Reduced-Order Modeling Techniques Based on Krylov Subspaces and Their Use in Circuit Simulation*

Roland W. Freund
Bell Laboratories
Lucent Technologies
Room 2C–420
700 Mountain Avenue
Murray Hill, NJ 07974–0636

Abstract

In recent years, reduced-order modeling techniques based on Krylov-subspace iterations, especially the Lanczos algorithm and the Arnoldi process, have become popular tools to tackle the large-scale time-invariant linear dynamical systems that arise in the simulation of electronic circuits. This paper reviews the main ideas of reduced-order modeling techniques based on Krylov subspaces and describes the use of reduced-order modeling in circuit simulation.

1 Introduction

Krylov-subspace methods, most notably the Lanczos algorithm [81, 82] and the Arnoldi process [5], have long been recognized as powerful tools for large-scale matrix computations. Matrices that occur in large-scale computations usually have some special structures that allow to compute matrix-vector products with such a matrix (or its transpose) much more efficiently than for a dense, unstructured matrix. The most common structure is sparsity, i.e., only few of the matrix entries are nonzero. Computing a matrix-vector product with a sparse $N \times N$ matrix with N_z nonzero entries requires only $\mathcal{O}(N_z)$ arithmetic operations, instead of $\mathcal{O}(N^2)$ for a general dense matrix. Examples for less obvious structures are matrices that are given as a product of sparse matrices and "easily invertible" matrices (such as sparse triangular matrices), and Toeplitz matrices, i.e., matrices whose entries are constant along each diagonal. Note that general Toeplitz matrices are dense. Nevertheless, by embedding an $N \times N$ Toeplitz matrix into a $2N \times 2N$ circulant matrix and using FFTs, multiplication of the Toeplitz matrix with any vector of length N can be computed with only $\mathcal{O}(N \log N)$ arithmetic operations; see, e.g., [20]. Krylov-subspace methods involve the matrix only in the form of matrix-vector products with the matrix or its transpose. Thus they naturally exploit the special matrix structures that occur in large-scale computations.

*Numerical Analysis Manuscript No. 98–3–02, Bell Laboratories, Murray Hill, New Jersey, February 1998. Available on WWW at `http://cm.bell-labs.com/cs/doc/98`.

Krylov-subspace methods have become standard tools and today are widely used for the iterative solution of large systems of linear equations and for large-scale eigenvalue computations. For surveys and further references, we refer the reader to [20, 53, 106] (for the solution of linear systems) and [28, 30, 31, 32, 105, 117] (for eigenvalue computations). However, somewhat surprisingly, it is only fairly recently that Krylov-subspace methods are being employed for reduced-order modeling of large-scale linear dynamical systems. This is surprising for two reasons. First, some of the theory relating Krylov-subspace methods and reduced-order modeling has been known for quite some time; see, e.g., [66, 67, 120], as well as the more recent survey paper [12]. In particular, Gragg's 1974 paper [66] already contains the connection between the Lanczos process and Padé approximation of transfer functions of single-input single-output systems. Second, within the last 10 years, practical procedures for reduced-order modeling based on Krylov subspaces have been described in a number of papers; see [26, 27, 78, 113, 114].

The recent surge of interest in Krylov-subspace methods for reduced-order modeling was triggered by the need of such techniques in the simulation of integrated electronic circuits. Today's integrated circuits are extremely complex, with up to hundreds of thousands or even millions of devices, and prototyping of such circuits is no longer possible. Instead, computational methods are used to simulate and analyze the behavior of the electronic circuit at the design stage; see, e.g., [36, 121]. This allows to correct the design before the circuit is actually fabricated in silicon. Today's trends in VLSI technology have brought to light a number of challenging simulation problems for which traditional circuit-simulation techniques are no longer adequate. In particular, due to ever-increasing clock rates and ever-decreasing feature sizes, the accurate simulation of the wires that connect the functional blocks of a circuit, the so-called *interconnect*, has become a crucial issue; see, e.g., [36, 92, 116] and the references given therein. At the chip level, interconnect can be modeled by large, lumped, linear networks that are generated by automatic parasitics-extraction programs. Typically, these networks are too large to be included directly in the simulation of the complete circuit. Instead, the large networks are replaced by much smaller reduced-order models that are then integrated into the general design and simulation methodology. Krylov-subspace techniques are emerging as the methods of choice for tackling the reduced-order modeling problems in circuit simulation. For example, in 1997, among the papers presented at DAC and ICCAD—two of the major circuit simulation conferences—the papers [37, 38, 44, 76, 86, 89] all deal with various aspects of Krylov-subspace methods for reduced-order modeling. Furthermore, the first commercial simulation tools based on Krylov-subspace methods are becoming available. Finally, we remark that the potential and the efficiency of Padé approximation for circuit-simulation problems was first recognized by Huang, Pillage, and Rohrer [69, 93], who introduced the **AWE** (**A**symptotic **W**aveform **E**valuation) method for the simulation of large linear, or linearized, electronic circuits. Chan [19] pointed out the connection of AWE with reduced-order modeling in control theory.

This paper reviews the main ideas of reduced-order modeling techniques based on Krylov subspaces and describes the use of reduced-order modeling in circuit simulation. The remainder of the paper is organized as follows. In Section 2, we formulate the reduced-order modeling problem for linear dynamical systems. In Section 3, we describe briefly how linear dynamical systems arise in circuit simulation. In Section 4, we introduce the notions of Krylov and block Krylov subspaces. In Section 5, we describe the classical Lanczos process and some of its extensions to multiple starting vectors. In Section 6, we discuss the use of the Lanczos process and of Lanczos-type methods for reduced-order modeling. In Section 7, we describe the Arnoldi process. In Section 8, we survey reduced-order modeling techniques that are based on the Arnoldi process. In Section 9, we discuss reduced-order modeling of noise-type transfer functions, which arise in circuit-noise computations. In Section 10, we offer some concluding remarks.

Throughout this article, all vectors and matrices are allowed to have real or complex entries. We use boldface letters to denote vectors and matrices. As usual, $\overline{\mathbf{M}} = \left[\overline{m_{jk}}\right]$, $\mathbf{M}^{\mathrm{T}} = \left[m_{kj}\right]$, and $\mathbf{M}^{\mathrm{H}} = \overline{\mathbf{M}}^{\mathrm{T}} = \left[\overline{m_{kj}}\right]$ denote the complex conjugate, transpose, and the conjugate transpose, respectively, of the matrix $\mathbf{M} = \left[m_{jk}\right]$. The vector norm $\|\mathbf{x}\| := \sqrt{\mathbf{x}^{\mathrm{H}}\mathbf{x}}$ is always the Euclidean norm. The sets of real and complex numbers are denoted by $\mathbb{R}$ and $\mathbb{C}$, respectively. We denote by $\mathbf{I}_{k\times k}$ the $k \times k$ identity matrix and by $\mathbf{0}_{k\times l}$ the $k \times l$ zero matrix. We set $\mathbf{0}_k = \mathbf{0}_{k\times 1}$ for the zero (column) vector of length k. If the actual dimensions of the identity matrix, zero matrix, or zero vector are apparent from the context, then we drop the indices and simply write $\mathbf{I}$ and $\mathbf{0}$. Finally, $\mathbf{M} \geq \mathbf{0}$ means that the matrix $\mathbf{M}$ is Hermitian positive semi-definite.

2 Reduced-Order Modeling of Linear Dynamical Systems

In this section, we formulate the reduced-order modeling problem for linear dynamical systems.

2.1 Linear Dynamical Systems

We consider *time-invariant multi-input multi-output linear dynamical systems* of the form

$$\begin{aligned} \mathbf{C}\,\frac{d\mathbf{x}}{dt} &= -\mathbf{G}\,\mathbf{x} + \mathbf{B}\,\mathbf{u}(t), \\ \mathbf{y}(t) &= \mathbf{E}^{\mathrm{T}}\,\mathbf{x}(t). \end{aligned} \tag{1}$$

Here, $\mathbf{C},\ \mathbf{G} \in \mathbb{C}^{N\times N}$, $\mathbf{B} \in \mathbb{C}^{N\times m}$, and $\mathbf{E} \in \mathbb{C}^{N\times p}$ are given matrices, m and p denote the number of inputs and outputs, respectively, the components of the given vector-valued function $\mathbf{u}\colon [0,\infty) \mapsto \mathbb{C}^m$ are the inputs, and $\mathbf{y}\colon [0,\infty) \mapsto \mathbb{C}^p$ is the unknown function of outputs. The components of the unknown vector-valued function $\mathbf{x}\colon [0,\infty) \mapsto \mathbb{C}^N$ are the state variables, and N is the state-space dimension. In general, the matrices $\mathbf{C}$ and $\mathbf{G}$ in (1) are allowed to be singular. However, we assume that $\mathbf{G} + s\,\mathbf{C}$ is a *regular pencil*, i.e., there exist points $s_0 \in \mathbb{C}$ such that the matrix $\mathbf{G} + s_0\,\mathbf{C}$ is nonsingular. We remark that, for the linear dynamical systems arising in circuit simulation, the matrices $\mathbf{C}$ and $\mathbf{G}$ are often singular, but the pencil $\mathbf{G} + s\,\mathbf{C}$ is always regular. Note that for nonsingular $\mathbf{C}$, the first equation in (1) is a first-order system of linear differential equations; in principle, by multiplying from the left by $\mathbf{C}^{-1}$, this case could always be reduced to the case $\mathbf{C} = \mathbf{I}$. If $\mathbf{C}$ is singular, then the first equation in (1) represents a first-order system of linear *differential-algebraic equations* (DAEs).

Linear dynamical systems of the form (1) have been studied extensively in control theory, especially for the case $\mathbf{C} = \mathbf{I}$, see, e.g., [24, 73]. More recently, there has been quite some interest in linear systems (1) with singular matrices $\mathbf{C}$, so-called descriptor systems or singular systems; see, e.g., [33, 61, 83, 118] and references given there. However, the techniques developed in control theory are mostly designed for systems (1) of small or moderate state-space dimension N, and their use becomes prohibitive in the large-scale case. We remark that, in circuit simulation, systems (1) with state-space dimensions as large as $N = \mathcal{O}(10^6)$ do arise. Fortunately, the matrices $\mathbf{G}$ and $\mathbf{C}$ are sparse, and by exploiting sparsity, it is possible to tackle such large-scale systems (1).

For large-scale systems (1), the goal usually is to obtain the input-output mapping $\mathbf{u}(t) \mapsto \mathbf{y}(t)$, rather than computing all the components of the state-space vector $\mathbf{x}(t)$. In frequency domain, this mapping is described by the so-called *transfer function* of (1). For vector-valued functions $\mathbf{g}(t)$, $t \in [0, \infty)$, with $\mathbf{g}(0) = \mathbf{0}$, we denote by

$$\hat{\mathbf{g}}(s) = \int_0^\infty \mathbf{g}(t) e^{-st}\, dt \tag{2}$$

the (frequency-domain) *Laplace transform* of $\mathbf{g}$. Here, the variable s is in general complex; physically meaningful are the purely imaginary values $s = \mathrm{j}\,\omega$, $\omega \geq 0$, that correspond to the frequency ω.

We now assume, for simplicity, zero initial conditions $\mathbf{x}(0) = \mathbf{0}$ and $\mathbf{u}(0) = \mathbf{0}$ in (1). By applying (2) to the linear system (1), we obtain the following frequency-domain formulation:

$$\begin{aligned} s\,\mathbf{C}\,\hat{\mathbf{x}} &= -\mathbf{G}\,\hat{\mathbf{x}} + \mathbf{B}\,\hat{\mathbf{u}}(s), \\ \hat{\mathbf{y}}(s) &= \mathbf{E}^{\mathrm{T}}\,\hat{\mathbf{x}}(s). \end{aligned} \tag{3}$$

Eliminating $\hat{\mathbf{x}}$ in (3) results in the desired frequency-domain input-output relation $\hat{\mathbf{y}}(s) = \mathbf{H}(s)\,\hat{\mathbf{u}}(s)$, where $\mathbf{H}$, the transfer function of (1), is given by

$$\mathbf{H}(s) = \mathbf{E}^{\mathrm{T}}\,(\mathbf{G} + s\,\mathbf{C})^{-1}\,\mathbf{B}. \tag{4}$$

Note that $\mathbf{H}\colon \mathbb{C} \mapsto (\mathbb{C} \cup \{\infty\})^{p\times m}$ is a matrix-valued rational function.

2.2 Reduced-Order Modeling

The basic idea of reduced-order modeling of dynamical systems is to replace the original system by a system of the same type, but with much smaller state-space dimension. A *reduced-order model* of a given linear dynamical system (1) is thus a linear dynamical system of the form

$$\begin{aligned} \mathbf{C}_n \frac{d\mathbf{z}}{dt} &= -\mathbf{G}_n\,\mathbf{z} + \mathbf{B}_n\,\mathbf{u}(t), \\ \mathbf{y}(t) &= \mathbf{E}_n^{\mathrm{T}}\,\mathbf{z}(t), \end{aligned} \tag{5}$$

where

$$\mathbf{C}_n,\ \mathbf{G}_n \in \mathbb{C}^{n\times n}, \quad \mathbf{B}_n \in \mathbb{C}^{n\times m}, \quad \mathbf{E}_n \in \mathbb{C}^{n\times p}, \tag{6}$$

and the state-space dimension n of (5) is much smaller than the dimension N of (1). Of course the goal is to choose the matrices (6) so that the reduced-order model (5) represents a meaningful approximation of the original system (1). For a description of standard reduced-order modeling techniques, we refer the reader to [18, 45, 62, 85, 120] for a description of various reduced-order modeling techniques. However, most of these techniques, such as balanced realizations [85] and Hankel-norm optimal approximations [62], do not exploit the underlying structures of and are therefore not applicable to large-scale dynamical systems.

