test set for initial value problem solvers, Report CWI Amsterdam, Amsterdam (available at: http://www.dm.uniba.it/testset).

[157] MA, W.; TRAJKOVĆ, L; MAYARAM, K. [2002]: HomSSPICE: A homotopy-based circuit simulator for periodic-steady state analysis of oscillators, Paper presented at ISCAS-2002.

[158] MÄRZ, R. [1992]: Numerical methods for differential-algebraic equations, Acta Numerica, 141–198.

[159] MÄRZ, R.; TISCHENDORF, C. [1997]: Recent results in solving index 2 differential-algebraic equations in circuit simulation, SIAM J. Sci. Stat. Comp. 18–1, 139–159.

[160] MARKOWITZ, H. [1957]: The elimination form of the inverse and its application to linear programming, Management Sci. 3, 255–269.

[161] MATHIS, W. [1987]: Theorie nichtlinearer Netzwerke, Springer, Berlin.

[162] MATHIS, W.; MAURITZ, H.; ZHUANG, M. [1994]: Entwurf von ODE- und DAE-Lösungsverfahren mit variabler Schrittweite nach regelungs-technischen Prinzipien: Möglichkeiten und Grenzen, 2. Workshop "Identifizierungs-, Analyse- und Entwurfsmethoden für Deskriptorsysteme", Paderborn.

[163] MATHIS, W. [1998]: An efficient method for the transient analysis of weakly damped crystal oscillators, Proc. MTNS'98, Padova, 313–316.

[164] MAYARAM, K.; LEE, D.C.; MOINIAN, S.; RICH, D.; ROYCHOWDHURY, J. [1997]: Overview of computer-aided analysis tools for RFIC: Algorithms, features, and limitations, Proc. CICC'97, Santa Clara, 505–512.

[165] MEETZ, K.; ENGL, W.L. [1980]: Elektromagnetische Felder, Springer, Berlin.

[166] MELVILLE, R.C.; FELDMANN, P.; ROYCHOWDHURY, J. [1995]: Efficient multi-tone distortion analysis of analog integrated circuits, Proc. CICC'97, Santa Clara, 241–244.

[167] MEYER, J.E. [1971]: MOS models and circuit simulation, RCA Rev. 32, 42–63.

[168] MILSOM, R.F.; JAMIESON, P.A.; SCOTT, K.J. [1994]: Accurate system level models of RF circuits, IEE Coll. Comp. Modelling of Comm. Systems 115–2, 1–6.

[169] MIURA-MATTAUSCH, M.; FELDMANN, U.; RAHM, A.; BOLLU, M.; SAVIGNAC, D. [1996]: Unified complete MOSFET model for analysis of digital and analog circuits, IEEE Trans. Comp. Aided Des. CAD 15, 1–7.

[170] NAGEL, W. [1975]: SPICE 2 — A computer program to simulate semiconductor circuits, MEMO ERL-M 520, University of California Berkeley, Berkeley.

[171] NEUBERT, R.; SELTING, P.; ZHENG, Q. [1998]: Analysis of autonomous oscillators — A multistage approach, Proc. MTNS'98, Padova, 1055–1058.

[172] NEUBERT, A.; SCHWARZ, A. [2001]: Efficient analysis of oscillatory circuits, in Proc. SCEE-2000, Warnemünde, Lecture Notes Comp. Sci. Eng. 18 (Editors: U. VAN RIENEN et al.), pp. 225–232, Springer, Berlin.

[173] NEWTON, A.R.; SANGIOVANNI-VINCENTELLI, A. [1984]: Relaxation-based electrical simulation, IEEE Trans. Comp. Aided Des. CAD 3, 308–330.

[174] NGOYA, E.; ROUSSET, J.; GAYRAL, W.; QUÉRÉ, R.; OBREGON, J. [1990]: Efficient algorithms for spectra calculations in nonlinear microwave circuits simulators, IEEE Trans. Circ. Syst. CAS 37, 1339–1355.

