Electronic Circuit Simulation and the Development of New Krylov-Subspace Methods

Roland W. Freund

Abstract Ever since the 1960s, the semiconductor industry has heavily relied on simulation in order to analyze and verify the design of integrated circuits before actual chips are manufactured. Over the decades, the algorithms and tools of circuit simulation have evolved in order to keep up with the ever-increasing complexity of integrated circuits, and at certain points of this evolution, new simulation techniques were required. Such a point was reached in the early 1990s, when a new approach was needed to efficiently and accurately simulate the effects of the ever-increasing amount of on-chip wiring on the proper functioning of the chip. The industry's proposed solution for this task, the AWE approach, worked well for small- to moderate-size networks of on-chip wiring, but suffered from numerical issues for larger networks. It turned out that for the special case of networks with single inputs and single outputs, these problems can be remedied by exploiting the connection between AWE and the classical Lanczos algorithm for single starting vectors. However, the general case of on-chip wiring involves networks with multiple inputs and outputs, and so a Lanczos-type algorithm was needed that could handle such multiple starting vectors. Since no such extension existed, a new band Lanczos algorithm for multiple starting vectors was developed. It turned out that this new band approach can also be employed to devise extensions of other Krylov-subspace methods. In this chapter, we describe the band Lanczos algorithm and the band Arnoldi process and how their developments were driven by the need to efficiently and accurately simulate the effects of on-chip wiring of integrated circuits.

Roland W. Freund
Department of Mathematics, University of California at Davis,
One Shields Avenue, Davis, California 95616, USA
e-mail: freund@math.ucdavis.edu

1 Introduction

Since the invention of integrated circuits in the late 1950s, the semiconductor industry has succeeded in manufacturing chips with ever-decreasing feature size and ever-increasing complexity. As a result, the number of transistors on state-of-the-art chips evolved from tens of transistors on a single chip in the late 1950s to tens of billions of transistors on a single chip in 2019. Already early in this evolution, it became apparent that computer simulation is indispensable in order to analyze and verify circuit designs before actual chips are manufactured. The methods, tools, and software used for such simulations are referred to as *electronic circuit simulation* or simply *circuit simulation*; see, e.g., [33]. The basic framework of circuit simulation was created in the 1960s and early 1970s, culminating in Nagel's SPICE circuit simulator [28]; accounts of these developments can be found in [31, 35]. Most of the circuit simulators in use today are variants or derivatives of SPICE.

1.1 The Central Numerical Task in Circuit Simulation

Circuit simulation uses the *lumped-element approach* to model integrated circuits as networks of idealized electrical circuit elements, such as resistors, capacitors, inductors, diodes, and transistors. The branches of such a network model correspond to the circuit elements, and the nodes of the network correspond to the interconnections of the circuit elements. The electrical performance of the network model is characterized by three types of equations. *Kirchhoff's current law* (KCL) states that for each node of the network, the currents flowing in and out of that node sum up to zero. *Kirchhoff's voltage law* (KVL) states that for each closed loop of the network, the voltage drops along that loop sum up to zero. *Branch constitutive relations* (BCRs) are equations that characterize the electrical performance of the idealized electrical circuit elements. For example, the BCR of a linear resistor is Ohm's law. The BCRs are linear equations for simple devices, such as linear resistors, capacitors, and inductors, and they are nonlinear equations for more complex devices, such as diodes and transistors. In general, the BCRs involve first time-derivatives of the unknowns and are thus first-order *ordinary differential equations* (ODEs). On the other hand, the KCLs and KVLs are linear algebraic equations that only depend on the topology of the network. The KCLs, KVLs, and BCRs can be summarized as a system of first-order, in general nonlinear, *differential-algebraic equations* (DAEs) of the form

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

together with suitable initial conditions. Here, f and q are vector-valued functions, each with N scalar component functions[1], and the unknown $x = x(t)$ is a vector-valued function of length N the entries of which are the circuit variables at time t. We stress that (1) is a system of DAEs rather than ODEs due to the fact that all KCLs and KVLs and the BCRs of some elements (e.g., resistors) are algebraic equations. In particular, the Jacobian $E = \mathrm{D}_x q(x,t)$ of $q(x,t)$ with respect to x is a singular matrix in general.

The numerical solution of systems (1) is the central task in circuit simulation. This is a very challenging task for a number of reasons. The electrical performance of circuits typically involves vastly different time scales, resulting in equations (1) that exhibit *stiffness* in general. Only a small fraction of the huge arsenal of methods for solving nonstiff ODEs are suitable for stiff DAEs. In particular, implicit methods need to be used. These are computationally expensive since the solution of a system of N algebraic equations for N unknowns is required at each time step. Moreover, since these systems are nonlinear in general, some variant of Newton's method needs to be employed, which in turn involves the solution of a system of N linear algebraic equations for N unknowns at each Newton step. Finally, the number N of circuit variables is so large that special algorithms for large-scale matrix computations need to be used in order to make the numerical solution of systems (1) feasible.

1.2 Large-Scale Matrix Computations and Krylov-Subspace Methods

The archetype of a matrix computation is the numerical solution of linear systems of equations

$$M z = b. \tag{2}$$

Here, M is a given $N \times N$ matrix, b is a given vector of length N, and z is the unknown solution vector of (2). For small to moderately large N, the standard approach for computing z is Gaussian elimination, which is based on factoring M into a product of a lower-triangular matrix L and an upper-triangular matrix U. For problems (2) with large N that actually arise in meaningful applications, the matrices M usually exhibit special structures, such as sparsity. An $N \times N$ matrix M is called *sparse* if only a small fraction of its N^2 entries are nonzero. The problem (2) is said to be *large-scale* if its solution z can be computed only by employing algorithms that exploit the special structure of M.

As we discussed in Sect. 1.1, the numerical solution of circuit equations (1) requires the repeated solution of linear systems of the form (2). For realistic circuit simulations, all these linear systems are large-scale and have sparse coefficient ma-

[1] We use the upper-case letter N for the number of components to indicate that this number is large in circuit simulation. The lower-case letter n is used to denote the iteration index in Krylov-subspace methods.

trices M. Furthermore, the matrices M are such that the linear systems (2) can be solved by means of variants of Gaussian elimination that are adapted to sparse matrices. The key feature of these variants is to generate reorderings of the rows and columns of M such that the triangular factors L and U of the reordered version of M remain reasonably sparse. For general sparse matrices, such reorderings are not always possible. However, for matrices M arising in circuit simulation, sparse Gaussian elimination works amazingly well and produces triangular factors L and U that are nearly as sparse as M. In fact, all circuit simulators employ some form of sparse Gaussian elimination to solve the large-scale linear systems (2) that arise in the context of the numerical solution of (1).

For general matrix computations, the same terminology as for linear systems (2) is used. A matrix computation problem is said to be *large-scale* if it can be solved only by employing algorithms that exploit special structures of the matrices describing the problem; see, e.g., [20].

One of the most versatile tools for large-scale matrix computations are iterative methods based on Krylov subspaces. Let M be a given $N \times N$ matrix and r be a given vector of length N. For any $n = 1, 2, \ldots$, the subspace of the space of vectors of length N that is spanned by the vectors

$$r, Mr, M^2 r, \ldots, M^{n-1} r \tag{3}$$

is called the *n-th Krylov subspace* (induced by M and r) and denoted by $\mathcal{K}_n(M, r)$. For many large-scale matrix computations arising in actual applications, very good approximate solutions of the large-scale problem in N-dimensional space can be obtained by solving corresponding n-dimensional problems that are obtained by means of n-th Krylov subspaces $\mathcal{K}_n(M, r)$ with $n \ll N$. However, the basis (3) used to define $\mathcal{K}_n(M, r)$ is not suitable for actual computations since the vectors (3) quickly become linearly dependent in finite-precision arithmetic as n increases. Instead, so-called Krylov-subspace methods are employed to generate more suitable bases. An important feature of these methods is that the matrix M is used only in the form of matrix-vector products with M and possibly with the transpose M^T of M. In particular, these products can be computed cheaply when M is sparse.

The two classical Krylov-subspace methods, the Lanczos algorithm [27] and the Arnoldi process [3], were introduced in the early 1950s in the context of iterative methods for systems of linear equations and eigenvalue computations. The Arnoldi process produces an orthonormal (and thus optimal) basis for $\mathcal{K}_n(M, r)$. Since the construction of such an orthonormal basis involves $(n+1)$-term recurrences of vectors of length N, the Arnoldi process requires $\mathcal{O}\left(n^2 N\right)$ operations, which makes its use problematic for very large-scale problems. The Lanczos algorithm generates a pair of bases, one for $\mathcal{K}_n(M, r)$ and one for the n-th Krylov subspace $\mathcal{K}_n(M^T, l)$ induced by M^T and a second given vector l of length N. The vectors of the two bases are constructed to be biorthogonal to each other, but neither one of the two Lanczos bases is orthonormal. As a result, the Lanczos bases are not as well-behaved in actual computations as the Arnoldi basis. However, since the construction of such biorthogonal bases can be done with three-term recurrences of vectors of length N, the Lanczos algorithm process requires only $\mathcal{O}(nN)$ operations, which allows its use for much larger problems than the Arnoldi process.

In the 6 decades since the introduction of the Lanczos algorithm and the Arnoldi process, Krylov-subspace methods have been studied extensively and have proven to be useful in many other applications besides the solution of systems of linear equations and eigenvalue computations. For example, the Lanczos algorithm was shown to be closely related to Padé approximation of transfer functions of single-input single-output time-invariant linear dynamical systems; see, e.g., Gragg's 1974 paper [24]. This so-called Lanczos-Padé connection is the basis for the PVL algorithm described in Sect. 2.2.

1.3 The Special Case of Circuit Interconnect Analysis

Given the success of sparse Gaussian elimination in solving the linear systems arising in the context of general circuit equations (1), there never was a need for even considering the use of Krylov-subspace methods for solving these linear systems. Nevertheless, in the early 1990s, Krylov-subspace methods turned out to be very efficient tools for tackling the special case of circuit equations (1) that arise in circuit interconnect analysis.