In analogy to (4), the transfer function $\mathbf{H}_n \colon \mathbb{C} \mapsto (\mathbb{C} \cup \{\infty\})^{p \times m}$ of the reduced-order model (5) is given by

$$\mathbf{H}_n(s) = \mathbf{E}_n^{\mathrm{T}} \left(\mathbf{G}_n + s\,\mathbf{C}_n\right)^{-1} \mathbf{B}_n. \tag{7}$$

Constructing a suitable reduced-order model (5) is equivalent to finding matrices (6) such that the reduced-order transfer function (7), $\mathbf{H}_n$, in some sense approximates the transfer function (4), $\mathbf{H}$, of the original system (1).

It turns out that most Krylov-subspace techniques for reduced-order modeling correspond to choosing $\mathbf{H}_n$ as a matrix-Padé or matrix-Padé-type approximant of $\mathbf{H}$. Let $s_0 \in \mathbb{C}$ be any expansion point such that the matrix $\mathbf{G} + s_0\,\mathbf{C}$ is nonsingular. Recall that $\mathbf{G} + s\,\mathbf{C}$ is assumed to be a regular pencil; this implies that $\mathbf{G} + s_0\,\mathbf{C}$ is singular only for at most N points s_0 in the complex plane. Expanding $\mathbf{H}$ about s_0 gives

$$\mathbf{H}(s) = \sum_{i=0}^{\infty} \mathbf{M}_i \,(s - s_0)^i, \quad \text{where} \quad \mathbf{M}_i \in \mathbb{C}^{p \times m} \quad \text{for all} \quad i. \tag{8}$$

The coefficients $\mathbf{M}_i$ are called the *moments* or *Markov parameters* (corresponding to the point s_0) of $\mathbf{H}$. A function $\mathbf{H}_n$ of the form (7) with matrices (6) is said to be an *n-th matrix-Padé approximant* (corresponding to the point s_0) of $\mathbf{H}$ if its expansion about s_0 matches the expansion (8) of $\mathbf{H}$ in as many leading coefficients as possible, i.e., if

$$\begin{aligned} \mathbf{H}_n(s) &= \mathbf{H}(s) + \mathcal{O}\left((s - s_0)^{q(n)}\right) \\ &= \sum_{i=0}^{q(n)-1} \mathbf{M}_i \,(s - s_0)^i + \mathcal{O}\left((s - s_0)^{q(n)}\right), \end{aligned} \tag{9}$$

where $q(n)$ is as large as possible. We will also consider reduced-order modeling based on functions $\mathbf{H}_n$ for which $q(n)$ in (9) is not maximal; in this case, $\mathbf{H}_n$ is called a *matrix-Padé-type approximant*. We refer the reader to [11] for an overview of Padé and Padé-type approximants. Finally, note that (7) is not the usual representation of matrix-Padé approximants in terms of numerator and denominator matrix polynomials. The reason for using (7) instead of the polynomial representation is that the latter is inherently numerically unstable; this will be discussed further in subsection 4.3 below.

2.3 Reduction to One Matrix

Recall that the linear dynamical system (1) involves two square matrices $\mathbf{C}$ and $\mathbf{G}$. Krylov-subspace techniques, however, can only be applied to one matrix. In order to generate reduced-order models by means of Krylov-subspace techniques, one thus has to first reduce $\mathbf{C}$ and $\mathbf{G}$ to one matrix. Let $s_0 \in \mathbb{C}$ be the expansion point used in (8) and (9), and let

$$\mathbf{G} + s_0\,\mathbf{C} = \mathbf{F}_1 \cdot \mathbf{F}_2, \quad \text{where} \quad \mathbf{F}_1,\ \mathbf{F}_2 \in \mathbb{C}^{N \times N}, \tag{10}$$

be any factorization of the matrix $\mathbf{G}+s_0\,\mathbf{C}$ into two square matrices. We stress that, in (10), no assumptions on $\mathbf{F}_1$ and $\mathbf{F}_2$ are made; in particular, the trivial factorizations with $\mathbf{F}_1 = \mathbf{I}$, respectively $\mathbf{F}_2 = \mathbf{I}$, are allowed. Recall that $\mathbf{G} + s_0\,\mathbf{C}$ is assumed to be nonsingular, and hence $\mathbf{F}_1$ and $\mathbf{F}_2$ are also nonsingular. Using (10), we can rewrite the transfer function (4) with only one square matrix, $\mathbf{A}$, as follows:

$$\begin{aligned}\mathbf{H}(s) &= \mathbf{L}^{\mathrm{T}}\left(\mathbf{I} - (s - s_0)\,\mathbf{A}\right)^{-1}\mathbf{R},\\ \text{where}\quad \mathbf{A} &:= -\mathbf{F}_1^{-1}\,\mathbf{C}\,\mathbf{F}_2^{-1}, \quad \mathbf{R} := \mathbf{F}_1^{-1}\,\mathbf{B}, \quad \mathbf{L} := \mathbf{F}_2^{-\mathrm{T}}\,\mathbf{E}.\end{aligned} \tag{11}$$

Expanding $\mathbf{H}$ in (11) about s_0, we get

$$\mathbf{H}(s) = \mathbf{L}^{\mathrm{T}}\left(\sum_{i=0}^{\infty}\mathbf{A}^i\,(s - s_0)^i\right)\mathbf{R} = \sum_{i=0}^{\infty}\mathbf{L}^{\mathrm{T}}\,\mathbf{A}^i\,\mathbf{R}\;(s - s_0)^i.$$

By comparison with (8), it follows that the moments of $\mathbf{H}$ are given by

$$\mathbf{M}_i = \mathbf{L}^{\mathrm{T}}\,\mathbf{A}^i\,\mathbf{R}, \quad i \geq 0. \tag{12}$$

Krylov-subspace methods use the matrix $\mathbf{A}$ in the form of matrix-vector products with $\mathbf{A}$, and possibly with $\mathbf{A}^{\mathrm{T}}$. Typically, in the large-scale case, the matrices $\mathbf{C}$ and $\mathbf{G}$, and also the factors $\mathbf{F}_1$ and $\mathbf{F}_2$ in (10) are sparse. In particular, this is the case for the linear dynamical systems in circuit simulation. However, the matrix $\mathbf{A}$ in (11) is, in general, dense even though $\mathbf{C}$, $\mathbf{F}_1$, and $\mathbf{F}_2$ are sparse. Nevertheless, by exploiting the factorization $\mathbf{A} = -\mathbf{F}_1^{-1}\,\mathbf{C}\,\mathbf{F}_2^{-1}$, matrix-vector products $\mathbf{p} = \mathbf{A}\cdot\mathbf{v}$ can still be computed efficiently as follows:

$$\begin{aligned}&\text{Solve}\quad \mathbf{F}_2\,\mathbf{t}_1 = -\mathbf{v} \quad \text{to obtain}\quad \mathbf{t}_1;\\ &\text{Set}\quad \mathbf{t}_2 = \mathbf{C}\cdot\mathbf{t}_1;\\ &\text{Solve}\quad \mathbf{F}_1\,\mathbf{p} = \mathbf{t}_2 \quad \text{to obtain}\quad \mathbf{p}.\end{aligned}$$

Similarly, each matrix-vector product with $\mathbf{A}^{\mathrm{T}}$ can be obtained by means of one solve with $\mathbf{F}_1^{\mathrm{T}}$, one multiplication with $\mathbf{C}^{\mathrm{T}}$, and one solve with $\mathbf{F}_2^{\mathrm{T}}$.

3 Linear Systems in Circuit Simulation

In this section, we describe briefly how linear dynamical systems (1) arise in circuit simulation. For introductions to circuit simulation and overviews of typical simulation tasks, we refer the reader to [42, 99, 100, 108, 121].

3.1 General Circuit Equations

Circuits are usually modeled as networks whose branches correspond to the circuit elements and whose nodes correspond to the interconnections of the circuit elements. Such networks are characterized by three types of equations: *Kirchhoff's current law* (KCLs), *Kirchhoff's voltage law* (KVLs), and the *branch constitutive relations* (BCRs). The unknowns in these equations are the currents through the branches of the network, the voltage drops along the branches, and the voltages at the nodes of the network. The KCLs and KVLs are linear algebraic equations that only depend on the topology of the circuit. The KCLs state that, at each node $\mathcal{N}$ of the network, the currents flowing in and out of $\mathcal{N}$ need to sum up to zero. The KVLs state that, for each closed loop $\mathcal{L}$ of the network, the voltage drops along $\mathcal{L}$ sum up to zero. The BCRs are equations that characterize the actual 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. Furthermore, in general, the BCRs involve time-derivatives of the unknowns, and thus they are ordinary differential equations.

The KCLs, KVLs, and BCRs can be summarized as a system of first-order, in general nonlinear, DAEs,

$$\mathbf{f}(\hat{\mathbf{x}}, t) + \frac{d}{dt}\mathbf{q}(\hat{\mathbf{x}}, t) = \mathbf{0}, \tag{13}$$

together with suitable initial conditions. Here, $\hat{\mathbf{x}} = \hat{\mathbf{x}}(t)$ is the unknown vector of circuit variables at time t, the vector-valued function $\mathbf{f}(\hat{\mathbf{x}}, t)$ represents the contributions of non-reactive elements such as resistors, sources, etc., and the vector-valued function $\frac{d}{dt}\mathbf{q}(\hat{\mathbf{x}}, t)$ represents the contributions of reactive elements such as capacitors and inductors. There are a number of established methods, such as sparse tableau, nodal formulation, modified nodal analysis, etc. [121], for writing down equation (13), and the vector functions $\hat{\mathbf{x}}$, $\mathbf{f}$, $\mathbf{q}$, as well as their dimension, $\hat{N}$, depend on the chosen formulation method. The most general method is sparse tableau, which consists of just listing all the KCLs, KVLs, and BCRs. The other formulation methods can be interpreted as starting from sparse tableau and eliminating some of the unknowns by using some of the KCL or KVL equations. For all the standard formulation methods, the dimension $\hat{N}$ is of the order of the number of devices in the circuit.

3.2 Linear Subcircuits and Linearized Circuits

Traditional SPICE-like circuit simulators are based on the numerical solution of the system of DAEs (13). However, for very large circuits, such a time-domain integration of (13) is often inefficient or even prohibitive. Instead, one replaces (13) by an approximate system of much smaller dimension that can then be solved numerically by time-domain integration. Such approximate systems can be obtained by employing reduced-order modeling techniques to a suitable dynamical linear system (1) that is derived from the original nonlinear system (13). Next, we sketch two typical constructions for obtaining a linear system (1) from (13).

Often, circuits contain large subnetworks of linear circuit elements. Let $\mathcal{C}_l$ be such a large linear subnetwork of a given circuit, and denote by $\mathcal{C}_r$ the, in general nonlinear, remainder of the circuit. Then, after a suitable reordering, the vector $\hat{\mathbf{x}}$ of circuit variables in (13) can be partitioned as follows:

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{x}}_r \\ \mathbf{y} \\ \hat{\mathbf{x}}_l \end{bmatrix}. \tag{14}$$

Here, $\hat{\mathbf{x}}_l$ and $\hat{\mathbf{x}}_r$ denote the circuit variables exclusive to $\mathcal{C}_l$ and $\mathcal{C}_r$, respectively, and $\mathbf{y}$ represents the variables shared by $\mathcal{C}_l$ and $\mathcal{C}_r$. Using the partitioning (14) and setting

$$\mathbf{x} := \begin{bmatrix} \mathbf{y} \\ \hat{\mathbf{x}}_l \end{bmatrix} \quad \text{and} \quad \mathbf{x}_0 := \begin{bmatrix} \hat{\mathbf{x}}_r \\ \mathbf{y} \end{bmatrix}, \tag{15}$$

the functions $\mathbf{f}$ and $\mathbf{q}$ in (13), after a suitable reordering of the equations in (13), can be expressed as follows:

$$\mathbf{f}(\hat{\mathbf{x}}, t) = \begin{bmatrix} \mathbf{f}_0(\mathbf{x}_0, t) \\ \mathbf{0}_k \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{G} \cdot \mathbf{x} \end{bmatrix}, \quad \mathbf{q}(\hat{\mathbf{x}}, t) = \begin{bmatrix} \mathbf{q}_0(\mathbf{x}_0, t) \\ \mathbf{0}_k \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{C} \cdot \mathbf{x} \end{bmatrix}. \tag{16}$$