[175] NGOYA, E.; SUÁREZ, A.; SOMMET, R.; QUÉRÉ, R. [1995]: Steady state analysis of free and forced oscillators by Harmonic Balance and stability investigation of periodic and quasi-periodic regimes, Int. J. Microwave Millimeter-Wave Comp.-Aided Eng. 5–3, 210–223.

[176] ODENT, P.; CLAESEN, L.; DE MAN, H. [1990]: Acceleration of relaxation-based circuit simulation using a multiprocesor system, IEEE Trans. Comp. Aided Des. CAD 9, 1063–1072.

[177] ODRYNA, P.; NASSIF, S. [1986]: The ADEPT timing simulation program, VLSI Systems Design, March 1986, 24–34.
oder???? ODRYNA,P.; NAZARETH, K.; CHRISTENSEN, C. [1986]: A workstation-mixed model circuit simulator, Proc. DAC'86, Las Vegas, 186-192.

[178] OKUMURA, M.; SUGAWARA, T.; TANIMOTO, H. [1990]: An efficient small signal frequency analysis method for nonlinear circuits with two frequency excitations, IEEE Trans. Comp. Aided Des. CAD 9, 225–235.

[179] OKUMURA, M.; TANIMOTO, H.; ITAKURA, T.; SUGAWARA, T. [1993]: Numerical noise analysis for nonlinear circuits with a periodic large signal excitation including cyclostationary noise sources, IEEE Trans. Circ. Syst. CAS I 40, 581–590.

[180] ONOZUKA, H.; KANH, M.; MIZUTA, C.; NAKATA, T.; TANABE, N. [1993]: Development of parallelism for circuit simulation by tearing, Proc. Europ. Des. Autom. Conf. EDAC'93, 12–17.

[181] PALUSINSKI, O.A.; GUARINI, M.W.; WRIGHT, S.J. [1988]: Spectral technique in electronic circuit analysis, Int. J. Num. Modelling: Electr. Netw., Dev. and Fields 1, 137–151.

[182] PANTELIDES, C.C. [1988]: The consistent initialization of differential-algebraic systems, SIAM J. Sci. Stat. Comp. 9, 213–231.

[183] PERŠIČ, B.; MEDIČ, I. [1996]: Extracting frequency of an autonomous system by Harmonic Balance, Proc. IEEE Asia Pacific Conf. Circ. Syst.'96, Seoul, Vol. 7–5, 437–440.

[184] PETZOLD, L.R. [1981]: An efficient numerical method for highly oscillatory ordinary differential equations, SIAM J. Num. Anal. 18–3, 455–479.

[185] PETZOLD, L.R.; JAY, L.; YEN, J. [1997]: Numerical solution of highly oscillatory ordinary differential equations, Acta Numerica 6, 437–484.

[186] PULCH, R.; GÜNTHER, M. [2002]: A method of characteristics for solving multirate partial differential equations in radio frequency application, Appl. Num. Math. 42, 397–409.

[187] RABBAT, G.; HSIEH, H.-Y. [1981]: Hierarchical computer-aided circuit design techniques, In: Proc. of the 1981 European Conference on Circuit Theory and Design, (The Hague), Delft University Press, Delft, 136–144.

[188] RABBAT, N.B.G.; SANGIOVANNI-VINCENTELLI, A.L.; HSIEH, H.Y. [1979]: A multilevel Newton algorithm with macromodeling and latency for the analysis of large-scale nonlinear circuits in the time domain, IEEE Trans. Circ. Syst. CAS 26, 733–741.

[189] RABIER, P.; RHEINBOLDT, W. [1996]: Time-dependent linear DAE's with discontinuous inputs, J. Lin. Algebra Appl. 247, 1–29.

[190] RABIER, P.J.; RHEINBOLDT, W.C. [2002]: Theoretical and numerical analysis of differential-algebraic equations, In: P.G. Ciarlet, J.L. Lions (Eds): Handbook of numerical analysis, Vol VIII: Techniques of scientific computing (Part 4), Elsevier Science BV, 183–540.

[191] REISSIG, G.; FELDMANN, U. [1996]: Computing the generic index of the circuit equations of linear active networks, Proc. ISCAS'96, Atlanta, Vol. III, 190-193.