Integrated circuits contain tiny on-chip "wires" to connect transistors and other components to each other. This on-chip wiring is called the circuit *interconnect*. As the number of transistors on a single chip evolved from tens of transistors in the late 1950s to tens of billions of transistors in 2019, the amount of interconnect increased accordingly. A state-of-the-art chip in 2019 contains interconnect wires with a total length of tens of miles. Interconnect analysis uses simulation to verify and correct the interconnect design of a chip in order to ensure that the on-chip wiring does not interfere with the proper functioning of the chip.

Circuit interconnect analysis employs the lumped-element approach described in Sect. 1.1 to model interconnect structures as *RCL networks* of resistors, capacitors, and inductors that correspond to small pieces of wires of the overall interconnect. Since the BCRs of all these circuit elements are linear time-invariant equations, the electrical performance of the interconnect network is described by a system of equations of the form (1) with functions f and q that are linear in x. Moreover, the main interest in interconnect analysis is the input-output behavior of the interconnect. Given input functions, such as the voltages of voltage sources and the currents of current sources, which drive the interconnect, the task is to compute certain output functions, such as the currents of the voltage sources and the voltages of the current sources. In this case, the input-output behavior of the interconnect is described by a system of linear DAEs of the form

$$\begin{aligned} E\,\frac{\mathrm{d}}{\mathrm{d}t}x &= Ax + Bu(t), \\ y(t) &= L^T x(t), \end{aligned} \tag{4}$$

together with suitable initial conditions. Here, A and E are $N \times N$ matrices, B is an $N \times m$ matrix, L is an $N \times p$ matrix, u is a vector-valued function of length m, y is a vector-valued function of length p, and the unknown $x = x(t)$ is a vector-valued function of length N the entries of which are the circuit variables at time t. The m entries of $u = u(t)$ are the given input functions, and the p entries of $y = y(t)$ are the output functions of interest. The numbers $m \geq 1$ and $p \geq 1$ are small and $m, p \ll N$. In general. the matrix E is singular and thus (4) is a system of DAEs, rather than ODEs. Finally, we always assume that the *matrix pencil*

$$sE - A, \quad s \in \mathbb{C}, \tag{5}$$

is *regular*, i.e., the matrix $sE - A$ is singular only for finitely many values of $s \in \mathbb{C}$. Here, $\mathbb{C}$ denotes the set of all complex numbers. The assumption of regularity of the matrix pencil (5) is satisfied for any realistic circuit interconnect simulation, see, e.g., [17].

Systems of equations of the form (4) are called *m-input p-output linear time-invariant linear dynamical systems*. They arise in many applications and not just in circuit simulation. However, in most applications the size N of (4) is small or only of moderate size, whereas one has to deal with large-scale systems in circuit simulation. In the large-scale case, the main interest is usually in the input-output behavior $u(t) \ \rightarrow \ y(t)$ of the system (4), rather than the complete solution vector $x(t)$. In fact, for very large N, it may not even be feasible to compute x. A standard approach to tackle large-scale systems (4) is to employ model order reduction; see, e.g. [38]. The basic idea is to replace the quantities of size N in (4) by corresponding quantities of size n. More precisely, a *reduced-order model* (ROM) of (4) is a system of the form

$$\begin{aligned} E_n \frac{\mathrm{d}}{\mathrm{d}t} \tilde{x}(t) &= A_n \tilde{x}(t) + B_n u(t), \\ \tilde{y}(t) &= L_n^T \tilde{x}(t), \end{aligned} \tag{6}$$

where A_n and E_n are $n \times n$ matrices, B_n is an $n \times m$ matrix, L_n is an $n \times p$ matrix, and $n \ll N$. Note that $u = u(t)$ is the same given input function in both the original system (4) and its ROM (6). The challenge of model order reduction is to find a value $n \ll N$ and matrices A_n, E_n, B_n, and L_n such that

$$\tilde{y}(t) \ \approx \ y(t) \quad \text{for all `relevant' times } \ t. \tag{7}$$

A standard approach in model order reduction of time-invariant linear dynamical systems is to transform (4) from time domain into complex Laplace domain. Applying the Laplace transform to (4), we obtain the Laplace-domain system

$$\begin{aligned} sE\hat{x}(s) &= A\hat{x}(s) + B\hat{u}(s), \\ \hat{y}(s) &= L^T \hat{x}(s), \end{aligned} \tag{8}$$

where $s \in \mathbb{C}$. Elimination of $\hat{x}(s)$ from (8) results in the Laplace-domain input-output relation

$$\hat{y}(s) = H(s)\,\hat{u}(s), \tag{9}$$

where

$$H : \mathbb{C} \mapsto \left(\mathbb{C} \cup \infty\right)^{p\times m}, \quad H(s) := L^T\left(sE - A\right)^{-1}B. \tag{10}$$

The function (10) is called the *transfer function* of the time-invariant linear dynamical system (4). We remark that H is a $(p \times m)$-matrix-valued rational function with potential poles at the finitely many values of $s \in \mathbb{C}$ for which the matrix $sE - A$ is singular. Analogously, the Laplace-domain input-output relation of the ROM (6) is given by

$$\hat{\tilde{y}}(s) = H_n(s)\,\hat{u}(s), \tag{11}$$

where

$$H_n : \mathbb{C} \mapsto \left(\mathbb{C} \cup \infty\right)^{p\times m}, \quad H_n(s) := L_n^T\left(sE_n - A_n\right)^{-1}B_n, \tag{12}$$

is the transfer function of the ROM (6). Finally, in view of (9) and (11), the desired approximation property (7) in time domain translates into the approximation property

$$H_n(s) \;\approx\; H(s) \quad \text{for all 'relevant' values of } s \in \mathbb{C} \tag{13}$$

in Laplace domain.

In control theory, the problem of constructing good approximations H_n of H in (13) has been studied extensively and many powerful methods have been developed. However, only few of these approaches are feasible in the large-scale case. The approach that is relevant for circuit interconnect analysis is *moment matching*. It is based on selecting a suitable expansion point $s_0 \in \mathbb{C}$ and then constructing H_n such that the Taylor series of $H_n(s)$ and $H(s)$ about s_0 agree in as many of their leading Taylor coefficients as possible.

1.4 Outline

In the first three decades of integrated circuits, it was sufficient to use a simple metric, the Elmore delay [7], to capture the effects of interconnect. Around 1990, the complexity of integrated circuits had reached the point where this simple metric was no longer accurate enough. Building on the concept of Elmore delay, *asymptotic waveform evaluation* (AWE) [32, 34] was proposed. Unfortunately, the initial excitement over AWE was followed by the disappointment that actual implementations of the method did not perform as expected. The remedy for these numerical problems was the *Padé via Lanczos* (PVL) algorithm [8, 9], which is based on the classical Lanczos algorithm [27]. In Sect. 2, we describe AWE and the PVL algorithm for the case $m = p = 1$ of single-input single-output linear time-invariant linear dynamical systems (4). The success of PVL quickly led to the question of how to extend the PVL algorithm to the case of general m-input p-output systems. Surprisingly, a corresponding extension of the classical Lanczos algorithm for general $m, p \geq 1$ did not exist at that time, and so a new such Krylov-subspace method, the *band Lanczos method*, was developed. We describe the underlying concept of block Krylov subspaces for multiple starting vectors in Sect. 3 and the band Lanczos method itself in Sect. 4. An important issue in interconnect simulation is to preserve crucial properties, such as passivity and reciprocity, of the interconnect network model in the ROMs that are constructed via suitable Krylov-subspace methods. In Sect. 5, we discuss the problem of structure preservation in reduced-order interconnect models and the construction of such structure-preserving models by means of explicit projections. In Sect. 6, we describe a new Krylov-subspace method, the *band Arnoldi process*, that was developed to facilitate reliable implementations of such projection approaches. Finally, in Sect. 7, we mention some open problems and make some concluding remarks.

2 From AWE to the PVL Algorithm

In this section, we consider only circuit interconnect models with single input and single output functions. The system of DAEs describing such models is given by (4) with $m = p = 1$. Since B and L in (4) are vectors in this case, we use b and l instead of upper-case letters. The system of DAEs is thus of the form

$$\begin{aligned} E\,\frac{\mathrm{d}}{\mathrm{d}t}x &= Ax + b\,u(t), \\ y(t) &= l^T x(t), \end{aligned} \tag{14}$$

and its transfer function is given by

$$H : \mathbb{C} \mapsto \big(\mathbb{C} \cup \infty\big), \quad H(s) := l^T \big(sE - A\big)^{-1} b. \tag{15}$$

Note that H is a scalar rational function.

2.1 Elmore Delay and AWE

Until the late 1980s, it was sufficient to model circuit interconnect as *RC networks*, i.e., networks that contain only resistors and capacitors, but no inductors, and the Elmore delay was used as a simple metric for such RC networks. In this approach, the RC network model of the interconnect is replaced by a simple reduced model that contains only a single resistor with resistance R and a single capacitor with capacitance C. The product of R and C is the actual Elmore delay. This whole process can be viewed as the construction of an approximation of the form

$$H_1(s) = \frac{a_1}{s - b_1} \tag{16}$$

to the transfer function H of the RC network model. Here a_1 and b_1 are real parameters that are determined such that the leading two Taylor coefficients of the Taylor series of $H_1(s)$ and $H(s)$ about the expansion point $s_0 = 0$ match:

$$H_1(s) = H(s) + \mathscr{O}\big(s^2\big).$$

The values of R and C are readily obtained from a_1 and b_1; see, e.g., [18].