Here, $\mathbf{q}_0$ and $\mathbf{f}_0$ represent the contributions of resistive and reactive elements from the subcircuit $\mathcal{C}_r$, and the matrices $\mathbf{C}$ and $\mathbf{G}$ represent, respectively, the contributions of resistive and reactive elements in the linear subcircuit $\mathcal{C}_l$. In (16), without loss of generality, we have assumed that the vector-valued functions $\mathbf{f}_0$ and $\mathbf{q}_0$ have the same number of components, and that the matrices $\mathbf{C}$ and $\mathbf{G}$ are square and of the same size, say $N \times N$; this can always be achieved by padding $\mathbf{f}_0$, $\mathbf{q}_0$, $\mathbf{C}$, and $\mathbf{G}$ with additional zeros, if necessary. Unless the subcircuit $\mathcal{C}_l$ is completely decoupled from the remainder circuit $\mathcal{C}_r$, we have $m := N - k > 0$ in (16). This means that, in (16), the last m components of the, in general nonlinear, functions $\mathbf{f}_0$ and $\mathbf{q}_0$ are connected with the first m components of the linear functions $\mathbf{G} \cdot \mathbf{x}$ and $\mathbf{C} \cdot \mathbf{x}$. By introducing an additional m-dimensional vector, $\mathbf{u} = \mathbf{u}(t)$, of circuit variables, these m connecting equations can be decoupled. Indeed, with (16), one readily verifies that the original system (13) is equivalent to the following system:

$$\mathbf{f}_0(\mathbf{x}_0, t) + \frac{d}{dt}\mathbf{q}_0(\mathbf{x}_0, t) + \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_m \end{bmatrix} \mathbf{u} = 0, \tag{17}$$

$$\mathbf{C}\,\frac{d\mathbf{x}}{dt} + \mathbf{G}\,\mathbf{x} = \begin{bmatrix} \mathbf{I}_m \\ \mathbf{0} \end{bmatrix} \mathbf{u}. \tag{18}$$

The additional variables $\mathbf{u}$ represent interface signals between $\mathcal{C}_l$ and $\mathcal{C}_r$. In particular, in the case that the system (13) corresponds to a nodal formulation of the circuit, the variables $\mathbf{u}$ represent the currents in the wires that connect $\mathcal{C}_l$ and $\mathcal{C}_r$, and the variables $\mathbf{y}$ in (15) represent the corresponding voltages. We now set

$$\mathbf{B} := \begin{bmatrix} \mathbf{I}_m \\ \mathbf{0}_{(N-m)\times m} \end{bmatrix} \quad \text{and} \quad \mathbf{E} := \begin{bmatrix} \mathbf{I}_p \\ \mathbf{0}_{(N-p)\times p} \end{bmatrix},$$

where p denotes the length of the vector $\mathbf{y}$ in (14). In view of (15), the matrix $\mathbf{E}^{\mathrm{T}}$ just selects the subvector $\mathbf{y}$ from $\mathbf{x}$, i.e.,

$$\mathbf{y} = \mathbf{E}^{\mathrm{T}}\,\mathbf{x}. \tag{19}$$

Equations (18) and (19) constitute a linear dynamical system of the form (1). We can thus employ reduced-order modeling techniques to replace (18) and (19) by a system with much smaller state-space dimension. By combining such a suitable reduced-order model for the linear part (18) with the nonlinear part (17), one obtains a system of DAEs that represents an approximation to the original system (13); see, e.g., [41, 94]. Provided that the size of $\mathcal{C}_l$ dominates that of $\mathcal{C}_r$, the approximate system has a much smaller state-space dimension than (13), and can thus be integrated by SPICE-type circuit simulators.

A second example of linear dynamical systems (1) arising from the original system (13) is *small-signal analysis*; see, e.g., [50]. We assume that all time-varying circuit elements are independent sources, and we denote by m the number of such sources. In this case, the system (13) simplifies to a system of the form

$$\mathbf{f}(\hat{\mathbf{x}}) + \frac{d}{dt}\mathbf{q}(\hat{\mathbf{x}}) = \mathbf{B}\,\mathbf{u}(t). \tag{20}$$

Here, $\mathbf{u}(t)$ is an m-dimensional vector the entries of which are the excitations applied to the sources, and $\mathbf{B}$ is a constant matrix with m columns. A constant vector $\hat{\mathbf{x}}_0$ is called a *DC operating point* of the circuit if it is a solution of (20) for zero excitation $\mathbf{u}(t) = \mathbf{0}$, i.e., if $\hat{\mathbf{x}}_0$ is a solution of the system of algebraic equations

$$\mathbf{f}(\hat{\mathbf{x}}_0) = \mathbf{0}. \tag{21}$$

Let $\hat{\mathbf{x}}_0$ be a DC operating point of the circuit. Then, by replacing $\mathbf{f}(\hat{\mathbf{x}})$ and $\mathbf{q}(\hat{\mathbf{x}})$ in (20) with their linearizations about $\hat{\mathbf{x}}_0$ and introducing the new variable $\mathbf{x} := \hat{\mathbf{x}} - \hat{\mathbf{x}}_0$, we obtain from (20) the linearized circuit equations

$$\mathbf{f}(\hat{\mathbf{x}}_0) + \mathbf{G}\,\mathbf{x} + \frac{d}{dt}\left(\mathbf{q}(\hat{\mathbf{x}}_0) + \mathbf{C}\,\mathbf{x}\right) = \mathbf{B}\,\mathbf{u}(t). \tag{22}$$

Here,

$$\mathbf{G} := D\,\mathbf{f}(\hat{\mathbf{x}}_0) \quad \text{and} \quad \mathbf{C} := D\,\mathbf{q}(\hat{\mathbf{x}}_0) \tag{23}$$

denote the Jacobian matrices of $\mathbf{f}$ and $\mathbf{g}$, respectively, at the point $\mathbf{x}_0$. Since $\hat{\mathbf{x}}_0$ is a constant vector, we trivially have $\frac{d}{dt}\mathbf{q}(\hat{\mathbf{x}}_0) = \mathbf{0}$. Together with (21), it follows that (22) reduces to

$$\mathbf{G}\,\mathbf{x} + \mathbf{C}\,\frac{d\mathbf{x}}{dt} = \mathbf{B}\,\mathbf{u}(t). \tag{24}$$

Let $\mathbf{E}$ be a constant matrix that selects from $\mathbf{x}$ the vector $\mathbf{y}$ of entries and, possibly, combinations of entries, of interest. Equation (24) together with the equation $\mathbf{y} = \mathbf{E}^{\mathrm{T}}\,\mathbf{x}$ again constitute a linear dynamical system (1). Finally, let us consider the important special case that all circuit elements are linear. In this case, $\mathbf{f}(\hat{\mathbf{x}}) = \mathbf{G} \cdot \hat{\mathbf{x}}$ and $\mathbf{q}(\hat{\mathbf{x}}) = \mathbf{C} \cdot \hat{\mathbf{x}}$ are linear functions. Moreover, the zero vector $\hat{\mathbf{x}}_0 = \mathbf{0}$ is a DC operating point. With this choice of $\hat{\mathbf{x}}_0$, we have $\mathbf{x} = \hat{\mathbf{x}}$, and the system of linearized circuit equations (24) is identical to the original system (20). Thus circuits with only linear elements are examples of linear dynamical systems of the form (1).

Recall from (10) that the application of Krylov-subspace techniques requires a factorization of the matrix $\mathbf{G} + s_0\,\mathbf{C}$. In circuit simulation, as we have just described, the matrices

$\mathbf{G}$ and $\mathbf{C}$ are defined by either (23) or (16), i.e., they are Jacobians of the circuit functions $\mathbf{f}$ and $\mathbf{q}$ from (13), or they are submatrices of the Jacobians. In both cases, $\mathbf{G}$ and $\mathbf{C}$ are sparse. Furthermore, they are such that, typically, sparse Gaussian elimination can be employed to compute an LU factorization of $\mathbf{G} + s_0\,\mathbf{C}$ with very little fill-in. More precisely, sparse Gaussian elimination produces a decomposition

$$\mathbf{G} + s_0\,\mathbf{C} = \mathbf{P}_1\,\mathbf{L}_0\,\mathbf{U}_0\,\mathbf{P}_2, \tag{25}$$

where $\mathbf{P}_1$ and $\mathbf{P}_2$ are permutation matrices, $\mathbf{L}_0$ is a lower triangular matrix, and $\mathbf{U}_0$ is an upper triangular matrix. Setting $\mathbf{F}_1 = \mathbf{P}_1\,\mathbf{L}_0$ and $\mathbf{F}_2 = \mathbf{U}_0\,\mathbf{P}_2$, we then use (25) as the factorization (10). The matrices $\mathbf{P}_1$, $\mathbf{P}_2$ in (25) are used to pivot for sparsity and numerical stability. Pivoting for sparsity means that the original ordering of the rows and columns of $\mathbf{G} + s_0\,\mathbf{C}$ is changed so that potential fill-in in the factors $\mathbf{L}_0$ and $\mathbf{U}_0$ is reduced. Here, nonzero entries in $\mathbf{L}_0$ and $\mathbf{U}_0$ are called fill-in if they occur at positions where $\mathbf{G} + s_0\,\mathbf{C}$ has zero entries. Pivoting for numerical stability means that the ordering is changed in order to avoid division by zero or near-zero numbers during the actual computation of the factorization. For circuit matrices, typically very little fill-in occurs in $\mathbf{L}_0$ and $\mathbf{U}_0$, although this cannot be guaranteed in general. We refer the reader to [35] for an introduction to sparse Gaussian elimination, and to [99, Chapter 6] for the application of such techniques to circuit matrices.

3.3 Linear RLC Circuits

Next, we briefly consider the important special case of linear circuits that consist of only resistors, inductors, and capacitors. For example, such linear RLC circuits, especially RC circuits that contain only resistors and capacitors, arise in interconnect analysis; see, e.g., [43, 51, 52, 80, 92].

The circuit equations describing linear RLC circuits are linear, and thus they are of the form (1). Furthermore, the equations can be formulated such that the matrices in (1) exhibit certain symmetries; see [51, 52]. More precisely, the matrices $\mathbf{G}$ and $\mathbf{C}$ are real and symmetric, and have the following block structure:

$$\mathbf{G} = \mathbf{G}^{\mathrm{T}} = \begin{bmatrix} \mathbf{G}_{11} & \mathbf{G}_{12} \\ \mathbf{G}_{12}^{\mathrm{T}} & \mathbf{0} \end{bmatrix} \quad \text{and} \quad \mathbf{C} = \mathbf{C}^{\mathrm{T}} = \begin{bmatrix} \mathbf{C}_{11} & \mathbf{0} \\ \mathbf{0} & -\mathbf{C}_{22} \end{bmatrix}. \tag{26}$$

Here, the submatrices $\mathbf{G}_{11}$, $\mathbf{C}_{11}$, and $\mathbf{C}_{22}$ are all square and symmetric positive semi-definite. If the RLC circuit is viewed as a p-terminal component with $m = p$ inputs and outputs, then the matrices $\mathbf{B}$ and $\mathbf{E}$ in (1) are identical and of the form

$$\mathbf{B} = \mathbf{E} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{0} \end{bmatrix}, \tag{27}$$

where the subblock $\mathbf{B}_1$ has the same number of rows as the blocks $\mathbf{G}_{11}$ and $\mathbf{C}_{11}$ in (26). For a p-terminal RLC circuit, in view of (26) and (27), the transfer function (4) reduces to

$$\mathbf{H}(s) = \mathbf{B}^{\mathrm{T}}\,(\mathbf{G} + s\,\mathbf{C})^{-1}\,\mathbf{B}, \quad \text{where} \quad \mathbf{G} = \mathbf{G}^{\mathrm{T}}, \quad \mathbf{C} = \mathbf{C}^{\mathrm{T}}. \tag{28}$$

We call a transfer function $\mathbf{H}$ *symmetric* if it is of the form (28). We remark that, for a p-terminal RLC circuit, the entries of the matrix function (28) are the so-called Z-parameters of the circuit.

For general RLC circuits, the matrices $\mathbf{G}$ and $\mathbf{C}$ are clearly indefinite. However, for the important special cases of RC, RL, and LC circuits, the corresponding matrices $\mathbf{G}$ and $\mathbf{C}$ can be guaranteed to be symmetric positive semi-definite; see, e.g., [51, 52]. For RC circuits, the second block row and second block column in (26) are empty, and thus

$$\mathbf{G} = \mathbf{G}_{11} \geq \mathbf{0} \quad \text{and} \quad \mathbf{C} = \mathbf{C}_{11} \geq \mathbf{0}. \tag{29}$$

Similarly, RL circuits are characterized by transfer functions (28) with $\mathbf{G} \geq \mathbf{0}$ and $\mathbf{C} \geq \mathbf{0}$. For LC circuits, the transfer function $\mathbf{H}(s)$ turns out to be a function of s^2, and it admits a representation of the form

$$\mathbf{H}(s) = \mathbf{B}^{\mathrm{T}} \left(\mathbf{G} + s^2\,\mathbf{C}\right)^{-1} \mathbf{B}, \quad \text{where} \quad \mathbf{G} \geq \mathbf{0}, \quad \mathbf{C} \geq \mathbf{0}, \tag{30}$$

see [51, 52].