[192] REISSIG, G. (1998): Beiträge zu Theorie und Anwendung impliziter Differentialgleichungen, PhD thesis, TU Dresden, Dresden.

[193] REISSIG, G. [2001]: On the performance of minimum degree and minimum local fill heuristics in circuit simulation, Report Febr. 2001, Dept. Chem. Eng., MIT, Cambridge.

[194] RENTROP, P.; ROCHE, M.; STEINEBACH, G. [1989]: The application of Rosenbrock-Wanner type methods with stepsize control in differential-algebraic equations, Num. Math. 55, 545–563.

[195] RENTROP, P. [1990]: ROW-type methods for the integration of electric circuits, in Proc. Oberwolfach Conf., Int. Ser. Num. Math. 93 (Editors: R.E. BANK et al.), pp. 59–71, Birkhäuser, Basel.

[196] RENTROP, P.; STREHMEL, K.; WEINER, R. [1996]: Ein Überblick über Einschrittverfahren zur numerischen Integration in der technischen Simulation, GAMM Mitteilungen 19–1, 9–43.

[197] RICE, J.R. [1960]: Split Runge-Kutta methods for simultaneous equations, J. Res. Natl. Buv. Stand. 64B, 151–170.

[198] RIZZOLI, V.; Costanzo, A.; Ghigi, P.R.; Mastri, F.; Masotti, D.; Cecchetti, C. [1992]: Recent advances in Harmonic-Balance techniques for nonlinear microwave circuit simulation, AEÜ 46, 286–297.

[199] RÖSCH, M. (1992): Schnelle Simulation des stationären Verhaltens nichtlinearer Schaltungen, PhD thesis, TU München, München.

[200] RÖSCH, M.; ANTREICH, K. [1992]: Schnelle stationäre Simulation nichtlinearer Schaltungen im Frequenzbereich, AEÜ 46, 168–176.

[201] ROYCHOWDHURY, J. [1997]: Efficient methods for simulating highly nonlinear multi-rate circuits, Proc. DAC'97, Anaheim, 269–274.

[202] ROYCHOWDHURY, J.; FELDMANN, P. [1997]: A new linear-time Harmonic Balance algorithm for cyclostationary noise analysis in RF, Proc. ASP-DAC'97, Chiba, 483–492.

[203] ROYCHOWDHURY, J.; LONG, D.; FELDMANN, P. [1998]: Cyclostationary noise analysis of large RF circuits with multitone excitations, IEEE J. Solid-State Circ. 33–3, 324–336.

[204] ROYCHOWDHURY, J. [2001]: Analyzing circuits with widely-separated time scales using numerical PDE methods, IEEE Trans. Circ. Syst. CAS I 48, 578-594.

[205] ROYCHOWDHURY, J. [2001]: Multi-time PDEs for dynamical system analysis, in Proc. SCEE-2000, Warnemünde, Lecture Notes Comp. Sci. Eng. 18 (Editors: U. VAN RIENEN et al.), pp. 3–14, Springer, Berlin.

[206] SAAD, Y. [1996]: Iterative Methods for Sparse Linear Systems, PWS Publ., Boston.

[207] SACKS-DAVIS, R. [1972]: Error estimates for a stiff differential equation procedure, SIAM J. Stat. Comp. 3, 367–384.

[208] SADAYAPPAN, P.; VISVANATHAN, V. [1988]: Circuit simulation on shared-memory multiprocessors, IEEE Trans. Computers 37–12, 1634–1642.

[209] SAKALLAH, K.A.; DIRECTOR, S.W. [1985]: SAMSON2: An event driven VLSI circuit simulator, IEEE Trans. Comp. Aided Des. CAD 4, 668–685.

[210] SAKALLAH, K.A.; YEN, Y.; GREENBERG, S.S. [1990]: A first-order charge conserving MOS capacitance model, IEEE Trans. Comp. Aided Des. CAD 9, 99–108.

[211] SALEH, R.A.; KLECKNER, J.E.; NEWTON, A.R. [1983]: Iterated timing analysis in SPLICE1, Proc. ICCAD'83, Santa Clara, 139–140.