AWE generalizes the simple approximation (16) to rational functions of the form

$$H_n(s) = \frac{a_1}{s-b_1} + \frac{a_2}{s-b_2} + \cdots + \frac{a_n}{s-b_n}, \tag{17}$$

where the $2n$ parameters $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ are determined such that the leading $2n$ Taylor coefficients of the Taylor series of $H_n(s)$ and $H(s)$ about some suitable expansion point $s_0 \in \mathbb{C}$ match:

$$H_n(s) = H(s) + \mathscr{O}\big((s-s_0)^{2n}\big). \tag{18}$$

In principle, any $s_0 \in \mathbb{C}$, except for the finitely many poles of H, can be chosen as expansion point in (18). For interconnect models, all poles of H have negative real parts and thus any s_0 with nonnegative real part is a safe choice. Since real values of s_0 are preferable in order to avoid complex arithmetic, expansion points $s_0 \geq 0$ are chosen in practice.

In theory, by increasing n in (17) until a sufficiently accurate approximation H_n of H is obtained, AWE should be able to easily handle much more complex interconnect models than the Elmore delay. However, in practice, the accuracy of H_n tends to stagnate already at modest values of n. The reason for this behavior is not the defining property (18) of H_n, but the algorithm that is used in AWE to compute H_n. AWE first explicitly generates the leading $2n$ coefficients (the so-called *moments*) of the Taylor series of $H(s)$ about the expansion point s_0 and then constructs the values of the $2n$ parameters in (17) such that H_n has the same $2n$ moments as H. Unfortunately, the computation of the moments is extremely sensitive to numerical round-off error and is viable only for very small values of n. For a detailed discussion of the numerical issues of AWE, we refer the reader to [9].

We remark that a function H_n defined by (17) and (18) is called a *Padé approximant* of H; see, e.g., [4]. The PVL algorithm generates the same Padé approximant H_n as AWE, but does so without computing the moments.

2.2 PVL Algorithm

The basis of the PVL algorithm is the connection between Padé approximants H_n to transfer functions H of the form (15) and the Lanczos algorithm.

Recall from Sect. 1.2 that the Lanczos algorithm involves an $N \times N$ matrix M and two vectors r and l of length N, and thus we rewrite (15) as follows:

$$\begin{aligned} H(s) &= l^T\big(sE-A\big)^{-1}b = l^T\big(I+(s-s_0)M\big)^{-1}r, \\ \text{where}\quad M &:= \big(s_0E-A\big)^{-1}E, \quad r := \big(s_0E-A\big)^{-1}b, \end{aligned} \tag{19}$$

and I denotes the identity matrix of the same size as M.

Running n iterations of the Lanczos algorithm (with M, r, and l from (19)) generates a pair of biorthogonal bases for the subspaces $\mathscr{K}_n(M,r)$ and $\mathscr{K}_n(M^T,l)$. The scalars in the three-term recurrences used to construct these bases is all that is needed to obtain H_n. More precisely,

$$H_n = (l^T r)\, e_1^T\big(I+(s-s_0)\,T_n\big)^{-1}e_1, \tag{20}$$

where e_1 denotes the first unit vector of length n and

$$T_n = \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 \\ \rho_2 & \alpha_2 & \beta_3 & \ddots & \vdots \\ 0 & \rho_3 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_n \\ 0 & \cdots & 0 & \rho_n & \alpha_n \end{bmatrix} \tag{21}$$

is an $n \times n$ tridiagonal matrix whose entries are computed during the first n iterations of the following algorithm.

Algorithm 1 (Lanczos algorithm)

```
Set $\hat{v}_1 = r$, $\hat{w}_1 = l$, $v_0 = w_0 = 0$, and $\delta_0 = 1$.
For $n = 1, 2, \dots,$ do:
1) Compute $\rho_n = \|\hat{v}_n\|_2$ and $\eta_n = \|\hat{w}_n\|_2$.
   If $\rho_n = 0$ or $\eta_n = 0$, set $n = n-1$ and stop.
   Otherwise, set $v_n = \hat{v}_n/\rho_n$ and $w_n = \hat{w}_n/\eta_n$.
2) Compute $\delta_n = w_n^T v_n$.
   If $\delta_n = 0$, stop: look-ahead would be needed to continue.
3) Compute $\hat{v}_{n+1} = M v_n$.
   Set $\beta_n = \eta_n\delta_n/\delta_{n-1}$ and $\hat{v}_{n+1} = \hat{v}_{n+1} - v_{n-1}\beta_n$.
4) Compute $\alpha_n = w_n^T \hat{v}_{n+1}/\delta_n$.
   Set $\hat{v}_{n+1} = \hat{v}_{n+1} - v_n\alpha_n$.
5) Compute $\hat{w}_{n+1} = M^T w_n$.
   Set $\gamma_n = \rho_n\delta_n/\delta_{n-1}$ and $\hat{w}_{n+1} = \hat{w}_{n+1} - w_n\alpha_n - w_{n-1}\gamma_n$.
```

For details and properties of the Lanczos algorithm, the reader is referred to [20, Sect. 64.5] or [37, Sect. 7.1]. Next, we list some facts about Algorithm 1 that are relevant for its use in the PVL algorithm and its extension to the band Lanczos method in Sect. 4:

1. In exact arithmetic, the algorithm terminates after finitely many iterations. Since $\mathscr{K}_n(M,r)$ and $\mathscr{K}_n(M^T,l)$ are subspaces of N-dimensional space, their dimensions

cannot exceed N. As a result, the check in step 1) is satisfied for some $n \leq N+1$. If $\rho_n = 0$, then $\mathscr{K}_{n-1}(M,r)$ has reached its maximum dimension $n-1$. If $\eta_n = 0$, then $\mathscr{K}_{n-1}(M^T,l)$ has reached its maximum dimension $n-1$.

2. In general, the algorithm may stop prematurely due to $\delta_n = 0$ in step 2). Such an event is called an *exact breakdown*. In practice, one also needs to stop if $\delta_n \neq 0$, but $\left|\delta_n\right|$ is 'close' to 0. Such an event is called a *near-breakdown*. Exact breakdowns and near-breakdowns can be avoided altogether by employing so-called 'look-ahead' strategies; see [22] and the references given there. The resulting *look-ahead* Lanczos algorithm is necessarily quite a bit more involved than Algorithm 1. To keep the exposition simple, we only discuss the Lanczos algorithm and the band Lanczos method without look-ahead.
3. The vectors $v_1, v_2, \ldots, v_n$ form a basis of $\mathscr{K}_n(M,r)$, and the vectors $w_1, w_2, \ldots, w_n$ form a basis of $\mathscr{K}_n(M^T,l)$. In exact arithmetic, the two bases are biorthogonal to each other. Using the notation

$$V_n := \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} \quad \text{and} \quad W_n := \begin{bmatrix} w_1 & w_2 & \cdots & w_n \end{bmatrix}, \tag{22}$$

the biorthogonality of the two bases can be stated compactly as follows:

$$W_n^T V_n = \Delta_n := \operatorname{diag}\bigl(\delta_1, \delta_2, \ldots, \delta_n\bigr) = \begin{bmatrix} \delta_1 & 0 & \cdots & 0 \\ 0 & \delta_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \delta_n \end{bmatrix}. \tag{23}$$

The recurrences to that are used in to generate the vectors $v_1, v_2, \ldots, v_n, \hat{v}_{n+1}$ and $w_1, w_2, \ldots, w_n, \hat{w}_{n+1}$ can be stated compactly as follows:

$$\begin{aligned} MV_n &= V_n T_n + \begin{bmatrix} 0 & 0 & \cdots & 0 & \hat{v}_{n+1} \end{bmatrix}, \\ M^T W_n &= W_n \tilde{T}_n + \begin{bmatrix} 0 & 0 & \cdots & 0 & \hat{w}_{n+1} \end{bmatrix}. \end{aligned} \tag{24}$$

Here T_n is the tridiagonal matrix (21) and $\tilde{T}_n$ is the tridiagonal matrix given by

$$\tilde{T}_n = \begin{bmatrix} \alpha_1 & \gamma_2 & 0 & \cdots & 0 \\ \eta_2 & \alpha_2 & \gamma_3 & \ddots & \vdots \\ 0 & \eta_3 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \gamma_n \\ 0 & \cdots & 0 & \eta_n & \alpha_n \end{bmatrix}.$$

These two tridiagonal matrices are related as follows:

$$\Delta_n T_n = \tilde{T}_n^{\,T} \Delta_n. \tag{25}$$

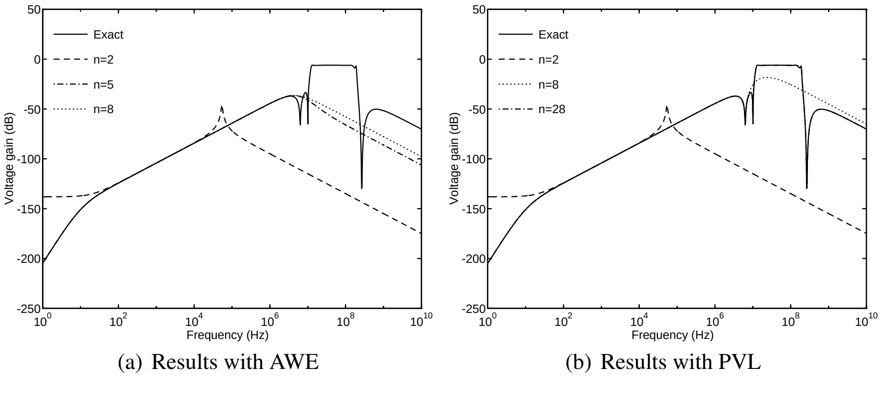
4. By multiplying the first equation in (24) from the left by W_n^T and by using (23) and $W_n^T \hat{v}_{n+1} = 0$, we obtain the expression

$$T_n = \left(W_n^T V_n\right)^{-1} W_n^T M V_n = \Delta_n^{-1} W_n^T M V_n \tag{26}$$

for T_n. Since the columns of V_n and W_n are biorthogonal bases of $\mathscr{K}_n(M,r)$ and $\mathscr{K}_n(M^T,l)$, the relation (26) means that the $n \times n$ matrix T_n is the *oblique projection* of the $N \times N$ matrix M onto the subspace $\mathscr{K}_n(M,r)$ and orthogonally to the subspace $\mathscr{K}_n(M^T,l)$.
5. For the PVL algorithm, only the tridiagonal matrix T_n is needed. Its entries are generated as scalar coefficients of the three-term recurrences that are used to produce the two biorthogonal bases. In order to run these recurrences, only the 6 vectors $v_{n-1}, v_n, \hat{v}_{n+1}, w_{n-1}, w_n, \hat{w}_{n+1}$ need to be stored at any stage of Algorithm 1.
6. Each iteration of Algorithm 1 requires one matrix-vector product with M and one with M^T. For the PVL algorithm, M is of the form $M = \left(s_0 E - A\right)^{-1} E$, where A and E are large-scale sparse matrices. To compute the matrix-vector products efficiently in this case, one employs sparse Gaussian elimination to precompute a sparse LU factorization of the matrix $s_0 E - A$. Each matrix-vector product with M or M^T can then be computed cheaply via one multiplication with a sparse matrix and two sparse triangular solves.