Recall from (10) and (25) that, in order to employ Krylov-subspace techniques, a factorization of the matrix $\mathbf{G} + s_0\,\mathbf{C}$ is required. For RC circuits, the computation of such a factorization can often be avoided, provided the expansion point $s_0 = 0$ is chosen, so that $\mathbf{G} + s_0\,\mathbf{C}$ reduces to $\mathbf{G}$. For general RC circuits, the matrix $\mathbf{G}$ has the form

$$\mathbf{G} = \mathbf{A}_G^{\mathrm{T}}\,\mathbf{D}\,\mathbf{A}_G, \tag{31}$$

where $\mathbf{D}$ is a diagonal matrix with positive diagonal entries (namely, the inverses of the resistances of the circuit), and $\mathbf{A}_G$ is the adjacency matrix associated with the graph that describes the connectivity of the circuit. Especially for the RC circuits arising in interconnect simulations, the underlying graph is a tree, or has a tree-like structure. If the graph is a tree, the matrix $\mathbf{A}_G$, after a possible reordering of its rows and columns, is triangular. Hence, in this case, the decomposition (31) is already a sparse triangular factorization of the form (25). If the circuit is tree-like in the sense that the underlying graph is "almost" a tree with a "small" number of co-tree branches, the matrix $\mathbf{G}$ differs from a "tree matrix" $\mathbf{G}_0$ only by a matrix of low rank. In this case, the triangular factorization of $\mathbf{G}_0$ in combination with the Sherman-Morrison formula (see, e.g., [65]) can be used to efficiently handle the small-rank modification. This approach is equivalent to the one used in RICE [96].

4 Krylov Subspaces and Moment Matching

In this section, we introduce the notions of Krylov and block Krylov subspaces for single and multiple, respectively, starting vectors. We also discuss the connection of Krylov subspaces to moment matching for single-input single-output linear dynamical systems.

4.1 Assumptions and a Convention

From now on, it is always assumed that $\mathbf{A} \in \mathbb{C}^{N \times N}$ is an $N \times N$ matrix,

$$\mathbf{R} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \cdots & \mathbf{r}_m \end{bmatrix} \in \mathbb{C}^{N \times m}$$

is a matrix of m *right starting vectors*, $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_m$, and

$$\mathbf{L} = \begin{bmatrix} \mathbf{l}_1 & \mathbf{l}_2 & \cdots & \mathbf{l}_p \end{bmatrix} \in \mathbb{C}^{N \times p}$$

is a matrix of p *left starting vectors*, $\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_m$. We stress that $m \geq 1$ and $p \geq 1$ are *arbitrary* integers, and in particular, m and p need not be identical. Recall from Section 2.3 that the matrices $\mathbf{A}$, $\mathbf{R}$, and $\mathbf{L}$ arise from rewriting the transfer function of a linear dynamical system (1) in the form (11), and that m and p denote the number of inputs and outputs of (1).

In the single-input single-output case, i.e., $m = p = 1$, the matrices $\mathbf{R}$ and $\mathbf{L}$ reduce to single vectors. Whenever we explicitly consider the case $m = p = 1$, we will use lowercase letters, $\mathbf{r}$ and $\mathbf{l}$, to denote these single vectors, and lowercase letters for the moments

$$m_i = \mathbf{l}^{\mathrm{T}}\, \mathbf{A}^i\, \mathbf{r}, \quad i \geq 0. \tag{32}$$

Furthermore, the transfer function (11) is now a scalar-valued rational function of the form

$$H(s) = \mathbf{l}^{\mathrm{T}} \left(\mathbf{I} - (s - s_0)\, \mathbf{A}\right)^{-1} \mathbf{r}. \tag{33}$$

Note that we use H, instead of $\mathbf{H}$, to indicate that H is a scalar-valued function; similarly, we use H_n, instead of $\mathbf{H}_n$ for approximations to H.

4.2 Single Starting Vectors

We start with the case of single starting vectors $\mathbf{r}$ and $\mathbf{l}$. For $n = 1, 2, \ldots$, the subspace

$$\mathcal{K}_n(\mathbf{A}, \mathbf{r}) := \operatorname{span}\{\, \mathbf{r}, \mathbf{A}\,\mathbf{r}, \mathbf{A}^2\,\mathbf{r}, \ldots, \mathbf{A}^{n-1}\,\mathbf{r}\,\} \tag{34}$$

of $\mathbb{C}^N$ is called the *n-th right Krylov subspace* (induced by $\mathbf{A}$ and $\mathbf{r}$), and the subspace

$$\mathcal{K}_n(\mathbf{A}^{\mathrm{T}}, \mathbf{l}) := \operatorname{span}\{\, \mathbf{l}, \mathbf{A}^{\mathrm{T}}\,\mathbf{l}, (\mathbf{A}^{\mathrm{T}})^2\,\mathbf{l}, \ldots, (\mathbf{A}^{\mathrm{T}})^{n-1}\,\mathbf{l}\,\} \tag{35}$$

of $\mathbb{C}^N$ is called the *n-th left Krylov subspace* (induced by $\mathbf{A}^{\mathrm{T}}$ and $\mathbf{l}$). Note that

$$\dim \mathcal{K}_n(\mathbf{A}, \mathbf{r}) = n \quad \Longleftrightarrow \quad n \leq n_r := \dim \mathcal{K}_N(\mathbf{A}, \mathbf{r}), \tag{36}$$

and

$$\dim \mathcal{K}_n(\mathbf{A}^{\mathrm{T}}, \mathbf{l}) = n \quad \Longleftrightarrow \quad n \leq n_l := \dim \mathcal{K}_N(\mathbf{A}^{\mathrm{T}}, \mathbf{l}). \tag{37}$$

4.3 Connection to Moment Matching

In this subsection, we consider only the single-input single-output case, $m = p = 1$. Recall that, in this case, the transfer function H is the scalar-valued rational function given by (33). Correspondingly, the n-th Padé approximant H_n defined by (7) and (9) is now also a scalar-valued rational function. Furthermore, instead of (7), the more common way of representing H_n is in terms of polynomials:

$$H_n(s) = \frac{\varphi_{n-1}(s)}{\psi_n(s)}. \tag{38}$$

Here, φ_{n-1} and ψ_n are polynomials of degree at most $n-1$ and n, respectively. There are $2n$ free parameters in (38), namely the coefficients of the polynomials φ_{n-1} and ψ_n. Except for certain degenerate cases, these parameters can be chosen such that, in (9), the first $2n$ moments match:

$$\begin{aligned} H_n(s) &= H(s) + \mathcal{O}\left((s-s_0)^{2n}\right) \\ &= \sum_{i=0}^{2n-1} m_i\,(s-s_0)^i + \mathcal{O}\left((s-s_0)^{2n}\right). \end{aligned} \tag{39}$$

Here, the m_i's are the moments defined in (32).

The standard approach to computing H_n is based on the representation (38) and on explicit moment generation via the formula (32). First, one uses (32) to compute the leading $2n$ moments,

$$m_0, m_1, \ldots, m_{2n-1}, \tag{40}$$

of H, and from these, one then generates the coefficients of the polynomials φ_{n-1} and ψ_n in (38) by solving a system of linear equations with a Hankel matrix whose entries are the moments (40). This standard approach to computing H_n is employed in the AWE technique for circuit simulation; we refer the reader to [25, 93, 95] for a survey of AWE and its derivatives. However, computing Padé approximants using explicit moment computations is inherently numerically unstable, and indeed, in practice, this approach can be used only for very moderate values of n, such as $n \leq 10$; see [39, 40]. As we will describe in more detail in Section 6.1, the numerical problems with AWE can easily be remedied by exploiting the Lanczos-Padé connection [66] and generating the Padé approximant H_n via the classical Lanczos process [81]. This approach is called PVL.

While AWE and PVL are mathematically equivalent, their behavior when run on an actual computer can be vastly different. The reason is that AWE is a numerically unstable algorithm and thus susceptible to round-off errors caused by finite-precision arithmetic. We illustrate the numerical differences between AWE and PVL with a circuit example taken from [39, 40]. The circuit simulated here is a voltage filter, where the frequency range of interest is $1 \leq \omega \leq 10^{10}$. This example was first run with AWE, and in Figure 1, we show the computed function $|H_n(\mathrm{j}\,\omega)|$, for $n = 2, 5, 8$, together with the exact function $|H(\mathrm{j}\,\omega)|$, each for the frequency range $1 \leq \omega \leq 10^{10}$. Note that H_8 clearly has not yet converged to H. On the other hand, it turned out that the H_n's practically did not change anymore for $n \geq 8$, and in particular, AWE never converged in this example. In Figure 2, we show the computed results obtained with PVL for $n = 2, 8, 28$, together with the exact function $|H|$. Note that the results for $n = 8$ (the dotted curves) in Figures 1 and 2 are vastly different, although they both correspond to the same function H_8. The reason for this is that AWE is numerically unstable, while PVL generates H_8 stably. Furthermore, note that PVL converges, with the computed 28-th Padé approximant being practically identical to H.

The main reason for the numerical instability of AWE is the explicit generation of the moments (40) by means of the formula (32). This computation is usually done as follows. One first generates the $2n$ vectors

$$\mathbf{r}, \mathbf{A}\,\mathbf{r}, \mathbf{A}^2\,\mathbf{r}, \ldots, \mathbf{A}^{2n-1}\,\mathbf{r}, \tag{41}$$

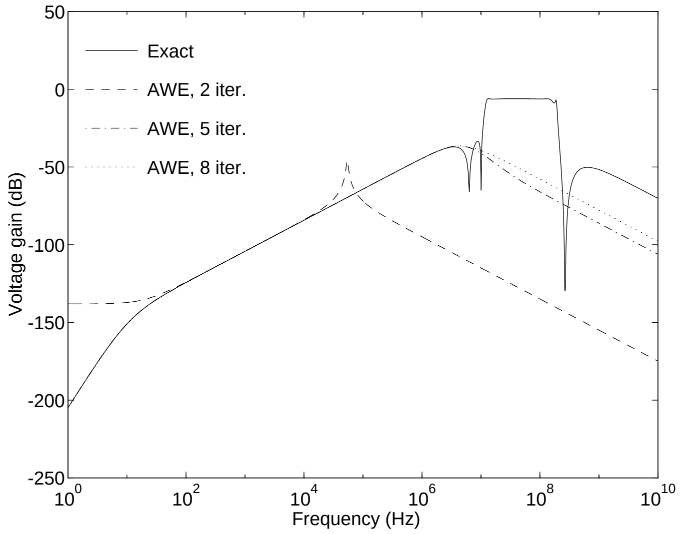

Figure 1: Results for simulation of voltage gain with AWE.

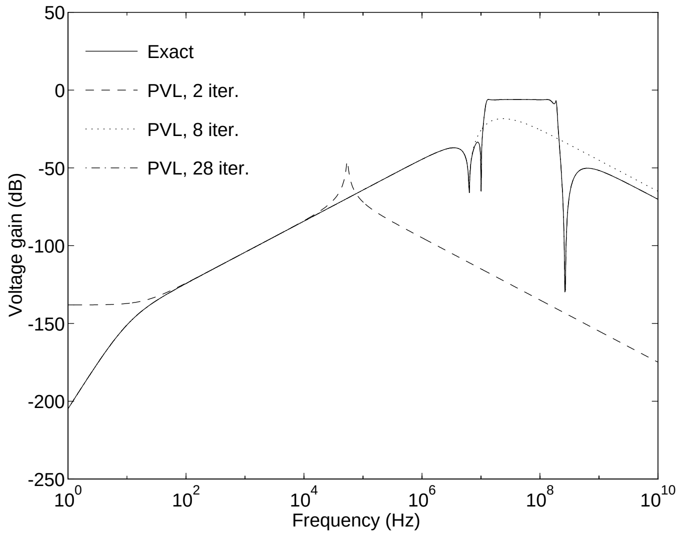

Figure 2: Results for simulation of voltage gain with PVL.

and then obtains (40) by computing the inner products

$$m_i = \mathbf{l}^{\mathrm{T}} \cdot \left(\mathbf{A}^i\, \mathbf{r}\right), \quad 0 \le i \le 2n-1, \tag{42}$$

of $\mathbf{l}$ with (41). An alternative is to first generate the vectors

$$\mathbf{r}, \mathbf{A}\,\mathbf{r}, \mathbf{A}^2\,\mathbf{r}, \ldots, \mathbf{A}^{n-1}\,\mathbf{r} \quad \text{and} \quad \mathbf{l}, \mathbf{A}^{\mathrm{T}}\,\mathbf{l}, (\mathbf{A}^{\mathrm{T}})^2\,\mathbf{l}, \ldots, (\mathbf{A}^{\mathrm{T}})^{n-1}\,\mathbf{l} \tag{43}$$

and to then obtain (40) by computing the inner products

$$m_{2i} = \left((\mathbf{A}^{\mathrm{T}})^i\,\mathbf{l}\right)^{\mathrm{T}} \cdot \left(\mathbf{A}^i\,\mathbf{r}\right) \quad \text{and} \quad m_{2i+1} = \left((\mathbf{A}^{\mathrm{T}})^i\,\mathbf{l}\right)^{\mathrm{T}} \cdot \left(\mathbf{A}^{i+1}\,\mathbf{r}\right) \tag{44}$$

for $i = 0, 1, \ldots, n-1$. The problem is that the vectors (41) quickly converge to an eigenvector corresponding to a dominant eigenvalue of $\mathbf{A}$. As a result, in finite-precision arithmetic, the moments m_i obtained from (42), already for rather moderate i, carry only information about this dominant eigenvalue. The Padé approximant H_n generated from the moments thus contains only information about part of the spectrum of $\mathbf{A}$. The transfer function H, however, depends in general on all eigenvalues of $\mathbf{A}$, and not just the dominant ones. This is the reason why, for AWE, the computed H_n usually does not converge to the transfer function H. The alternative approach suffers from the same problem since the two sequences of vectors in (43) quickly converge to a right, respectively left, eigenvector corresponding to a dominant eigenvalue of $\mathbf{A}$.