[212] SALEH, R.; WEBBER, D.; XIA, E.; SANGIOVANNI-VINCENTELLI, A. [1987]: Parallel waveform Newton algorithms for circuit simulation, Proc. Int. Conf. Comp. Des.'87, Port Chester, 660–668.

[213] SALEH, R.A.; WHITE, J.K. [1990]: Accelerating relaxation algorithms for circuit simulation using waveform-Newton and step-size refinement, IEEE Trans. Comp. Aided Des. CAD 9, 951–958.

[214] SANGIOVANNI-VINCENTELLI, A.; CHEN, L.K.; CHUA, L.O. [1977]: A new tearing approach — Node-tearing nodal analysis, Proc. ISCAS'77, Phoenix, 143–147.

[215] SARKANI, E.; LINIGER, W. [1974]: Exponential fitting of matricial multistep methods for ordinary differential equations, Math. Comput. 28, 1035–1052.

[216] SCHENK, O. (2000): Scalable parallel sparse LU factorization methods on shared memory multiprocessors, PhD thesis, ETH Zürich, Zürich (ISBN-3-89649-532-1, Series in microelectronics 89, Hartung-Gorre, Konstanz).

[217] SCHILDERS, W. [2000]: Iterative Solution of linear systems in circuit simulation, In: Progress in Industrial Mathematics at ECMI 2000, Mathematics in Industry, Vol. 1, Springer-Verlag, 272 – 277, 2002.

[218] SCHMIDT-KREUSEL, C. (1997): Zur rechnergestützten Analyse von Quarzoszillatoren, PhD thesis, Bergische Universität Wuppertal, Wuppertal.

[219] SEVAT, M.F. [1994]: Fourier transformation for harmonic balance, Philips Research Laboratories, Nat.lab. Report Nr. 6813/94.

[220] SELTING, P.; ZHENG, Q. [1997]: Numerical stability analysis of oscillating integrated circuits, J. Comp. Appl. Math. 82, 367–378.

[221] SIEBER, E.-R.; FELDMANN, U.; SCHULTZ, R.; WRIEDT, H. [1994]: Timestep control for charge conserving integration in circuit simulation, in Proc. Oberwolfach Conf., Int. Ser. Num. Math. 117 (Editors: R.E. BANK et al.), pp. 103–113, Birkhäuser, Basel.

[222] SILVEIRA, L.M.; WHITE, J.K.; NETO, H.; VIDIGAL, L. [1992]: On exponential fitting for circuit simulation, IEEE Trans. Comp. Aided Des. CAD 11, 566–574.

[223] SIMEON, B. [1998]: Order reduction of stiff solvers at elastic multibody systems, Appl. Num. Math. 28, 459–475.

[224] SKELBOE, S. [1982]: Time-domain steady-state analysis of nonlinear electrical systems, Proc. of the IEEE 70, 1210–1228.

[225] SKELBOE, S. [1984]: Multirate integration methods, Report ECR-150, University of Horsholm, Horsholm.

[226] SMITH, D.A.; FORD, W.F.; SIDI, A. [1987]: Extrapolation methods for vector sequences, SIAM Review 29–2, 199–233.

[227] SÖDERLIND, G. [2001]: Automatic control and adaptive time-stepping, Proc. ANODE 2001, Auckland.

[228] STRIEBEL, M.; GÜNTHER, M. [2002]: Towards Distributed Time Integration in Full Chip Design. To appear in Appl. Numer. Math.

[229] STROHBAND, P.H.; LAUR, R.; ENGL, W.L. [1977]: TNPT — An efficient method to simulate forced nonlinear RF networks in time domain, IEEE J. Solid-State Circ. SC 12–3, 243–246.

[230] DE SWART, J.J.B. [1997]: Parallel software for implicit differential equations, PhD-Thesis Univ. of Amsterdam.

[231] TELICHEVESKY, R.; KUNDERT, K.S.; WHITE, J.K. [1995]: Efficient steady-state analysis based on matrix-free Krylov-subspace methods, Proc. DAC'95, San Francisco, 480–484.