2.3 An Example

The following example, which is taken from [8, 9], illustrates the numerical differences between AWE and the PVL algorithm. The circuit simulated here is a voltage filter, where the frequency range of interest is $1 \le \omega \le 10^{10}$. In Fig. 1(a) we show



(a) Results with AWE (b) Results with PVL

Fig. 1 Simulation of a voltage filter

the exact function $|H(\mathrm{i}\,\omega)|$ and the approximations $|H_n(\mathrm{i}\,\omega)|$ generated by AWE for $n = 2, 5, 8$. Note that H_8 has clearly not yet converged to H. It turns out that the H_n's practically do not change anymore for $n \geq 8$, and so AWE never converges in this example. In Fig. 1(b) we show the exact function $|H(\mathrm{i}\,\omega)|$ and the approximations $|H_n(\mathrm{i}\,\omega)|$ generated by the PVL algorithm for $n = 2, 8, 28$. Note that the results for $n = 8$ (the dotted curves) in Fig. 1(a) and Fig. 1(b) are vastly different, although they both correspond to the same function H_8. Furthermore, note that the algorithm PVL converges, with the computed Padé approximant H_{28} being practically identical to H.

3 Krylov Subspaces with Multiple Starting Vectors

While the PVL algorithm remedies the numerical issues of AWE, it can be used only for the special case of single-input single-output systems (14). Since circuit interconnect models have multiple inputs and outputs in general, after its introduction in 1994, it quickly became clear that the PVL algorithm needed to be extended to m-input p-output systems (4). To motivate the type of Krylov-subspace method that is needed for such an extension, we first rewrite the transfer function (10) of (4) as follows:

$$\begin{aligned} H(s) &= L^T \big(sE - A\big)^{-1} B = L^T \big(I + (s - s_0)M\big)^{-1} R, \\ \text{where} \quad M &:= \big(s_0 E - A\big)^{-1} E \quad \text{and} \quad R := \big(s_0 E - A\big)^{-1} B. \end{aligned} \tag{27}$$

Note that M is an $N \times N$ matrix, R is an $N \times m$ matrix, L is an $N \times p$ matrix, and H is a $(p \times m)$-matrix-valued function. Instead of starting vectors r and l in the PVL algorithm, we now have blocks R and L of *multiple starting vectors*. A Lanczos-type algorithm that produces Padé approximants H_n of H via a suitable adaption of the PVL formula (4) to the m-input p-output case needs to be able to handle such multiple starting vectors.

In this section, we describe the concept of block Krylov subspaces for multiple starting vectors and briefly review the *block Lanczos method*.

3.1 Block Krylov Subspaces

Recall that for a single starting vector r, the n-th Krylov subspace $\mathscr{K}_n(M, r)$ is defined as the n-dimensional subspace spanned by the vectors (3). Here, $1 \leq n \leq n_{\max}$ and $n_{\max}$ denotes the maximum value of n such that the vectors (3) are still linearly independent.

For a block R of m starting vectors, we have $N \times m$ matrices $M^i R$, $i = 0, 1, \ldots,$ instead of vectors (3). To properly define Krylov subspaces $\mathscr{K}_n(M, R)$ in this case, we put the first N of these matrices into a single $N \times mN$ *right block Krylov matrix* as follows:

$$\begin{bmatrix} R & MR & M^2R & \cdots & M^{N-1}R \end{bmatrix}. \tag{28}$$

Next, we scan the mN columns of this matrix from left to right and delete any column that is linearly dependent on columns to its left. The result of this operation is the matrix

$$\begin{bmatrix} R_1 & MR_2 & M^2R_3 & \cdots & M^{i_{\max}-1}R_{i_{\max}} \end{bmatrix} \tag{29}$$

the columns of which are all linearly independent. This process of detecting and deleting the linearly dependent columns of the matrix (28) is called *exact deflation*. Note that a column of the form $M^i r$ being linearly dependent on columns to its left in (28) implies that any column $M^j r$, $j = i, i+1, \ldots,$ is linearly dependent on columns to its right. Therefore, in (29), for each $i = 1, 2, \ldots, i_{\max}$, the matrix R_i is a submatrix of R_{i-1}, where, for $i = 1$, we set $R_0 = R$. Denoting by m_i the number of columns of R_i, the number of columns of the matrix (29) is given by

$$n_{\max}^{(\mathrm{R})} := m_1 + m_2 + \cdots + m_{i_{\max}}. \tag{30}$$

By construction, the matrix (29) has full column rank $n_{\max}^{(\mathrm{R})}$.

For $n = 0, 1, \ldots, n_{\max}^{(\mathrm{R})}$, the n-dimensional subspace of the space of vectors of length N that is spanned by the first n columns of the matrix (29) is called the *n-th block Krylov subspace* (induced by M and R) and denoted by $\mathcal{K}_n(M, R)$.

To define $\mathcal{K}_n(M^T, L)$, where L is a block of p starting vectors, we proceed analogously. Applying the process of exact deflation to the *left block Krylov matrix*

$$\begin{bmatrix} L & M^TL & (M^T)^2L & \cdots & (M^T)^{N-1}L \end{bmatrix},$$

we obtain an $N \times n_{\max}^{(\mathrm{L})}$ matrix of the form

$$\begin{bmatrix} L_1 & M^TL_2 & (M^T)^2L_3 & \cdots & (M^T)^{j_{\max}-1}L_{j_{\max}} \end{bmatrix}, \tag{31}$$

where each L_j is a submatrix of L_{j-1}. The matrix (31) has full column rank

$$n_{\max}^{(\mathrm{L})} := p_1 + p_2 + \cdots + p_{j_{\max}}, \tag{32}$$

where p_j denotes the number of columns of L_j.

For $n = 0, 1, \ldots, n_{\max}^{(\mathrm{L})}$, the n-dimensional subspace of the space of vectors of length N that is spanned by the first n columns of the matrix (31) is called the *n-th block Krylov subspace* (induced by M^T and L) and denoted by $\mathcal{K}_n(M^T, L)$.

To distinguish the two types of block Krylov subspaces, we refer to $\mathcal{K}_n(M, R)$ and $\mathcal{K}_n(M^T, L)$ as *right* and *left* block Krylov subspaces, respectively.

3.2 Block Lanczos Method

In 1994, the problem of extending the Lanczos process for single to multiple starting vectors was not new, and a number of algorithms had been proposed. With the exception of Ruhe's band variant [36] of the symmetric Lanczos algorithm, all existing algorithms at that time are based on a block-wise construction of basis vectors for the underlying block Krylov subspaces.

For symmetric matrices $M = M^T$ and starting vectors $r = l$, the right and left Krylov subspaces $\mathscr{K}_n(M,r)$ and $\mathscr{K}_n(M^T,l)$ are identical, and the general Lanczos algorithm simplifies to the *symmetric* Lanczos algorithm. The first block extensions of the Lanczos algorithm were developed for this special case [5, 39, 23, 6]. Since the matrices M in (27) are nonsymmetric in general, block variants of the symmetric Lanczos algorithm cannot be used to extend the PVL algorithm to m-input p-output systems (4).

For the general case, Kim and Craig [25, 26] were the first to develop a block version of the classical Lanczos algorithm. Their block Lanczos method requires that $m = p$ and is essentially a variant of Algorithm 1, where vectors are replaced by blocks of m vectors, scalars are replaced by $m \times m$ matrices, and division by a scalar is replaced by multiplication with the inverse of an $m \times m$ matrix. The $m \times m$ matrices are chosen such that the generated blocks of basis vectors for the right and left block Krylov subspaces are block-biorthogonal to each other. Clearly, such a block approach cannot be extended to the case $m \neq p$, as this would involve 'inverses' of nonsquare matrices. Furthermore, even for the special case $m = p$, the block Lanczos method requires that the sizes for the right and left blocks of basis vectors remain the same throughout the run of the algorithm. As a result, necessary deflations to handle linearly dependent blocks can only be performed if these linear dependencies occur simultaneously in the right and left blocks. However, this is not the case in general.

In order to extend the PVL algorithm to general m-input p-output systems of the form (4), a new Lanczos-type method was needed to overcome the limitations of the block Lanczos method. Such a procedure needs to be able to handle the general case $m, p \geq 1$ and include an efficient deflation procedure. The band Lanczos method, which we describe in Sect.. 4, is such a procedure. The key to the development of the band Lanczos method was the insight to construct the basis vectors of the right and left block Krylov subspaces to be vectorwise biorthogonal to each other, instead of the blockwise biorthogonality that is used in the block Lanczos method.