Note that the vectors (41) and (43) also appear in the definitions (34) and (35) of the Krylov subspaces $\mathcal{K}_n(\mathbf{A}, \mathbf{r})$ and $\mathcal{K}_n(\mathbf{A}^{\mathrm{T}}, \mathbf{l})$. While Krylov subspaces are very useful for large-scale matrix computations, the vectors in the definitions (34) and (35) are in general unsuitable as basis vectors. Indeed, as we just mentioned, they quickly converge, and in particular, they quickly become almost linearly dependent. The remedy is to construct more suitable basis vectors

$$\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n, \ldots, \quad \text{and} \quad \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n, \ldots, \tag{45}$$

such that, for all $n = 1, 2, \ldots$,

$$\mathcal{K}_n(\mathbf{A}, \mathbf{r}) = \operatorname{span}\{\, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n \,\} \tag{46}$$

and

$$\mathcal{K}_n(\mathbf{A}^{\mathrm{T}}, \mathbf{l}) = \operatorname{span}\{\, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n \,\}. \tag{47}$$

There are two main approaches for constructing basis vectors (45), the Lanczos algorithm and the Arnoldi process, which will be discussed in Sections 5 and 7, respectively.

Using the basis vectors (45), the explicit moment computations can now easily be avoided. Indeed, instead of the moments (40), one now computes so-called *modified moments*

$$\mathbf{w}_i^{\mathrm{T}}\,\mathbf{v}_i \quad \text{and} \quad \mathbf{w}_i^{\mathrm{T}}\,\mathbf{A}\,\mathbf{v}_i, \quad i = 1, 2, \ldots, n. \tag{48}$$

In view of (46), (47), (34), (35), and (44), the modified moments (48) contain the very same information as the moments (40), and each m_i, $0 \le i \le 2n-1$, can be expressed as a suitable linear combination of the numbers (48).

4.4 Multiple Starting Vectors

Recall from (11) that transfer functions for general m-input p-output linear dynamical systems are described by three matrices, $\mathbf{A} \in \mathbb{C}^{N \times N}$, $\mathbf{R} \in \mathbb{C}^{N \times m}$, and $\mathbf{L} \in \mathbb{C}^{N \times p}$. In the context of block Krylov subspaces and Lanczos-type algorithms, we refer to the columns of $\mathbf{R}$ and $\mathbf{L}$ as right and left starting vectors, respectively. In this subsection, we introduce our notion of block Krylov subspaces induced by the data $\mathbf{A}$, $\mathbf{R}$, and $\mathbf{L}$.

We start by defining the *right block Krylov matrix*

$$\mathbf{K}(\mathbf{A},\mathbf{R}) := \begin{bmatrix}\mathbf{R} & \mathbf{A}\,\mathbf{R} & \mathbf{A}^2\,\mathbf{R} & \cdots & \mathbf{A}^{N-1}\,\mathbf{R}\end{bmatrix} \tag{49}$$

and the *left block Krylov matrix*

$$\mathbf{K}(\mathbf{A}^{\mathrm{T}},\mathbf{L}) := \begin{bmatrix}\mathbf{L} & \mathbf{A}^{\mathrm{T}}\,\mathbf{L} & (\mathbf{A}^{\mathrm{T}})^2\,\mathbf{L} & \cdots & (\mathbf{A}^{\mathrm{T}})^{N-1}\,\mathbf{L}\end{bmatrix}. \tag{50}$$

Our goal is to define two sequences of ascending n-dimensional subspaces, $n = 1,2,\ldots,$ that are spanned by the first n linearly independent columns of the matrices $\mathbf{K}(\mathbf{A},\mathbf{R})$ and $\mathbf{K}(\mathbf{A}^{\mathrm{T}},\mathbf{L})$, respectively. To properly define these subspaces, we need to delete the linearly dependent columns in (49) and (50). This is done by scanning the columns of each of the matrices $\mathbf{K}(\mathbf{A},\mathbf{R})$ and $\mathbf{K}(\mathbf{A}^{\mathrm{T}},\mathbf{L})$, from left to right and deleting each column that is linearly dependent on earlier columns within the same matrix. This process of deleting linearly dependent vectors is referred to as *exact deflation* in the sequel. Applying exact deflation to (49) and (50), we obtain the *deflated* right and left block Krylov matrices $\mathbf{K}^{\mathrm{dl}}(\mathbf{A},\mathbf{R})$ and $\mathbf{K}^{\mathrm{dl}}(\mathbf{A}^{\mathrm{T}},\mathbf{L})$, respectively. By the structure (49) of $\mathbf{K}(\mathbf{A},\mathbf{R})$, a column $\mathbf{A}^{j-1}\mathbf{r}_i$ being linearly dependent on earlier columns implies that all columns $\mathbf{A}^k\mathbf{r}_i$, $j = k, k+1, \ldots, N-1$, are also linearly dependent on earlier columns. An analogous statement holds for (50). Consequently, $\mathbf{K}^{\mathrm{dl}}(\mathbf{A},\mathbf{R})$ and $\mathbf{K}^{\mathrm{dl}}(\mathbf{A}^{\mathrm{T}},\mathbf{L})$ have the following form:

$$\begin{aligned}\mathbf{K}^{\mathrm{dl}}(\mathbf{A},\mathbf{R}) &= \begin{bmatrix}\mathbf{R}_1 & \mathbf{A}\,\mathbf{R}_2 & \mathbf{A}^2\,\mathbf{R}_3 & \cdots & \mathbf{A}^{j_{\max}-1}\,\mathbf{R}_{j_{\max}}\end{bmatrix},\\ \mathbf{K}^{\mathrm{dl}}(\mathbf{A}^{\mathrm{T}},\mathbf{L}) &= \begin{bmatrix}\mathbf{L}_1 & \mathbf{A}^{\mathrm{T}}\,\mathbf{L}_2 & (\mathbf{A}^{\mathrm{T}})^2\,\mathbf{L}_3 & \cdots & (\mathbf{A}^{\mathrm{T}})^{k_{\max}-1}\,\mathbf{L}_{k_{\max}}\end{bmatrix}.\end{aligned} \tag{51}$$

Here, for each $j = 1,2,\ldots,j_{\max}$, $\mathbf{R}_j$ is a submatrix of $\mathbf{R}_{j-1}$, with $\mathbf{R}_j \neq \mathbf{R}_{j-1}$ if, and only if, exact deflation occurs within the j-th right Krylov block $\mathbf{A}^{j-1}\,\mathbf{R}$ in (49). (For $j = 1$, we set $\mathbf{R}_0 = \mathbf{R}$.) Similarly, for each $k = 1,2,\ldots,k_{\max}$, $\mathbf{L}_k$ is a submatrix of $\mathbf{L}_{k-1}$, with $\mathbf{L}_k \neq \mathbf{L}_{k-1}$ if, and only if, exact deflation occurs within the k-th left Krylov block $(\mathbf{A}^{\mathrm{T}})^{k-1}\,\mathbf{L}$ in (50). (For $k = 1$, we set $\mathbf{L}_0 = \mathbf{L}$.)

By construction, the columns of each matrix $\mathbf{K}^{\mathrm{dl}}(\mathbf{A},\mathbf{R})$ and $\mathbf{K}^{\mathrm{dl}}(\mathbf{A}^{\mathrm{T}},\mathbf{L})$ are linearly independent. Furthermore, $\mathbf{K}^{\mathrm{dl}}(\mathbf{A},\mathbf{R})$ and $\mathbf{K}^{\mathrm{dl}}(\mathbf{A}^{\mathrm{T}},\mathbf{L})$ have n_r and n_l columns, respectively, where

$$n_r := \operatorname{rank}\mathbf{K}(\mathbf{A},\mathbf{R}) \quad \text{and} \quad n_l := \operatorname{rank}\mathbf{K}(\mathbf{A}^{\mathrm{T}},\mathbf{L}). \tag{52}$$

For $n = 1,2,\ldots,n_r$, we denote by $\mathcal{K}_n(\mathbf{A},\mathbf{R})$ the subspace of $\mathbb{C}^N$ spanned by the first n columns of the matrix $\mathbf{K}^{\mathrm{dl}}(\mathbf{A},\mathbf{R})$ in (51). We call $\mathcal{K}_n(\mathbf{A},\mathbf{R})$ the *n-th right block Krylov subspace* (induced by $\mathbf{A}$ and $\mathbf{R}$). Similarly, for $n = 1,2,\ldots,n_l$, the *n-th left block Krylov subspace* (induced by $\mathbf{A}^{\mathrm{T}}$ and $\mathbf{L}$), denoted by $\mathcal{K}_n(\mathbf{A}^{\mathrm{T}},\mathbf{L})$, is defined as the subspace of $\mathbb{C}^N$ spanned by the first n columns of the matrix $\mathbf{K}^{\mathrm{dl}}(\mathbf{A}^{\mathrm{T}},\mathbf{L})$ in (51). By construction, $\mathcal{K}_n(\mathbf{A},\mathbf{R})$ and $\mathcal{K}_n(\mathbf{A}^{\mathrm{T}},\mathbf{L})$ are subspaces of dimension n for $n \leq n_r$ and $n \leq n_l$, respectively. Furthermore, we remark that for single starting vectors $\mathbf{R} = \mathbf{r}$ and $\mathbf{L} = \mathbf{l}$, the block Krylov subspaces introduced in this subsection are identical to the Krylov subspaces $\mathcal{K}_n(\mathbf{A},\mathbf{r})$ and $\mathcal{K}_n(\mathbf{A}^{\mathrm{T}},\mathbf{l})$ defined by (34) and (35).

Note that in our construction of block Krylov subspaces, we only have used exact deflation. In an actual algorithm for constructing basis vectors for $\mathcal{K}_n(\mathbf{A}, \mathbf{R})$ and $\mathcal{K}_n(\mathbf{A}^{\mathrm{T}}, \mathbf{L})$ in finite-precision arithmetic, one also needs to delete vectors that are in some sense "almost" linearly dependent on earlier vectors. We will refer to the deletion of such almost linearly dependent vectors as *inexact deflation.* In Section 5.2 below, we describe how exact and inexact deflation can be built easily into a Lanczos-type algorithm for multiple starting vectors. In the sequel, the word "deflation" is used whenever both exact and inexact deflation are performed. While inexact deflation is crucial in practice, concise statements of theoretical results are much simpler if only exact deflation is performed. Throughout this paper, theoretical results are thus formulated for exact deflation only.

5 The Lanczos Process

In this section, we review the classical Lanczos process and some of its extensions.

5.1 The Classical Algorithm for Single Starting Vectors

Given a matrix $\mathbf{A} \in \mathbb{C}^{N \times N}$, a right starting vector $\mathbf{r} \in \mathbb{C}^N$, and a left starting vector $\mathbf{l} \in \mathbb{C}^N$, the classical Lanczos algorithm [81] generates two sequences of vectors,

$$\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{N_0} \quad \text{and} \quad \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{N_0}, \tag{53}$$

the so-called *right and left Lanczos vectors.* Here,

$$N_0 := \min\{n_r, n_l\}, \tag{54}$$

where n_r and n_l are defined in (36) and (37). The Lanczos vectors (53) span the Krylov subspaces $\mathcal{K}_n(\mathbf{A}, \mathbf{r})$ and $\mathcal{K}_n(\mathbf{A}^{\mathrm{T}}, \mathbf{l})$, i.e., they satisfy (46) and (47), for all $n = 1, 2, \ldots, N_0$. Moreover, the Lanczos vectors are constructed to be biorthogonal:

$$\mathbf{w}_i^{\mathrm{T}} \mathbf{v}_n = \begin{cases} \delta_n & \text{if } i = n, \\ 0 & \text{if } i \neq n, \end{cases} \qquad \text{for all} \quad 1 \leq i, n \leq N_0. \tag{55}$$

The classical Lanczos algorithm employs three-term recurrences to generate the Lanczos vectors. To state these recurrences, as well as other properties of the algorithm, it is convenient to introduce the notation

$$\mathbf{V}_n := \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{bmatrix} \quad \text{and} \quad \mathbf{W}_n := \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \cdots & \mathbf{w}_n \end{bmatrix} \tag{56}$$

for the matrices whose columns are just the first n right and left Lanczos vectors. With (56), the biorthogonality condition (55) can be stated as follows:

$$\mathbf{W}_n^{\mathrm{T}} \mathbf{V}_n = \mathbf{D}_n := \operatorname{diag}(\delta_1, \delta_2, \ldots, \delta_n) \quad \text{for all} \quad 1 \leq n \leq N_0. \tag{57}$$