[232] TELICHEVESKY, R.; KUNDERT, K.S.; WHITE, J. [1996]: Efficient AC and noise analysis of two-tone RF circuits, Proc. DAC'96, Las Vegas, 292–297.

[233] TELICHEVESKY, R.; KUNDERT, K.S.; ELFADEL, I.; WHITE, J.K. [1996]: Fast simulation algorithms for RF circuits, Proc. CICC'96, San Diego, 437–444.

[234] TER MATEN, E.J.W. [1999]: Numerical methods for frequency domain analysis of electronic circuits, Surv. Math. Ind. 8, 171–185.

[235] TER MATEN, E.J.W.; VAN DE WIEL, M.C.J. [1999]: Time-domain simulation of noise in dynamic non-linear circuits, in Proc. ECMI'98, Göteborg (Editors: L. ARKERYD et al.), pp. 413–422, Teubner, Stuttgart.

[236] TISCHENDORF, C. [1996]: Solution of index-2 differential algebraic equations and its application in circuit simulation, Logos Verlag Berlin, PhD-Thesis Humboldt Univ. zu Berlin.

[237] TISCHENDORF, C. [1999]: Topological index calculation of differential-algebraic equations in circuit simulation, Surv. Math. Ind. 8, 187–199.

[238] TROYANOVSKY, B.; ROTELLA, F.M.; YU, Z.; DUTTON, R.W.; SATA-IWANAGA, J. [1996]: Large signal analysis of RF/microwave devices with parasitics using Harmonic Balance device simulation, Proc. SASIMI'96, Fukuoka, ????–????.

[239] TROYANOVSKY, B. (1997): Frequency domain algorithms for simulating large signal distortion in semiconductor devices, PhD thesis, Stanford University, Stanford.

[240] URUHAMA, K. [1987]: Convergence of relaxation-based circuit simulation techniques, IEICE Trans. E 70, 887–889.

[241] URUHAMA, K. [1988]: Relaxation based circuit simulation, IEICE Trans. E 71, 1189–1194.

[242] VALSA, J.; VLACH, J. [1995]: SWANN — A program for analysis of switched analog nonlinear networks, Proc. ISCAS'95, Seattle, 1752–1755.

[243] VAN BOKHOVEN, W.M.G. [1987]: Piecewise linear solution techniques, in Circuit Analysis, Simulation and Design, Part II (Editor: A.E. RUEHLI), pp. 41–127, North-Holland, Amsterdam.

[244] VAN DER HOUWEN, P.J.; DE SWART, J.J.B. [1997]: Parallel linear system solvers for Runge-Kutta methods, Advances in Computational Mathematics, Vol 7, 157–181.

[245] VAN DER HOUWEN, P.J.; DE SWART, J.J.B. [1997]: Triangularly implicit iteration methods for ODE-IVP solvers, SIAM J. Sci. Comput., Vol 18-1, 41–55.

[246] VAN DER HOUWEN, P.J.; SOMMEIJER, B.P. [1987]: Explicit Runge-Kutta-Nyström methods with reduced phase errors for computing oscillating solutions, SIAM J. Num. Anal. 24, 595–617.

[247] VAN EIJNDHOVEN, J.T.J. (1984): A piecewise linear simulator for large scale integrated circuits, PhD thesis, Eindhoven University of Technology, Eindhoven.

[248] VIDIGAL, L.; NASSIF, S.; DIRECTOR, S. [1986]: CINNAMON: Coupled integration and nodal analysis of MOS networks, Proc. DAC'86, Las Vegas, 179–185.

[249] VLACH, M. [1988a]: LU decomposition algorithms for parallel and vector computation, in Analog Methods for Computer-Aided Circuit Analysis and Design (Editor: T. OZAWA), pp. 37–64, Marcel Dekker, New York.

[250] VLACH, M. [1988b]: Decomposition techniques in circuit simulation, in Analog Methods for Computer-Aided Circuit Analysis and Design (Editor: T. OZAWA), pp. 93–115, Marcel Dekker, New York.