4 A New Approch: the Band Lanczos Method

As in Sect. 3, we assume that M is an $N \times N$ matrix, R is an $N \times m$ matrix, and L is an $N \times N$ matrix. We use the notation

$$R = \begin{bmatrix} r_1 & r_2 & \cdots & r_m \end{bmatrix} \quad \text{and} \quad L = \begin{bmatrix} l_1 & l_2 & \cdots & l_p \end{bmatrix} \tag{33}$$

for the columns of R and L.

4.1 *Defining Properties*

Like Algorithm 1, the band Lanczos method generates two sets of *right* and *left Lanczos vectors*

$$v_1, v_2, \ldots, v_n \quad \text{and} \quad w_1, w_2, \ldots, w_n \tag{34}$$

that are constructed to be (vectorwise) biorthogonal to each other. If only exact deflations are performed in the method, the vectors (34) form bases of the right and left block Krylov subspaces $\mathscr{K}_n(M,R)$ and $\mathscr{K}_n(M^T,L)$. Using the notation from (22) and (23), the biorthogonality of the vectors (34) can be stated compactly as

$$W_n^T V_n = \Delta_n := \operatorname{diag}\bigl(\delta_1, \delta_2, \ldots, \delta_n\bigr). \tag{35}$$

At any stage of the band Lanczos method, there are *right* and *left candidate vectors*

$$\hat{v}_{n+1}, \hat{v}_{n+2}, \ldots, \hat{v}_{n+m_c} \quad \text{and} \quad \hat{w}_{n+1}, \hat{w}_{n+2}, \ldots, \hat{w}_{n+p_c} \tag{36}$$

for the Lanczos vectors $v_{n+1}, v_{n+2}, \ldots, v_{n+m_c}$ and $w_{n+1}, w_{n+2}, \ldots, w_{n+p_c}$ to be generated in the following iterations. The vectors (36) are constructed to be biorthogonal to the Lanczos vectors (34):

$$W_n^T \hat{v}_{n+j} = 0,\ j = 1,2,\ldots,m_c, \quad \text{and} \quad V_n^T \hat{w}_{n+k} = 0,\ k = 1,2,\ldots,p_c. \tag{37}$$

At the start of the algorithm, the right and left candidate vectors are initialized as the columns of R and L in (33), $m_c = m$, and $p_c = p$.

The candidate vectors (36) allow for an easy way to check for necessary deflations. The next exact deflation in the right block Krylov matrix (28) occurs if, and only if, $\hat{v}_{n+1} = 0$. The next exact deflation in the left block Krylov matrix (31) occurs if, and only if, $\hat{w}_{n+1} = 0$. In practice, one also needs to perform deflations when these vectors are 'close' to zero vectors. In an actual algorithm, we check if

$$\|\hat{v}_{n+1}\|_2 \le \mathtt{dftol}_{\mathrm{v}} \quad \text{or} \quad \|\hat{w}_{n+1}\|_2 \le \mathtt{dftol}_{\mathrm{w}}, \tag{38}$$

where $\mathtt{dftol}_{\mathrm{v}}, \mathtt{dftol}_{\mathrm{w}}$ are suitably small *deflation tolerances*. If the first check in (38) is true, $\hat{v}_{n+1}$ is labeled a *deflated* right vector, the indices of $\hat{v}_{n+2}, \ldots, \hat{v}_{n+m_c}$ are shifted by -1, and m_c is reduced by 1. If the second check in (38) is true, $\hat{w}_{n+1}$ is labeled a deflated left vector, the indices of $\hat{w}_{n+2}, \ldots, \hat{w}_{n+p_c}$ are shifted by -1, and p_c is reduced by 1. We refer to this process as *deflation* in general, and as exact deflation when both deflation tolerances in (38) are set to 0. Note that $m - m_c$ and $p - p_c$ is the number of deflations of right and left vectors, respectively, that have occurred so far.

Similar to the relations (24) for Algorithm 1, the recurrences that are used to generate the vectors (34) and (36) can be stated compactly as follows:

$$\begin{aligned} MV_n &= V_n T_n + \begin{bmatrix} 0 & 0 & \cdots & 0 & \hat{v}_{n+1} & \hat{v}_{n+2} & \cdots & \hat{v}_{n+m_{\mathrm{c}}} \end{bmatrix} + V_n^{(\mathrm{dl})}, \\ M^T W_n &= W_n \tilde{T}_n + \begin{bmatrix} 0 & 0 & \cdots & 0 & \hat{w}_{n+1} & \hat{w}_{n+2} & \cdots & \hat{w}_{n+p_{\mathrm{c}}} \end{bmatrix} + W_n^{(\mathrm{dl})}. \end{aligned} \tag{39}$$

The matrices $V_n^{(\mathrm{dl})}$ and $W_n^{(\mathrm{dl})}$ contain mostly zero columns, together with the $m - m_{\mathrm{c}}$ right and $p - p_{\mathrm{c}}$ left deflated vectors, respectively. In particular, $V_n^{(\mathrm{dl})}$ and $W_n^{(\mathrm{dl})}$ are zero matrices if no deflations have occurred so far or if only exact deflations are performed. The matrices T_n and $\tilde{T}_n$ contain the scalar coefficients of the recurrences that are used to generate the Lanczos vectors and the candidate vectors. Since these recurrences involve at most $m_{\mathrm{c}} + p_{\mathrm{c}} + 1$ terms, the matrices T_n and $\tilde{T}_n$ are 'essentially' banded. More precisely, T_n has lower bandwidth $m_{\mathrm{c}} + 1$ and upper bandwidth $p_{\mathrm{c}} + 1$, where the lower bandwidth is reduced by 1 every time a right vector is deflated and the upper bandwidth is reduced by 1 every time a left vector is deflated. In addition, each deflation of a left vector causes T_n to have nonzero elements in a fixed row outside and to the right of the banded part. Analogously, $\tilde{T}_n$ has lower bandwidth $p_{\mathrm{c}} + 1$ and upper bandwidth $m_{\mathrm{c}} + 1$, where the lower bandwidth is reduced by 1 every time a left vector is deflated, and the upper bandwidth is reduced by 1 every time a right vector is deflated. In addition, each deflation of a right vector causes $\tilde{T}_n$ to have nonzero elements in a fixed row outside and to the right of the banded part.

Recall that the tridiagonal matrices from Algorithm 1 are connected via the relation (25). The banded parts of T_n and $\tilde{T}_n$ in (39) are related in a similar way:

$$\Delta_n T_n^{(\mathrm{pr})} = \left(\tilde{T}_n^{(\mathrm{pr})}\right)^T \Delta_n, \tag{40}$$

where

$$T_n^{(\mathrm{pr})} := T_n + \Delta_n^{-1} W_n^T V_n^{(\mathrm{dl})} \quad \text{and} \quad \tilde{T}_n^{(\mathrm{pr})} := \tilde{T}_n + \Delta_n^{-1} V_n^T W_n^{(\mathrm{dl})}. \tag{41}$$

Note that the matrix $\Delta_n^{-1} W_n^T V_n^{(\mathrm{dl})}$ and $\Delta_n^{-1} V_n^T W_n^{(\mathrm{dl})}$ has nonzero entries only below the banded part of T_n and $\tilde{T}_n$ and in the columns corresponding to the $m - m_{\mathrm{c}}$ deflated right vectors and $p - p_{\mathrm{c}}$ deflated left vectors, respectively.

By multiplying the first equation in (39) from the left by W_n^T and using (35), (37), and (41), we obtain the expression

$$T_n^{(\mathrm{pr})} = \left(W_n^T V_n\right)^{-1} W_n^T M V_n = \Delta_n^{-1} W_n^T M V_n \tag{42}$$

for $T_n^{(\mathrm{pr})}$. The relation (42) means that the $n \times n$ matrix $T_n^{(\mathrm{pr})}$ is the oblique projection of the $N \times N$ matrix M onto the subspace spanned by $v_1, v_2, \ldots, v_n$ and orthogonally to the subspace spanned by $w_1, w_2, \ldots, w_n$. If only exact deflations are performed, then these vectors span the right and left block Krylov subspaces and thus $T_n^{(\mathrm{pr})}$ is the oblique projection of M onto $\mathscr{K}_n(M, R)$ and orthogonally to $\mathscr{K}_n(M^T, L)$.

4.2 Reduced-Order Models and Matrix Padé Approximants

In this subsection, we discuss the use of the band Lanczos method in model order reduction of m-input p-output systems (4). For this application, M, R, and L are the matrices from the representation (27) of the transfer function H of (4). In addition to the oblique projection (42), $T_n^{(\mathrm{pr})}$, of M onto the subspaces generated by n iterations of the band Lanczos method, we also need the one-sided oblique projections of R and L corresponding to (42). These projections are defined as follows:

$$\rho_n^{(\mathrm{pr})} := \Delta_n^{-1} W_n^T R \quad \text{and} \quad \eta_n^{(\mathrm{pr})} := \Delta_n^{-1} V_n^T L.$$

Using the quantities $T_n^{(\mathrm{pr})}$, $\rho_n^{(\mathrm{pr})}$, $\eta_n^{(\mathrm{pr})}$, and Δ_n, we define the approximation

$$H_n(s) = \left(\eta_n^{(\mathrm{pr})}\right)^T \Delta_n \left(I + (s - s_0)\, T_n^{(\mathrm{pr})}\right)^{-1} \rho_n^{(\mathrm{pr})} \tag{43}$$

of H. An actual reduced-order model (ROM) (6) that corresponds to H_n is readily obtained by comparing the representation (12) of the ROM transfer function with (43) and defining the matrices in (6) as follows:

$$A_n := I - s_0 T_n^{(\mathrm{pr})}, \quad E_n := T_n^{(\mathrm{pr})}, \quad B_n := \rho_n^{(\mathrm{pr})}, \quad \text{and} \quad L_n := \Delta_n \eta_n^{(\mathrm{pr})}.$$

We remark that (43) generalizes the PVL formula (20) for the single-input single-output case to the general multiple-input multiple-output case. In fact, for $m = p = 1$, the band Lanczos method reduces to Algorithm 1 and formula (43) reduces to (20).