The recurrences used to compute the first n right and left Lanczos vectors can be summarized as follows:

$$\begin{aligned} \mathbf{A}\,\mathbf{V}_n &= \mathbf{V}_n\,\mathbf{T}_n + [\,\underbrace{\mathbf{0} \quad \cdots \quad \mathbf{0}}_{n-1} \quad \hat{\mathbf{v}}_{n+1}\,], \\ \mathbf{A}^{\mathrm{T}}\,\mathbf{W}_n &= \mathbf{W}_n\,\widetilde{\mathbf{T}}_n + [\,\underbrace{\mathbf{0} \quad \cdots \quad \mathbf{0}}_{n-1} \quad \hat{\mathbf{w}}_{n+1}\,]. \end{aligned} \tag{58}$$

Here,

$$\mathbf{T}_n = \begin{bmatrix} t_{11} & t_{12} & 0 & \cdots & 0 \\ t_{21} & t_{22} & t_{23} & \ddots & \vdots \\ 0 & t_{32} & t_{33} & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & t_{n-1,n} \\ 0 & \cdots & 0 & t_{n,n-1} & t_{nn} \end{bmatrix} \tag{59}$$

and

$$\widetilde{\mathbf{T}}_n = \begin{bmatrix} \tilde{t}_{11} & \tilde{t}_{12} & 0 & \cdots & 0 \\ \tilde{t}_{21} & \tilde{t}_{22} & \tilde{t}_{23} & \ddots & \vdots \\ 0 & \tilde{t}_{32} & \tilde{t}_{33} & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \tilde{t}_{n-1,n} \\ 0 & \cdots & 0 & \tilde{t}_{n,n-1} & \tilde{t}_{nn} \end{bmatrix} \tag{60}$$

are tridiagonal $n \times n$ matrices, reflecting the fact that three-term recurrences are employed. Moreover, in (58), $\hat{\mathbf{v}}_{n+1}$ and $\hat{\mathbf{w}}_{n+1}$ are unnormalized versions of the Lanczos vectors $\mathbf{v}_{n+1}$ and $\mathbf{w}_{n+1}$, respectively, and in particular, they satisfy the biorthogonality conditions

$$\mathbf{W}_n^{\mathrm{T}}\,\hat{\mathbf{v}}_{n+1} = \mathbf{0} \quad \text{and} \quad \mathbf{V}_n^{\mathrm{T}}\,\hat{\mathbf{w}}_{n+1} = \mathbf{0}. \tag{61}$$

At the $(n+1)$-st step of the Lanczos algorithm, $\hat{\mathbf{v}}_{n+1}$ and $\hat{\mathbf{w}}_{n+1}$ are then normalized to yield $\mathbf{v}_{n+1}$ and $\mathbf{w}_{n+1}$. Such a normalization is necessary to avoid potential over- and underflow; usually, one normalizes the Lanczos vectors to have unit length:

$$\|\mathbf{v}_n\| = \|\mathbf{w}_n\| = 1 \quad \text{for all} \quad n.$$

We remark that the matrices (59) and (60) essentially contain the same information. In fact, they are connected through the relation

$$\widetilde{\mathbf{T}}_n^{\mathrm{T}} = \mathbf{D}_n\,\mathbf{T}_n\,\mathbf{D}_n^{-1}, \tag{62}$$

which readily follows by multiplying the first and second equations in (58) from the left by $\mathbf{W}_n^{\mathrm{T}}$ and $\mathbf{V}_n^{\mathrm{T}}$, respectively, and by using (57) and (61).

Next, we present a complete statement of the classical Lanczos algorithm.

Algorithm 5.1 (Classical Lanczos algorithm without look-ahead.)

INPUT: A matrix $\mathbf{A} \in \mathbb{C}^{N \times N}$;
Right and left starting vectors $\mathbf{r}, \mathbf{l} \in \mathbb{C}^N$.

OUTPUT: An $n \times n$ tridiagonal matrix $\mathbf{T}_n = \left[t_{ij}\right]_{1 \le i,j \le n}$, where n is the iteration index at termination of the algorithm.

(0) *Set* $\hat{\mathbf{v}}_1 = \mathbf{r}$, $\hat{\mathbf{w}}_1 = \mathbf{l}$, *and* $\mathbf{v}_0 = \mathbf{w}_0 = \mathbf{0}_N$.
Set $\delta_0 = 1$.

For $n = 1, 2, \ldots,$ *do* (Build n-th pair of Lanczos vectors $\mathbf{v}_n$ and $\mathbf{w}_n$.) :

(1) *Compute* $t_{n,n-1} = \|\hat{\mathbf{v}}_n\|$ *and* $\tilde{t}_{n,n-1} = \|\hat{\mathbf{w}}_n\|$.
If $t_{n,n-1} = 0$ *or* $\tilde{t}_{n,n-1} = 0$, *then stop.*
(The right, respectively left, Krylov subspace is exhausted, i.e., $n = N_0 + 1$.)

(2) *Set* $\mathbf{v}_n = \dfrac{\hat{\mathbf{v}}_n}{t_{n,n-1}}$ *and* $\mathbf{w}_n = \dfrac{\hat{\mathbf{w}}_n}{\tilde{t}_{n,n-1}}$.

(3) *Compute* $\delta_n = \mathbf{w}_n^{\mathrm{T}} \mathbf{v}_n$.
If $\delta_n = 0$, *then stop.*
(In this case, look-ahead would be needed to continue the algorithm.)

(4) *Set*

$$
\begin{aligned}
t_{n-1,n} &= \tilde{t}_{n,n-1} \frac{\delta_n}{\delta_{n-1}}, \\
\hat{\mathbf{v}}_{n+1} &= \mathbf{A}\,\mathbf{v}_n - \mathbf{v}_{n-1}\, t_{n-1,n}, \\
t_{nn} &= \frac{\mathbf{w}_n^{\mathrm{T}}\, \hat{\mathbf{v}}_{n+1}}{\delta_n}, \quad \tilde{t}_{nn} = t_{nn}, \quad \tilde{t}_{n-1,n} = t_{n,n-1} \frac{\delta_n}{\delta_{n-1}}, \\
\hat{\mathbf{v}}_{n+1} &= \hat{\mathbf{v}}_{n+1} - \mathbf{v}_n\, t_{nn}, \\
\hat{\mathbf{w}}_{n+1} &= \mathbf{A}^{\mathrm{T}}\, \mathbf{w}_n - \mathbf{w}_n\, \tilde{t}_{nn} - \mathbf{w}_{n-1}\, \tilde{t}_{n-1,n}.
\end{aligned}
$$

Remark 5.1 In Algorithm 5.1, only the subdiagonal entries of $\mathbf{T}_n$ and $\widetilde{\mathbf{T}}_n$, and the diagonal entries of $\mathbf{T}_n$ are directly computed as inner products or norms of vectors of length N. The superdiagonal entries of $\mathbf{T}_n$ and $\widetilde{\mathbf{T}}_n$, and the diagonal entries of $\widetilde{\mathbf{T}}_n$ are obtained via a few scalar operations, by exploiting the relation (62). This procedure minimizes computational costs, and, in general, is also more numerically stable than directly computing all entries of $\mathbf{T}_n$ and $\widetilde{\mathbf{T}}_n$.

Remark 5.2 If $\delta_n = 0$ in Step (3), Algorithm 5.1 must be terminated prematurely, because otherwise a division by $\delta_n = 0$ would occur in Step (4). The occurrence of $\delta_n = 0$ is called an *exact breakdown.* In finite-precision arithmetic, one also needs to check if δ_n is in some sense close to zero; such an event is called a *near-breakdown.* Although exact breakdowns or near-breakdowns occur rarely in practice, their possible occurrence had brought the classical Lanczos algorithm into disrepute. In recent years, starting with [91], so-called *look-ahead* variants of the Lanczos process were developed that remedy the problem of breakdowns; see, e.g., [13, 53, 54] and the references given therein. Furthermore, a FORTRAN implementation of the look-ahead Lanczos algorithm described in [54] is available in public domain, as part of the software package QMRPACK [58].

Remark 5.3 For the sake of simplicity, in this paper, we only present explicit statements of the Lanczos algorithm and of Lanczos-type algorithms without look-ahead. Instead, we only describe very briefly the basic concept of look-ahead. The main idea is to relax the vector-wise biorthogonality (55) to a cluster-wise biorthogonality condition whenever an exact breakdown or a near-breakdown would occur in the classical Lanczos algorithm. The matrix $\mathbf{D}_n$ in (57) then becomes block diagonal, where each diagonal block is square and its size is equal to the number of vectors in the associated clusters of right and left Lanczos vectors. In particular, a diagonal block of $\mathbf{D}_n$ of size bigger than one corresponds to the occurrence of an exact breakdown or a near-breakdown. The clusters are constructed such that $\mathbf{v}_n$ and $\mathbf{w}_n$ are the last vectors in the cluster if, and only if, $\mathbf{D}_n$ is in some sense sufficiently nonsingular. Furthermore, the Lanczos vectors are generated by recurrences that involve three successive clusters of vectors, and thus the matrices $\mathbf{T}_n$ and $\widetilde{\mathbf{T}}_n$ in (58) become block tridiagonal, where the diagonal blocks are all square and of the same size as the blocks of $\mathbf{D}_n$.

5.2 A Lanczos-Type Algorithm for Multiple Starting Vectors

We now return to the general m-input p-output case, where $m,\, p \geq 1$ are *arbitrary* integers. Here, the given data are three matrices $\mathbf{A} \in \mathbb{C}^{N\times N}$, $\mathbf{R} \in \mathbb{C}^{N\times m}$, and $\mathbf{L} \in \mathbb{C}^{N\times p}$. The goal is to construct right and left Lanczos vectors (53) that span the right and left block Krylov subspaces $\mathcal{K}_n(\mathbf{A},\mathbf{R})$ and $\mathcal{K}_n(\mathbf{A}^{\mathrm{T}},\mathbf{L})$, respectively, i.e.,

$$\begin{aligned} \mathcal{K}_n(\mathbf{A},\mathbf{R}) &= \operatorname{span}\{\, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n \,\} \\ \text{and} \quad \mathcal{K}_n(\mathbf{A}^{\mathrm{T}},\mathbf{L}) &= \operatorname{span}\{\, \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n \,\} \end{aligned} \tag{63}$$

for all $n = 1, 2, \ldots, N_0$. Here, N_0 is defined as in (54) with n_r and n_l given by (52), and hence

$$N_0 = \min\{\, \operatorname{rank} \mathbf{K}(\mathbf{A},\mathbf{R}),\, \operatorname{rank} \mathbf{K}(\mathbf{A}^{\mathrm{T}},\mathbf{L}) \,\}. \tag{64}$$

Recall that the matrices $\mathbf{K}(\mathbf{A},\mathbf{R})$, $\mathbf{K}(\mathbf{A}^{\mathrm{T}},\mathbf{L})$ and the block Krylov subspaces $\mathcal{K}_n(\mathbf{A},\mathbf{R})$, $\mathcal{K}_n(\mathbf{A}^{\mathrm{T}},\mathbf{L})$ were introduced in Section 4.4.

For general data $\mathbf{A}$, $\mathbf{R}$, and $\mathbf{L}$, a Lanczos-type algorithm for constructing basis vectors (63) must have the following three features:

(i) It can handle arbitrary, in general different, block sizes m and p.

(ii) It has a built-in deflation procedure to detect and delete linearly dependent and almost linearly dependent Krylov vectors. Deflation in the right and left Krylov sequences occurs, in general, at different stages of the algorithm. Furthermore, each deflation reduces the corresponding block size by one. As a result, the effective right and left block sizes are different once the first deflation has occurred, even if $m = p$ initially.

(iii) Just as in the case of single starting vectors, breakdowns cannot be excluded, and thus look-ahead needs to be incorporated.

A first such Lanczos-type algorithm, which has all features (i)–(iii), was developed only recently, by Aliaga, Boley, Freund, and Hernández [2]. A key property of this algorithm is the vector-wise, rather than block-wise, construction of the Lanczos vectors (53), even though they span block Krylov subspaces. Furthermore, as in the case of single starting vectors, the Lanczos vectors still satisfy the vector-wise (respectively cluster-wise) biorthogonality condition (57) where $\mathbf{D}_n$ is diagonal if no look-ahead occurs (respectively block diagonal if clusters are built due to look-ahead). It is this vector-wise construction of the Lanczos vectors that allows to handle different block sizes (see (i) above), and leads to a simple deflation procedure (see (ii) above). In contrast, earlier nonsymmetric Lanczos-type procedures for multiple starting vectors, such as the ones in [78, 79, 90], generate block-biorthogonal Lanczos vectors in a block-wise fashion. However, such a block-wise construction is restricted to the special case $m = p$ of identical right and left block sizes. In addition, in view of (ii) above, deflation is not possible, since it would result in different block sizes. We remark that, for Hermitian matrices $\mathbf{A}$ and single blocks of starting vectors $\mathbf{R}$, there are earlier Lanczos-type algorithms that have built-in deflation procedures; see [29, 87, 101]. However, the Hermitian case is substantially simpler, since the issue (i) of different block sizes and the issue (iii) of look-ahead do not arise here. Nevertheless, the Lanczos-type method proposed in [2] can be viewed as an extension of the Hermitian algorithm [101] to the general non-Hermitian case.