[251] WALLAT, O. (1997): Partitionierung und Simulation elektrischer Netzwerke mit einem parallelen mehrstufigen Newton-Verfahren, PhD thesis, Universität Hamburg, Hamburg.

[252] WANG, S.; DENG, A.-C. [2001]: Delivering a full-chip hierarchical circuit simulation & analysis solution for nanometer designs, White paper, Nassda Corporation.

[253] WARD, D.E.; DUTTON, R.W. [1978]: A charge-oriented model for MOS transistor capacitances, IEEE J. Solid-State Circ. SC 13, 703–708.

[254] WEHRHAHN, E. [1989]: Hierarchical circuit analysis, 1989 IEEE International Symposium on Circuits and Systems (Cat. No.89CH2692-2), Portland, OR, USA, Vol. 1, 701–704.

[255] WEHRHAHN, E. [1991]: Hierarchical sensitivity analysis of circuits, 1991 IEEE International Symposium on Circuits and Systems (Cat. No.91CH3006-4), Singapore, Vol. 2, 864–867.

[256] WELSCH, G. (1998): Analyse des eingeschwungenen Zustands autonomer und nicht-autonomer elektronischer Schaltungen, PhD thesis, Universität Bremen, Bremen (ISBN 3-8265-4233-9, Shaker, Aachen).

[257] WELSCH, G.; BRACHTENDORF, H.-G.; SABELHAUS, C.; LAUR, R. [2001]: Minimization of the error in the calculation of the steady state by shooting methods, IEEE Trans. on Circuits and Systems I, Fund. Theory and Applics., Vol. 48-10, 1252-1257.

[258] WEVER, U.; ZHENG, Q. [1996]: Parallel transient analysis for circuit simulation, Proc. 29th Annual Hawaii International Conference on System Sciences, Hawaii, 442–447.

[259] WHITE, J.K.; SANGIOVANNI-VINCENTELLI, A. [1987]: Relaxation Techniques for the Simulation of VLSI Circuits, Kluwer, Boston.

[260] WIEDL, W. [1994]: Multilevel Newton Verfahren in der Transientenanalyse elektrischer Netzwerke, in Proc. Oberwolfach Conf., Int. Ser. Num. Math. 117 (Editors: R.E. BANK et al.), pp. 103–113, Birkhäuser, Basel.

[261] WILKINSON, J.H. [1982]: Note on the practical significance of the Drazin inverse, in Recent Applications of Generalized Inverses (Editor: S.L. CAMPBELL), pp. 82–99, Pitman, London.

[262] WING, O.; GIELCHINSKY, J. [1972]: Computation of time response of large networks by partitioning, Proc. Int. Symp. Circ. Theory, ????wo (Calif.), 125–129.

[263] WU, F.F. [1976]: Solution of large-scale networks by tearing, IEEE Trans. Circ. Syst. CAS 23, 706–713.

[264] YU, Q.; WING, O. [1984]: PLMAP: A piecewise linear MOS circuit analysis program, Proc. ISCAS'84, Montreal, 530–533.

[265] ZECEVIC, A.I.; GACIC, N. [1999]: A partitioning algorithm for the parallel solution of differential-algebraic equations by waveform relaxation, IEEE Trans. Circ. Syst. CAS I 46, 421–434.

[266] ZHANG, X. (1989): Parallel computation for the solution of block bordered nonlinear equations and their applications, PhD thesis, University of Colorado, Boulder.

[267] ZHANG, X.; BYRD, R.H.; SCHNABEL, R.B. [1992]: Parallel methods for solving nonlinear block bordered systems of equations, SIAM J. Sci. Stat. Comp. 13, 841–859.

[268] ZHENG, Q. [1994]: The transient behavior of an oscillator, in Proc. Oberwolfach Conf., Int. Ser. Num. Math. 117 (Editors: R.E. BANK et al.), pp. 143–154, Birkhäuser, Basel.

[269] ZHENG, Q.; NEUBERT, R. [1997]: Computation of periodic solutions of differential-algebraic equations in the neighborhood of Hopf bifurcation points, Int. J. Bifurcation and Chaos 7, 2773–2779.