The ROM transfer function (12), H_n, is called an *n-th matrix Padé approximant* (about the expansion point s_0) of the transfer function H of (4) if

$$H_n(s) = H(s) + \mathscr{O}\big((s - s_0)^{q(n)}\big),$$

where $q(n)$ is as large as possible. The ROM transfer function (43) generated via the band Lanczos method is an n-th matrix Padé approximant of H provided that only exact deflations are performed. To properly state this result, recall the definitions of $n_{\max}^{(\mathrm{R})}$ and $n_{\max}^{(\mathrm{L})}$ in (30) and (32). In addition, we define $j(n)$ and $k(n)$ as the largest values of j and k such that

$$m_1 + m_2 + \cdots + m_j \leq n \quad \text{and} \quad p_1 + p_2 + \cdots + p_k \leq n.$$

Theorem 2. *Let* $\max\{\, m_1, p_1 \,\} \leq n \leq \min\{\, n_{\max}^{(R)}, n_{\max}^{(L)} \,\}$. *If only exact deflations are performed in the band Lanczos method, then the function* (43)*,* H_n*, is an n-th matrix Padé approximant of the function* (27)*,* H*, and*

$$H_n(s) = H(s) + \mathscr{O}\big((s - s_0)^{j(n)+k(n)}\big).$$

A proof of Theorem 2 is given in [11].

4.3 An Actual Algorithm

The first simple version of the band Lanczos method appeared in the 1995 paper [10]. The algorithm in [10] has no built-in deflation procedure, and the computation of the Lanczos vectors is arranged such that rectangular $n \times (n+m)$ and $n \times (n+p)$ matrices instead of the $n \times n$ matrices $T_n^{(\mathrm{pr})}$ and $\tilde{T}_n^{(\mathrm{pr})}$ are generated. After that, in the joint work [1] with Aliaga, Boley, and Hernández, we developed a complete version of this algorithm that included a proper deflation procedure and a look-ahead strategy to deal with potential breakdowns.

Working on actual code for the band Lanczos method, it became clear that arranging the computations so that square matrices $T_n^{(\mathrm{pr})}$ and $\tilde{T}_n^{(\mathrm{pr})}$ are produced is preferable. A first version of this rearranged band Lanczos method appeared in [12]. The connection (40) of the matrices $T_n^{(\mathrm{pr})}$ and $\tilde{T}_n^{(\mathrm{pr})}$ is exploited to explicitly compute only half of the entries of these matrices. An improved version of this algorithm was included in the survey paper [14] on Krylov-subspace methods for model order reduction. The following algorithm is essentially the version from [14]. The quantities $T_n^{(\mathrm{pr})}$, $\rho_n^{(\mathrm{pr})}$, $\eta_n^{(\mathrm{pr})}$, and Δ_n, which are needed to form the ROM corresponding to (43), are generated as outputs of this algorithm.

Algorithm 3 (Band Lanczos algorithm)

```
For $k = 1, 2, \ldots, m$, set $\hat{v}_k = r_k$.
For $k = 1, 2, \ldots, p$, set $\hat{w}_k = l_k$.
Set $m_c = m$, $p_c = p$, and $\mathscr{I}_v = \mathscr{I}_w = \emptyset$.
For $n = 1, 2, \ldots$, until convergence or $m_c = 0$ or $p_c = 0$ or $\delta_n = 0$ do:

1) Compute $t_{n,n-m_c} = \|\hat{v}_n\|_2$.
   Decide if $\hat{v}_n$ should be deflated. If yes, do the following:

   a) Set $\hat{v}_{n-m_c}^{(\mathrm{dl})} = \hat{v}_n$ and store this deflated vector. Set $\mathscr{I}_v = \mathscr{I}_v \cup \{n - m_c\}$.
   b) Set $m_c = m_c - 1$. If $m_c = 0$, set $n = n - 1$ and stop.
   c) For $k = n, n+1, \ldots, n + m_c - 1$, set $\hat{v}_k = \hat{v}_{k+1}$.
   d) Repeat all of step 1).

2) Compute $\tilde{t}_{n,n-p_c} = \|\hat{w}_n\|_2$.
   Decide if $\hat{w}_n$ should be deflated. If yes, do the following:

   a) Set $\hat{w}_{n-p_c}^{(\mathrm{dl})} = \hat{w}_n$ and store this deflated vector. Set $\mathscr{I}_w = \mathscr{I}_w \cup \{n - p_c\}$.
   b) Set $p_c = p_c - 1$. If $p_c = 0$, set $n = n - 1$ and stop.
   c) For $k = n, n+1, \ldots, n + p_c - 1$, set $\hat{w}_k = \hat{w}_{k+1}$.
   d) Repeat all of step 2).

3) Set $v_n = \hat{v}_n / t_{n,n-m_c}$ and $w_n = \hat{w}_n / \tilde{t}_{n,n-p_c}$.
4) Compute $\delta_n = w_n^T v_n$.
   If $\delta_n = 0$, stop: look-ahead would be needed to continue.
5) For $k = n+1, n+2, \ldots, n + m_c - 1$, do:
       Compute $t_{n,k-m_c} = w_n^T \hat{v}_k / \delta_n$ and set $\hat{v}_k = \hat{v}_k - v_n t_{n,k-m_c}$.
```

```
6) For $k = n+1, n+2, \dots, n+p_{\mathrm{c}}-1$, do:
       Compute $\tilde{t}_{n,k-p_{\mathrm{c}}} = \hat{w}_k^T v_n / \delta_n$ and set $\hat{w}_k = \hat{w}_k - w_n \tilde{t}_{n,k-p_{\mathrm{c}}}$.
7) Compute $\hat{v}_{n+m_{\mathrm{c}}} = M v_n$.
8) a) For $k \in \mathscr{I}_{\mathrm{w}}$ (in ascending order), do:
          Compute $\tilde{\sigma} = \big(\hat{w}_k^{(\mathrm{dl})}\big)^T v_n$ and set $\tilde{t}_{n,k} = \tilde{\sigma}/\delta_n$.
          If $k > 0$, set $t_{k,n} = \tilde{\sigma}/\delta_k$ and $\hat{v}_{n+m_{\mathrm{c}}} = \hat{v}_{n+m_{\mathrm{c}}} - v_k t_{k,n}$.
   b) Set $k_{\mathrm{v}} = \max\{1, n - p_{\mathrm{c}}\}$.
   c) For $k = k_{\mathrm{v}}, k_{\mathrm{v}}+1, \dots, n-1$, do:
          Set $t_{k,n} = \tilde{t}_{n,k}\delta_n/\delta_k$ and $\hat{v}_{n+m_{\mathrm{c}}} = \hat{v}_{n+m_{\mathrm{c}}} - v_k t_{k,n}$.
   d) Compute $t_{n,n} = w_n^T \hat{v}_{n+m_{\mathrm{mc}}}/\delta_n$ and set $\hat{v}_{n+m_{\mathrm{mc}}} = \hat{v}_{n+m_{\mathrm{mc}}} - v_n t_{n,n}$.
9) Compute $\hat{w}_{n+p_{\mathrm{c}}} = M^T w_n$.
10) a) For $k \in \mathscr{I}_{\mathrm{v}}$ (in ascending order), do:
          Compute $\sigma = w_n^T \hat{v}_k^{(\mathrm{dl})}$ and set $t_{n,k} = \sigma/\delta_n$.
          If $k > 0$, set $\tilde{t}_{k,n} = \sigma/\delta_k$ and $\hat{w}_{n+p_{\mathrm{c}}} = \hat{w}_{n+p_{\mathrm{c}}} - w_k \tilde{t}_{k,n}$.
    b) Set $k_{\mathrm{w}} = \max\{1, n - m_{\mathrm{c}}\}$.
    c) For $k = k_{\mathrm{w}}, k_{\mathrm{w}}+1, \dots, n-1$, do:
          Set $\tilde{t}_{k,n} = t_{n,k}\delta_n/\delta_k$ and $\hat{w}_{n+p_{\mathrm{c}}} = \hat{w}_{n+p_{\mathrm{c}}} - w_k \tilde{t}_{k,n}$.
    d) Set $\tilde{t}_{n,n} = t_{n,n}$ and $\hat{w}_{n+p_{\mathrm{c}}} = \hat{w}_{n+p_{\mathrm{c}}} - w_n \tilde{t}_{n,n}$.
11) Set $T_n^{(\mathrm{pr})} = \big[t_{i,k}\big]_{i,k=1,2,\dots,n}$ and $\Delta_n = \mathrm{diag}\big(\delta_1, \delta_2, \dots, \delta_n\big)$ .
    Set $k_\rho = m + \min\{0, n - m_{\mathrm{c}}\}$ and $\rho_n^{(\mathrm{pr})} = \big[t_{i,k-m}\big]_{i=1,2,\dots,n;k=1,2,\dots,k_\rho}$.
    Set $k_\eta = p + \min\{0, n - p_{\mathrm{c}}\}$ and $\eta_n^{(\mathrm{pr})} = \big[\tilde{t}_{i,k-p}\big]_{i=1,2,\dots,n;k=1,2,\dots,k_\eta}$.
12) Check if $n$ is large enough. If yes, stop.
```

5 Structure Preservation

An important class of interconnect models are RCL networks with only independent voltage and current sources. Such models are described by DAEs of the form (4) where $m = p$. Moreover, the equations in (4) can be formulated such that

$$B = L, \quad E = E^T \succeq 0, \quad A + A^T \preceq 0 \tag{44}$$

and the matrices A, E, and B have certain block structures; see, e.g., [17, 19]. Here, the notation "$\succeq$" and "$\preceq$" means that a matrix is *symmetric positive semidefinite* and *symmetric negative semidefinite*, respectively. In this section, we consider the problem of model order reduction of DAEs (4) for this class of RCL networks.