Next, we describe a variant of the algorithm [2] that originated from [41] and was further developed in [47, 49]. For simplicity, we only present a version of the algorithm without look-ahead. The Lanczos-type algorithm uses essentially $(m+p+1)$-term recurrences to construct right and left Lanczos vectors (53) that satisfy the biorthogonality condition (57), and that in the case of exact deflation, fulfill the basis property (63). Using again the notation (56), these recurrences can be summarized in matrix form as follows:

$$\begin{aligned}
\mathbf{A}\,\mathbf{V}_n &= \mathbf{V}_n\,\mathbf{T}_n + \big[\,\underbrace{\mathbf{0}\ \ \cdots\ \ \mathbf{0}}_{n-m_c}\ \ \underbrace{\hat{\mathbf{v}}_{n+1}\ \ \cdots\ \ \hat{\mathbf{v}}_{n+m_c}}_{m_c}\,\big] + \mathbf{V}_n^{\mathrm{dl}},\\
\mathbf{A}^{\mathrm{T}}\,\mathbf{W}_n &= \mathbf{W}_n\,\widetilde{\mathbf{T}}_n + \big[\,\underbrace{\mathbf{0}\ \ \cdots\ \ \mathbf{0}}_{n-p_c}\ \ \underbrace{\hat{\mathbf{w}}_{n+1}\ \ \cdots\ \ \hat{\mathbf{w}}_{n+p_c}}_{p_c}\,\big] + \mathbf{W}_n^{\mathrm{dl}}.
\end{aligned} \tag{65}$$

Here, $\mathbf{T}_n$ and $\widetilde{\mathbf{T}}_n$ are $n \times n$ matrices that are banded if only exact deflation is performed, and that are banded plus a few rows with entries close to zero in their upper triangular parts if also inexact deflation is performed. The banded part of the matrix $\mathbf{T}_n$ has lower and upper bandwidth m and p, respectively, and the banded part of the matrix $\widetilde{\mathbf{T}}_n$ has lower and upper bandwidth p and m, respectively. Moreover, in (65),

$$\hat{\mathbf{v}}_{n+1}, \hat{\mathbf{v}}_{n+2}, \ldots, \hat{\mathbf{v}}_{n+m_c} \quad \text{and} \quad \hat{\mathbf{w}}_{n+1}, \hat{\mathbf{w}}_{n+2}, \ldots, \hat{\mathbf{w}}_{n+p_c} \tag{66}$$

are *auxiliary vectors* that are biorthogonal to the first n left and right Lanczos vectors, respectively:

$$\begin{aligned}
\mathbf{W}_n^{\mathrm{T}}\,\big[\hat{\mathbf{v}}_{n+1}\ \ \hat{\mathbf{v}}_{n+2}\ \ \cdots\ \ \hat{\mathbf{v}}_{n+m_c}\big] &= \mathbf{0}\\
\text{and}\quad \mathbf{V}_n^{\mathrm{T}}\,\big[\hat{\mathbf{w}}_{n+1}\ \ \hat{\mathbf{w}}_{n+2}\ \ \cdots\ \ \hat{\mathbf{w}}_{n+p_c}\big] &= \mathbf{0}.
\end{aligned}$$

The vectors (66) are turned into right and left Lanczos vectors, or deflated, during the next m_c, respectively p_c, iterations of the algorithm. More precisely, $\hat{\mathbf{v}}_{n+1} = \mathbf{0}$ ($\hat{\mathbf{w}}_{n+1} = \mathbf{0}$) if, and only if, an exact deflation occurs in the right (left) block Krylov sequence. Therefore, if only exact deflation is performed, auxiliary vectors $\hat{\mathbf{v}}_{n+1} = \mathbf{0}$, respectively $\hat{\mathbf{w}}_{n+1} = \mathbf{0}$, are deleted. If inexact deflation is included, then vectors with $\|\hat{\mathbf{v}}_{n+1}\| \le \texttt{dtol}$, respectively $\|\hat{\mathbf{w}}_{n+1}\| \le \texttt{dtol}$, are deleted; here `dtol` is a suitably chosen, small deflation tolerance. The integers m_c and p_c in (65) and (66) denote the *current* block sizes. Initially, $m_c = m$ and $p_c = p$. In the algorithm they are then reset to $m_c := m_c - 1$, respectively $p_c := p_c - 1$, whenever a deflation occurs in the right, respectively left, block Krylov sequence. Finally, the matrices $\mathbf{V}_n^{\mathrm{dl}}$ and $\mathbf{W}_n^{\mathrm{dl}}$ in (65) contain zero vectors, together with the deflated vectors. In particular, in the case of exact deflation only, $\mathbf{V}_n^{\mathrm{dl}} = \mathbf{W}_n^{\mathrm{dl}} = \mathbf{0}$.

After these preliminaries, we now present a precise statement of the Lanczos-type algorithm for multiple starting vectors.

Algorithm 5.2 (Lanczos-type algorithm for multiple starting vectors; this version has deflation, but no look-ahead.)

INPUT: A matrix $\mathbf{A} \in \mathbb{C}^{N \times N}$;

A block of m right starting vectors

$$\mathbf{R} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \cdots & \mathbf{r}_m \end{bmatrix} \in \mathbb{C}^{N \times m};$$

A block of p left starting vectors

$$\mathbf{L} = \begin{bmatrix} \mathbf{l}_1 & \mathbf{l}_2 & \cdots & \mathbf{l}_p \end{bmatrix} \in \mathbb{C}^{N \times p};$$

A deflation tolerance `dtol`.

(Set `dtol`:=0 for exact deflation only.)

OUTPUT: An $n \times n$ matrix $\mathbf{T}_n = \begin{bmatrix} t_{ij} \end{bmatrix}_{1 \le i,j \le n}$, where n is the iteration

index at termination of the algorithm;

An $m_1 \times m$ upper triangular matrix $\boldsymbol{\rho} = \begin{bmatrix} \rho_{ij} \end{bmatrix}_{1 \le i \le m_1, 1 \le j \le m}$,

where $m - m_1$ is the number of deflations during the first m

iterations of the algorithm;

An $p_1 \times p$ upper triangular matrix $\boldsymbol{\eta} = \begin{bmatrix} \eta_{ij} \end{bmatrix}_{1 \le i \le p_1, 1 \le j \le p}$,

where $p - p_1$ is the number of deflations during the first p

iterations of the algorithm.

(0) *For* $i = 1, 2, \ldots, m$, *set* $\hat{\mathbf{v}}_i = \mathbf{r}_i$.

For $i = 1, 2, \ldots, p$, *set* $\hat{\mathbf{w}}_i = \mathbf{l}_i$.

Set $m_{\mathrm{c}} = m$ *and* $p_{\mathrm{c}} = p$.

(m_{c} and p_{c} are the current right and left block sizes, respectively.)

Set $\mathcal{I}_{\mathbf{v}} = \emptyset$ *and* $\mathcal{I}_{\mathbf{w}} = \emptyset$.

($\mathcal{I}_{\mathbf{v}}$ and $\mathcal{I}_{\mathbf{w}}$ record indices of vectors that must be preserved due to inexact deflation.

If $i \in \mathcal{I}_{\mathbf{v}}$, respectively $i \in \mathcal{I}_{\mathbf{w}}$, then the newly constructed $\mathbf{v}$, respectively $\mathbf{w}$, vectors need to be biorthogonalized against $\mathbf{w}_i$, respectively $\mathbf{v}_i$.)

For $n = 1, 2, \ldots,$ *do* (Build n-th pair of Lanczos vectors $\mathbf{v}_n$ and $\mathbf{w}_n$.) :

(1) (If necessary, deflate $\hat{\mathbf{v}}_n$.)
If $\|\hat{\mathbf{v}}_n\| > \texttt{dtol}$, *then continue with Step* (2).
Otherwise, deflate $\hat{\mathbf{v}}_n$ *by doing the following* :

(1a) *If* $m_{\mathrm{c}} = 1$, *then stop.*
(The right block Krylov subspace is exhausted.)

(1b) *If* $\mu = n - m_{\mathrm{c}} > 0$ *and the deflated vector* $\hat{\mathbf{v}}_n$ *is nonzero, then set* $\mathcal{I}_{\mathbf{w}} = \mathcal{I}_{\mathbf{w}} \cup \{\mu\}$ *and save the vectors* $\mathbf{v}_\mu$ *and* $\mathbf{w}_\mu$.

(1c) *For* $i = n, n+1, \ldots, n + m_{\mathrm{c}} - 2$, *set* $\hat{\mathbf{v}}_i = \hat{\mathbf{v}}_{i+1}$.
(The vector $\hat{\mathbf{v}}_n$ is deflated. The indices of the remaining right auxiliary vectors are reduced by one.)
Set $m_{\mathrm{c}} = m_{\mathrm{c}} - 1$.
(The current right block size is reduced by one.)

(1d) *Repeat all of Step* (1).

(2) (If necessary, deflate $\hat{\mathbf{w}}_n$.)
If $\|\hat{\mathbf{w}}_n\| > \texttt{dtol}$, *then continue with Step* (3).
Otherwise, deflate $\hat{\mathbf{w}}_n$ *by doing the following* :

(2a) *If* $p_{\mathrm{c}} = 1$, *then stop.*
(The left block Krylov subspace is exhausted.)

(2b) *If* $\phi = n - p_{\mathrm{c}} > 0$ *and the deflated vector* $\hat{\mathbf{w}}_n$ *is nonzero, then set* $\mathcal{I}_{\mathbf{v}} = \mathcal{I}_{\mathbf{v}} \cup \{\phi\}$ *and save the vectors* $\mathbf{v}_\phi$ *and* $\mathbf{w}_\phi$.

(2c) *For* $i = n, n+1, \ldots, n + p_{\mathrm{c}} - 1$, *set* $\hat{\mathbf{w}}_i = \hat{\mathbf{w}}_{i+1}$.
(The vector $\hat{\mathbf{w}}_n$ is deflated. The indices of the remaining left auxiliary vectors are reduced by one.)
Set $p_{\mathrm{c}} = p_{\mathrm{c}} - 1$.
(The current left block size is reduced by one.)

(2d) *Repeat all of Step* (2).

(3) (Normalize $\hat{\mathbf{v}}_n$ and $\hat{\mathbf{w}}_n$ to obtain $\mathbf{v}_n$ and $\mathbf{w}_n$.)
Set $\mu = n - m_{\mathrm{c}}$, $\phi = n - p_{\mathrm{c}}$, $t_{n,\mu} = \|\hat{\mathbf{v}}_n\|$, $\tilde{t}_{n,\phi} = \|\hat{\mathbf{w}}_n\|$,

$$\mathbf{v}_n = \frac{\hat{\mathbf{v}}_n}{t_{n,\mu}}, \quad \textit{and} \quad \mathbf{w}_n = \frac{\hat{\mathbf{w}}_n}{\tilde{t}_{n,\phi}}.$$

(4) (Advance the right block Krylov subspace and obtain $\hat{\mathbf{v}}_{n+m_{\mathrm{c}}}$.)

(4a) *Set* $\mathbf{v} = \mathbf{A}\,\mathbf{v}_n$.

(4b) (Biorthogonalize $\mathbf{v}$ against previous Lanczos vectors $\mathbf{w}_i$, $i < n$.)
Set $i_{\mathbf{v}} = \max\{1, n - p_c\}$ *and define the temporary index set*

$$\mathcal{I} = \mathcal{I}_{\mathbf{v}} \cup \{\, i_{\mathbf{v}}, i_{\mathbf{v}} + 1, \ldots, n-1 \,\}.$$

For all $i \in \mathcal{I}$ *(in ascending order), set*

$$t_{i,n} = \begin{cases} \tilde{t}_{n,i}\, \dfrac{\delta_n}{\delta_i} & \text{if } i = n - p_c, \\[2ex] \dfrac{\mathbf{w}_i^{\mathrm{T}} \mathbf{v}}{\delta_i} & \text{otherwise,} \end{cases}$$
$$\mathbf{v} = \mathbf{v} - \mathbf{v}_i\, t_{i,n}.$$

Set $\hat{\mathbf{v}}_{n+m_c} = \mathbf{v}$.

(5) (Advance the left block Krylov subspace and obtain $\hat{\mathbf{w}}_{n+p_c}$.)

(5a) *Set* $\mathbf{w} = \mathbf{A}^{\mathrm{T}} \mathbf{w}_n$.

(5b) (Biorthogonalize $\mathbf{w}$ against previous Lanczos vectors $\mathbf{v}_i$, $i < n$.)
Set $i_{\mathbf{w}} = \max\{1, n - m_c\}$ *and define the temporary index set*

$$\mathcal{I} = \mathcal{I}_{\mathbf{w}} \cup \{\, i_{\mathbf{w}}, i_{\mathbf{w}} + 1, \ldots, n-1 \,\}.$$

For all $i \in \mathcal{I}$ *(in ascending order), set*

$$\tilde{t}_{i,n} = \begin{cases} t_{n,i}\, \dfrac{\delta_n}{\delta_i} & \text{if } i = n - m_c, \\[2ex] \dfrac{\mathbf{w}^{\mathrm{T}} \mathbf{v}_i}{\delta_i} & \text{otherwise,} \end{cases}$$
$$\mathbf{w} = \mathbf{w} - \mathbf{w}_i\, \tilde{t}_{i,n}.$$

Set $\hat{\mathbf{w}}_{n+m_c} = \mathbf{w}$.