An important property of RCL networks is *passivity*, which means that such networks do not generate energy. In fact, the matrix properties (44) imply passivity. It is desirable and for some applications crucial that ROMs of RCL networks are also passive. One of the disadvantages of Lanczos-based approaches is that the resulting ROMs are not guaranteed to be passive for general RCL networks.

A simple approach to generate passive ROMs is based on *explicit projection* of the matrices in (4). Let V_n be a real $N \times n$ matrix with full column rank n. Setting

$$A_n := V_n^T A V_n, \quad E_n := V_n^T A V_n, \quad B_n := V_n^T B, \quad \text{and} \quad L_n := B_n, \tag{45}$$

one obtains a ROM (6) with matrices that satisfy the same conditions (44) as the matrices of (4). In particular, this ROM is passive.

By combining the projection approach with block Krylov subspaces, the transfer function H_n of the ROM defined by (45) satisfies a *Padé-type approximation* property. To this end, we choose a suitable expansion point $s_0 \geq 0$ and rewrite the transfer function of (4) (with $B = L$) as follows:

$$\begin{gathered} H(s) = B^T \big(sE - A\big)^{-1} B = B^T \big(I + (s - s_0) M\big)^{-1} R, \\ \text{where} \quad M := \big(s_0 E - A\big)^{-1} E \quad \text{and} \quad R := \big(s_0 E - A\big)^{-1} B. \end{gathered} \tag{46}$$

If we choose the matrix V_n such that its range[2] contains the $\hat{n}$-the block Krylov subspace $\mathscr{K}_{\hat{n}}(M,R)$ for some $\hat{n} \leq n$, then the ROM transfer function H_n is an n-th matrix Padé-type approximant of H. As in Theorem 2, $n_{\max}^{(\mathrm{R})}$ is the integer defined in (30) and $j(\hat{n})$ denotes the largest value of j such that $m_1 + m_2 + \cdots + m_j \leq \hat{n}$.

Theorem 4. *Let V_n be an $N \times n$ matrix with full column rank n, and assume that*

$$\mathscr{K}_{\hat{n}}(M,R) \subseteq \operatorname{range}(V_n) \tag{47}$$

for some $m_1 \leq \hat{n} \leq n_{\max}^{(R)}$. Then, the ROM transfer function H_n is an n-th matrix Padé-type approximant of the transfer function (46)*, H, and*

$$H_n(s) = H(s) + \mathscr{O}\big((s - s_0)^{j(\hat{n})}\big). \tag{48}$$

A proof of Theorem 4 is given in [16].

The first ROM algorithm based on explicit projection was PRIMA [29, 30]. It employs a simple block variant of the Arnoldi process without deflation to generate an orthonormal basis for $\mathscr{K}_n(M,R)$, and uses these basis vectors as the columns of the projection matrix V_n. Note that for PRIMA, we have $\hat{n} = n$ and equality of the two subspaces in (47). While the ROMs generated by PRIMA are passive by construction, they do not preserve any of the other properties of RCL networks, such as reciprocity and the block structure of the matrices A, E, and B. We also remark that for a robust implementation of PRIMA, a variant of the Arnoldi process with a proper built-in deflation procedure needs to be used. The band Arnoldi process discussed in Sect. 6 is such a variant.

SRIM [15, 19] was introduced as an improvement of PRIMA that in addition to passivity, preserves both reciprocity and the block structure of the matrices A, E, and B. SPRIM employs the band Arnoldi process to first generate an orthonormal basis for $\mathscr{K}_{\hat{n}}(M,R)$. Let $V_{\hat{n}}$ denote the $N \times \hat{n}$ matrix the columns of which are these basis vectors. Instead of using $V_{\hat{n}}$ as the projection matrix, $V_{\hat{n}}$ is turned into an $N \times n$

[2] The *range*, denoted by range(M), of a matrix M is defined as the subspace spanned by the columns of M.

matrix V_n for some $\hat{n} < n \leq 2\hat{n}$ such that (47) is satisfied and the projected matrices (45) of the SPRIM ROMs preserve both reciprocity and the block structure of A, E,and B; see [17, 19] for details of the construction of V_n. Finally, we remark that the block structure of the SPRIM ROMs imply a higher accuracy than the corresponding PRIMA ROMs. More precisely, as shown in [16], the SPRIM transfer function H_n is an n-th matrix Padé-type approximant of H that matches $2j(\hat{n})$ moments instead of $j(\hat{n})$ moments in (48).

The following example, which is taken from [19], illustrates the higher accuracy of SPRIM. The example is a RCL network with $m = p = 16$ and $N = 1841$ that models the pin package of a chip. The expansion point $s_0 = 2\pi \times 10^{10}$ was used. For this example, $\hat{n} = 128$ was needed for the SPRIM transfer function H_n to converge to the exact transfer function H. Fig. 2 depicts the absolute values of the $(8,1)$-component of the transfer functions. Note that for $\hat{n} = 128$ PRIMA has not converged yet.

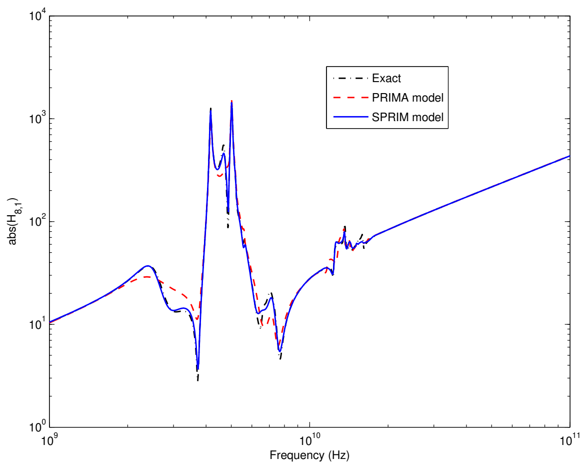


Fig. 2 Package example, $(8,1)$-component of transfer functions

6 Band Arnoldi Process

In this section, we state an algorithm for the band Arnoldi process applied to an $N \times N$ matrix M and an $N \times m$ matrix R. This is essentially the algorithm that first appeared in [14]. Since M and R are complex matrices for some applications of the band Arnoldi process, we use the *complex conjugate transpose* $v^H := \overline{v}^T$ instead of the transpose v^T in the statement of the algorithm.

The notation is similar to the one we used in Sect. 4. The algorithm produces *Arnoldi vectors* and *candidate vectors*

$$v_1.v_2, \ldots, v_n \quad \text{and} \quad v_{n+1}, v_{n+2}, \ldots, v_{n+m_c}, \tag{49}$$

where the Arnoldi vectors are constructed to be orthonormal to each other and the candidate vectors are constructed to be orthogonal to the Arnoldi vectors. Using the notation from (22), the orthogonality of the vectors (49) can be stated compactly as

$$V_n^H V_n = I, \quad \text{and} \quad V_n^H v_{n+j} = 0, \ j = 1, 2, \ldots, m_c, \tag{50}$$

where I denotes the $n \times n$ identity matrix. The recurrences that are employed to generate the vectors (49) can be stated compactly as follows:

$$MV_n = V_n H_n + \begin{bmatrix} 0 & 0 & \cdots & 0 & \hat{v}_{n+1} & \hat{v}_{n+2} & \cdots & \hat{v}_{n+m_c} \end{bmatrix} + V_n^{(\mathrm{dl})}. \tag{51}$$

Similarly to Algorithm 3, the following algorithm produces the matrices

$$H_n^{(\mathrm{pr})} := H_n + V_n^H V_n^{(\mathrm{dl})}, \quad \rho_n^{(\mathrm{pr})} := V_n^H R \tag{52}$$

as outputs. We remark that by multiplying (51) from the left by V_n^H and using (50) and (52), it follows that $H_n^{(\mathrm{pr})} = V_n^H M V_n$. In particular, $H_n^{(\mathrm{pr})}$ and $\rho_n^{(\mathrm{pr})}$ are the orthogonal projections of M and R onto the subspace spanned by the Arnoldi vectors $v_1 . v_2, \ldots, v_n$. If only exact deflations are performed in the algorithm or if no deflations occur, then these vectors span the n-th block Krylov subspace $\mathscr{K}_n(M, R)$.

Algorithm 5 (Band Arnoldi process)

```
For $k = 1, 2, \ldots, m$, set $\hat{v}_k = r_k$.
Set $m_c = m$ and $\mathscr{I} = \emptyset$.
For $n = 1, 2, \ldots,$ until convergence or $m_c = 0$ do:
1) Compute $h_{n,n-m_c} = \|\hat{v}_n\|_2$.
   Decide if $\hat{v}_n$ should be deflated. If yes, do the following:
   a) Set $\hat{v}_{n-m_c}^{(\mathrm{dl})} = \hat{v}_n$ and store this deflated vector. Set $\mathscr{I} = \mathscr{I} \cup \{ n - m_c \}$.
   b) Set $m_c = m_c - 1$. If $m_c = 0$, set $n = n - 1$ and stop.
   c) For $k = n, n+1, \ldots, n + m_c - 1$, set $\hat{v}_k = \hat{v}_{k+1}$.
   d) Repeat all of step 1).
2) Set $v_n = \hat{v}_n / h_{n,n-m_c}$.
3) For $k = n+1, n+2, \ldots, n + m_c - 1$, do:
       Compute $h_{n,k-m_c} = v_n^H \hat{v}_k$ and set $\hat{v}_k = \hat{v}_k - v_n h_{n,k-m_c}$.
4) Compute $\hat{v}_{n+m_c} = M v_n$.
5) For $k = 1, 2, \ldots, n$, do:
       Compute $h_{k,n} = v_k^H \hat{v}_{n+m_c}$ and set $\hat{v}_{n+m_c} = \hat{v}_{n+m_c} - v_k h_{k,n}$.
       Compute $h_{n,k} = v_n^H \hat{v}_k^{(\mathrm{dl})}$ if $k \in \mathscr{I}$, and set $h_{n,k} = 0$ if $k \notin \mathscr{I}$.
6) Set $H_n^{(\mathrm{pr})} = \left[ h_{i,k} \right]_{i,k=1,2,\ldots,n}$.
   Set $k_\rho = m + \min\{ 0, n - m_c \}$ and $\rho_n^{(\mathrm{pr})} = \left[ h_{i,k-m} \right]_{i=1,2,\ldots,n; k=1,2,\ldots,k_\rho}$.
7) Check if $n$ is large enough. If yes, stop.
```

7 Concluding Remarks

In this chapter, we gave an account of how the need to efficiently and accurately simulate the effects of on-chip wiring of integrated circuits has led to the development of new band versions of the Lanczos algorithm and the Arnoldi process for multiple starting vectors. We stress that the applications of these new band Krylov-subspace methods are not restricted to electronic circuit simulation. In fact, they can be employed wherever there is a need for model order reduction of large-scale time-invariant linear dynamical systems with multiple inputs and outputs. Such applications include structural analysis, microelectromechanical systems, transport networks, and computational acoustics. Multiple starting vectors also arise in the context of matrix functions. The use of band Krylov-subspace methods in the efficient evaluation of matrix functions has yet to be explored.

Just as the classical Lanczos algorithm is related to formally orthogonal polynomials (FOPs), there is a connection of the band Lanczos method to matrix-valued FOPs. Some of the underlying relations were derived in [13], but as the recent paper [2] indicates, the connection to matrix-valued FOPs needs to be explored further.

Robust implementations of band Krylov-subspace methods, especially of the band Lanczos method, are not as straightforward as implementations of the classical Krylov-subspace methods. In order to facilitate the use of these band algorithms, the author has produced the software package BANDITS [21], which provides Matlab implementations of various band Krylov-subspace methods.

The band Lanczos method described in this chapter (and implemented in BANDITS) does not include a look-ahead procedure to deal with potential breakdowns. While a version of the band Lanczos method with look-ahead is described in [1], this algorithm was never implemented in an actual code. Since Algorithm 3 is preferable to the one in [1], a version of Algorithm 3 with look-ahead should be developed and implemented in a production-quality code.

References

1. Aliaga, J.I., Boley, D.L., Freund, R.W., Hernández, V.: A Lanczos-type method for multiple starting vectors. Math. Comp. **69**(232), 1577–1601 (2000)
2. Alqahtani, H., Reichel, L.: Multiple orthogonal polynomials applied to matrix function evaluation. BIT Numer. Math. **58**, 835–849 (2018)
3. Arnoldi, W.E.: The principle of minimized iterations in the solution of the matrix eigenvalue problem. Quart. Appl. Math. **9**, 17–29 (1951)
4. Baker, Jr., G.A., Graves-Morris, P.: Padé Approximants, second edn. Cambridge University Press, New York, New York (1996)
5. Cullum, J.K., Donath, W.E.: A block Lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace for large, sparse symmetric matrices. In: Proc. 1974 IEEE Conference on Decision and Control, pp. 505–509. IEEE Press, New York, New York (1974)
6. Cullum, J.K., Willoughby, R.A.: Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Volume 1, Theory. Birkhäuser, Basel, Switzerland (1985)

7. Elmore, W.C.: The transient response of damped linear networks with particular regard to wideband amplifiers. J. Appl. Phys. **19**(1), 55–63 (1948)
8. Feldmann, P., Freund, R.W.: Efficient linear circuit analysis by Padé approximation via the Lanczos process. In: Proceedings of EURO-DAC '94 with EURO-VHDL '94, pp. 170–175. IEEE Computer Society Press, Los Alamitos, California (1994)
9. Feldmann, P., Freund, R.W.: Efficient linear circuit analysis by Padé approximation via the Lanczos process. IEEE Trans. Computer-Aided Design **14**, 639–649 (1995)
10. Feldmann, P., Freund, R.W.: Reduced-order modeling of large linear subcircuits via a block Lanczos algorithm. In: Proc. 32nd ACM/IEEE Design Automation Conference, pp. 474–479. ACM, New York, New York (1995)
11. Freund, R.W.: Computation of matrix Padé approximations of transfer functions via a Lanczos-type process. In: C. Chui, L. Schumaker (eds.) Approximation Theory VIII, Vol. 1: Approximation and Interpolation, pp. 215–222. World Scientific Publishing Co., Inc., Singapore (1995)
12. Freund, R.W.: Band Lanczos method (Section 4.6). In: Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst (eds.) Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, pp. 80–88. SIAM Publications, Philadelphia, Pennsylvania (2000)
13. Freund, R.W.: Computation of matrix-valued formally orthogonal polynomials and applications. J. Comput. Appl. Math. **127**(1–2), 173–199 (2001)
14. Freund, R.W.: Model reduction methods based on Krylov subspaces. Acta Numerica **12**, 267–319 (2003)
15. Freund, R.W.: SPRIM: structure-preserving reduced-order interconnect macromodeling. In: Tech. Dig. 2004 IEEE/ACM International Conference on Computer-Aided Design, pp. 80–87. IEEE Computer Society Press, Los Alamitos, California (2004)
16. Freund, R.W.: On Padé-type model order reduction of J-Hermitian linear dynamical systems. Linear Algebra Appl. **429**, 2451–2464 (2008)
17. Freund, R.W.: Structure-preserving model order reduction of RCL circuit equations. In: W. Schilders, H. van der Vorst, J. Rommes (eds.) Model Order Reduction: Theory, Research Aspects and Applications, Mathematics in Industry, vol. 13, pp. 49–73. Springer-Verlag, Berlin/Heidelberg, Germany (2008)
18. Freund, R.W.: Recent advances in structure-preserving model order reduction. In: P. Li, L. Silveira, P. Feldmann (eds.) Simulation and Verification of Electronic and Biological Systems, chap. 3, pp. 43–70. Springer-Verlag, Dordrecht/Heidelberg/London/New York (2011)
19. Freund, R.W.: The SPRIM algorithm for structure-preserving order reduction of general RCL circuits. In: P. Benner, M. Hinze, E.J.W. ter Maten (eds.) Model Reduction for Circuit Simulation, Lecture Notes in Electrical Engineering, vol. 74, chap. 2, pp. 25–52. Springer-Verlag, Dordrecht/Heidelberg/London/New York (2011)
20. Freund, R.W.: Large-scale matrix computations. In: L. Hogben (ed.) Handbook of Linear Algebra, second edn., chap. 64. Chapman & Hall/CRC, Boca Raton (2013)
21. Freund, R.W.: BANDITS: a Matlab package of band Krylov subspace iterations. Software package (2019). Available online at http://www.math.ucdavis.edu/˜freund/BANDITS/
22. Freund, R.W., Gutknecht, M.H., Nachtigal, N.M.: An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. SIAM J. Sci. Comput. **14**, 137–158 (1993)
23. Golub, G.H., Underwood, R.: The block Lanczos method for computing eigenvalues. In: J.R. Rice (ed.) Mathematical Software III, pp. 361–377. Academic Press, New York, New York (1977)
24. Gragg, W.B.: Matrix interpretations and applications of the continued fraction algorithm. Rocky Mountain J. Math. **4**, 213–225 (1974)
25. Kim, H.M., Craig, Jr., R.R.: Structural dynamics analysis using an unsymmetric block Lanczos algorithm. Internat. J. Numer. Methods Engrg. **26**, 2305–2318 (1988)
26. Kim, H.M., Craig, Jr., R.R.: Computational enhancement of an unsymmetric block Lanczos algorithm. Internat. J. Numer. Methods Engrg. **30**, 1083–1089 (1990)
27. Lanczos, C.: An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. J. Res. Nat. Bur. Standards **45**, 255–282 (1950)

28. Nagel, L.W.: SPICE2: A computer program to simulate semiconductor circuits. Ph.D. thesis, EECS Department, University of California, Berkeley (1975)
29. Odabasioglu, A., Celik, M., Pileggi, L.T.: PRIMA: passive reduced-order interconnect macromodeling algorithm. In: Tech. Dig. 1997 IEEE/ACM International Conference on Computer-Aided Design, pp. 58–65. IEEE Computer Society Press, Los Alamitos, California (1997)
30. Odabasioglu, A., Celik, M., Pileggi, L.T.: PRIMA: passive reduced-order interconnect macromodeling algorithm. IEEE Trans. Computer-Aided Design **17**(8), 645–654 (1998)
31. Pederson, D.O.: A historical review of circuit simulation. IEEE Trans. Circuits Syst. **31**(1), 103–111 (1984)
32. Pillage, L.T., Rohrer, R.A.: Asymptotic waveform evaluation for timing analysis. IEEE Trans. Computer-Aided Design **9**, 352–366 (1990)
33. Pillage, L.T., Rohrer, R.A., Visweswariah, C.: Electronic Circuit and System Simulation Methods. McGraw-Hill, Inc., New York, New York (1995)
34. Raghavan, V., Rohrer, R.A., Pillage, L.T., Lee, J.Y., Bracken, J.E., Alaybeyi, M.M.: AWE–inspired. In: Proc. IEEE Custom Integrated Circuits Conference, pp. 18.1.1–18.1.8. IEEE, Piscataway, New Jersey (1993)
35. Rohrer, R.A.: Circuit simulation — the early years. IEEE Circuits Devices Mag. **8**(3), 32–37 (1992)
36. Ruhe, A.: Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices. Math. Comp. **33**(146), 680–687 (1979)
37. Saad, Y.: Iterative Methods for Sparse Linear Systems, second edn. SIAM Publications, Philadelphia, Pennsylvania (2003)
38. Schilders, W., van der Vorst, H., Rommes, J. (eds.): Model Order Reduction: Theory, Research Aspects and Applications, Mathematics in Industry, vol. 13. Springer-Verlag, Berlin/Heidelberg, Germany (2008)
39. Underwood, R.: An iterative block Lanczos method for the solution of large sparse symmetric eigenproblems. Ph.D. thesis, Computer Science Department, Stanford University, Stanford, California (1975)