(6) (Compute δ_n, check for breakdown, and update the existing auxiliary vectors.)

(6a) *Set* $\delta_n = \mathbf{w}_n^{\mathrm{T}} \mathbf{v}_n$.

(6b) *If* $\delta_n = 0$, *then stop.*
(In this case, look-ahead would be needed to continue the algorithm.)

(6c) (Biorthogonalize the vectors $\hat{\mathbf{v}}_{n+1}, \ldots, \hat{\mathbf{v}}_{n+m_c}$ against $\mathbf{w}_n$.)
For $i = n - m_c + 1, n - m_c + 2, \ldots, n$, *set*

$$t_{n,i} = \begin{cases} \dfrac{\mathbf{w}_n^{\mathrm{T}} \hat{\mathbf{v}}_{m_c+i}}{\delta_n} & \text{if } i \le 0 \text{ or } i = n, \\[2ex] \tilde{t}_{i,n}\, \dfrac{\delta_i}{\delta_n} & \text{otherwise,} \end{cases}$$
$$\hat{\mathbf{v}}_{m_c+i} = \hat{\mathbf{v}}_{m_c+i} - \mathbf{v}_n\, t_{n,i}.$$

(6d) (Biorthogonalize the vectors $\hat{\mathbf{w}}_{n+1}, \ldots, \hat{\mathbf{w}}_{n+p_c}$ against $\mathbf{v}_n$.)
For $i = n - p_c + 1, n - p_c + 2, \ldots, n$, *set*

$$\tilde{t}_{n,i} = \begin{cases} \dfrac{\hat{\mathbf{w}}_{p_c+i}^{\mathrm{T}} \mathbf{v}_n}{\delta_n} & \text{if } i \leq 0, \\ t_{i,n} \, \dfrac{\delta_i}{\delta_n} & \text{otherwise}, \end{cases}$$
$$\hat{\mathbf{w}}_{p_c+i} = \hat{\mathbf{w}}_{p_c+i} - \mathbf{w}_n \, \tilde{t}_{n,i}.$$

(7) (In the initial iterations, set up $\boldsymbol{\rho}$ and $\boldsymbol{\eta}$.)
If $n \leq m_c$, *set*

$$\rho_{n,i} = t_{n,i-m} \quad \text{for all} \quad i = n - m_c + m, \ldots, m,$$

and set $m_1 = m_c$.
If $n \leq p_c$, *set*

$$\eta_{n,i} = \tilde{t}_{n,i-p} \quad \text{for all} \quad i = n - p_c + p, \ldots, p,$$

and set $p_1 = p_c$.

Remark 5.4 The matrices $\boldsymbol{\rho}$ and $\boldsymbol{\eta}$ contain the recurrence coefficients used to biorthogonalize the starting blocks $\mathbf{R}$ and $\mathbf{L}$. More precisely, these matrices satisfy the relations

$$\mathbf{R} = \mathbf{V}_{m_1} \, \boldsymbol{\rho} \quad \text{and} \quad \mathbf{L} = \mathbf{W}_{p_1} \, \boldsymbol{\eta}.$$

5.3 Exploiting Symmetry

Recall from Section 3.3 that for linear RLC circuits, the associated transfer function $\mathbf{H}$ is symmetric and of the form (28). In this section, we describe how this symmetry can be exploited when matrix-Padé approximants $\mathbf{H}_n$ to $\mathbf{H}$ are generated via a Lanczos-type algorithm.

Let $\mathbf{C}$, $\mathbf{G}$, and $\mathbf{B}$ be the matrices in the representation (28) of $\mathbf{H}$. Note that the matrices $\mathbf{C}$ and $\mathbf{G}$ are real and symmetric. Recall that employing any Krylov-subspace technique requires a factorization (10) of the matrix $\mathbf{G} + s_0 \, \mathbf{C}$, where s_0 denotes the expansion point for the matrix-Padé approximation. For RLC circuits, meaningful expansion points are real and nonnegative, and thus we assume $s_0 \geq 0$. The matrix $\mathbf{G} + s_0 \, \mathbf{C}$ is then real and symmetric, and it admits a factorization of the form

$$\mathbf{G} + s_0 \, \mathbf{C} = \mathbf{M} \, \mathbf{J}^{-1} \, \mathbf{M}^{\mathrm{T}}, \quad \text{where} \quad \mathbf{M}, \; \mathbf{J} = \mathbf{J}^{\mathrm{T}} \in \mathbb{R}^{N \times N}. \tag{67}$$

Moreover, $\mathbf{J}$ is assumed to be a "simple" matrix, such as, e.g., a diagonal or a block-diagonal matrix. For example, for RC, RL, and LC circuits, $\mathbf{G} + s_0 \, \mathbf{C}$ is symmetric positive definite, and then one can choose $\mathbf{J} = \mathbf{I}$. If $\mathbf{G} + s_0 \, \mathbf{C}$ is symmetric indefinite, then $\mathbf{J}$ can always be chosen as a block-diagonal matrix with 1×1 and 2×2 diagonal blocks. A factorization (67) can be computed via a suitable version of the Bunch-Parlett-Kaufman algorithm if $\mathbf{G} + s_0 \, \mathbf{C}$ is indefinite, or a version of the Cholesky algorithm if $\mathbf{G} + s_0 \, \mathbf{C}$ is symmetric positive definite; see, e.g., [65].

Using (67), the transfer function (28) can be rewritten as follows:

$$\begin{aligned}\mathbf{H}(s) &= \left(\mathbf{J}\,\mathbf{M}^{-1}\,\mathbf{B}\right)^{\mathrm{T}} \left(\mathbf{I} + (s - s_0)\,\mathbf{M}^{-1}\,\mathbf{C}\,\mathbf{M}^{-\mathrm{T}}\,\mathbf{J}\right)^{-1} \left(\mathbf{M}^{-1}\,\mathbf{B}\right)\\ &= (\mathbf{J}\,\mathbf{R})^{\mathrm{T}} \left(\mathbf{I} - (s - s_0)\,\mathbf{A}\right)^{-1} \mathbf{R},\\ &\quad \text{where} \quad \mathbf{A} := -\mathbf{M}^{-1}\,\mathbf{C}\,\mathbf{M}^{-\mathrm{T}}\,\mathbf{J}, \quad \mathbf{R} := \mathbf{M}^{-1}\mathbf{B}.\end{aligned} \tag{68}$$

The matrix $\mathbf{A}$ is $\mathbf{J}$-*symmetric*, i.e., it satisfies

$$\mathbf{A}^{\mathrm{T}}\,\mathbf{J} = \mathbf{J}\,\mathbf{A}.$$

Moreover, in (68), the right and left starting blocks are $\mathbf{R}$ and $\mathbf{J}\,\mathbf{R}$, which are just $\mathbf{J}$-multiples of each other. Together with the $\mathbf{J}$-symmetry of $\mathbf{A}$, this can be used to show (see, e.g., [57]) that the right and left Lanczos vectors generated by Algorithm 5.2 are—when normalized properly—$\mathbf{J}$-multiples of each other:

$$\mathbf{w}_n = \mathbf{J}\,\mathbf{v}_n \quad \text{for all} \quad n. \tag{69}$$

Consequently, all the updates for the $\mathbf{w}_n$'s and $\hat{\mathbf{w}}_n$'s in Algorithm 5.2 can be omitted, and instead the $\mathbf{w}_n$'s are computed from the $\mathbf{v}_n$'s via (69). The resulting version of Algorithm 5.2 is called the $\mathbf{J}$-*symmetric Lanczos-type algorithm* and is applicable whenever $\mathbf{A}$ is $\mathbf{J}$-symmetric and the right and left starting blocks are connected through $\mathbf{L} = \mathbf{J}\,\mathbf{R}$. Note that, in view of (69), the general biorthogonality condition (55) reduces to

$$\mathbf{v}_i^{\mathrm{T}}\,\mathbf{J}\,\mathbf{v}_n = \begin{cases} \delta_n & \text{if } i = n, \\ 0 & \text{if } i \neq n, \end{cases} \qquad \text{for all} \quad 1 \le i, n \le N_0.$$

Thus the $\mathbf{J}$-symmetric Lanczos-type algorithm generates $\mathbf{J}$-*orthogonal* vectors $\mathbf{v}_n$, $n = 1, 2, \ldots, N_0$. Finally, the storage requirements and, provided matrix-vector products with $\mathbf{J}$ are cheap, also the computational work of the $\mathbf{J}$-symmetric Lanczos-type algorithm are only half of those of the general Algorithm 5.2. In particular, by using (69), all the matrix-vector products with $\mathbf{A}^{\mathrm{T}}$, which occur in Algorithm 5.2, have been eliminated. For RLC circuits, the products with $\mathbf{J}$ are indeed cheap, and for RC, RL, and LC circuits, there are no such products at all since $\mathbf{J} = \mathbf{I}$ for these three special cases. Therefore, by using the $\mathbf{J}$-symmetric Lanczos-type algorithm, the symmetry of the transfer function of RLC circuits can be exploited to halve the computational work and storage requirements of the procedure for general circuits.

6 Lanczos-Based Reduced-Order Modeling

In this section, we discuss the use of the Lanczos process and of some of its variants for reduced-order modeling.

6.1 The Classical Lanczos-Padé Connection

We start with the single-input single-output case. Recall that this case is characterized by the data $\mathbf{A} \in \mathbb{C}^{N\times N}$, $\mathbf{r}$, $\mathbf{l} \in \mathbb{C}^N$, and that the transfer function H is given by (33). The classical Lanczos-Padé connection [66] states that the n-th Padé approximant to H is given by

$$H_n(s) = \mathbf{l}^{\mathrm{T}}\,\mathbf{r}\,\mathbf{e}_1^{\mathrm{T}}\left(\mathbf{I} - (s - s_0)\,\mathbf{T}_n\right)^{-1}\mathbf{e}_1. \tag{70}$$

Here, $\mathbf{T}_n$ denotes the tridiagonal matrix generated by n iterations of the Lanczos Algorithm 5.1 (applied to $\mathbf{A}$, $\mathbf{r}$, $\mathbf{l}$), and $\mathbf{e}_1$ denotes the first unit vector (of length n). Furthermore, it is assumed that no exact breakdown occurs during the first n iterations of Algorithm 5.1. Clearly, the function (70) is of the required form (7). Thus in order to verify that (70) is indeed an n-th Padé approximant to H, it remains to show that the first $2n$ moments (corresponding to the expansion point s_0) of H and H_n match. By (32), the i-th moment of H is $\mathbf{l}^{\mathrm{T}}\,\mathbf{A}^i\,\mathbf{r}$. From (70), the i-th moment of H_n is $\mathbf{l}^{\mathrm{T}}\,\mathbf{r}\,\mathbf{e}_1^{\mathrm{T}}\,\mathbf{T}_n^i\,\mathbf{e}_1$. Therefore, it remains to show that

$$\mathbf{l}^{\mathrm{T}}\,\mathbf{A}^i\,\mathbf{r} = \mathbf{l}^{\mathrm{T}}\,\mathbf{r}\,\mathbf{e}_1^{\mathrm{T}}\,\mathbf{T}_n^i\,\mathbf{e}_1 \quad \text{for all} \quad i = 0, 1, \ldots, 2n-1. \tag{71}$$

Using the governing equations (57), (58), (61), and (62) of the Lanczos process, the moment-matching condition (71) can be verified by means of simple matrix calculus; see [39, 40].

The Lanczos-Padé connection can easily be extended to the more general case that look-ahead steps occur in the Lanczos process; see, e.g., [40, 67]. Next, we present a precise statement of this result.

Theorem 6.1 *Let $1 \le n \le N_0$, where N_0 is given by (54). Let $\mathbf{T}_n$ be the $n \times n$ block-tridiagonal matrix of recurrence coefficients and $\mathbf{D}_n$ be the $n \times n$ block-diagonal matrix (see Remark 5.3) generated by n iterations of the look-ahead Lanczos algorithm (applied to $\mathbf{A}$, $\mathbf{r}$, $\mathbf{l}$). Assume that $\mathbf{D}_n$ is nonsingular. (This assumption is always satisfied if no look-ahead occurs, and in the case of look-ahead, it is satisfied if n corresponds to the end of a look-ahead step.)*
Then,

$$H_n(s) = \left(\mathbf{D}_n^{\mathrm{T}}\,\mathbf{e}_1\,\eta\right)\left(\mathbf{I} - (s - s_0)\,\mathbf{T}_n\right)^{-1}(\mathbf{e}_1\,\rho), \tag{72}$$

where $\rho := \|\mathbf{r}\|$ and $\eta := \|\mathbf{l}\|$, is the unique n-th Padé approximant to the transfer function $H(s) = \mathbf{l}^{\mathrm{T}}\left(\mathbf{I} - (s - s_0)\,\mathbf{A}\right)^{-1}\mathbf{r}$.

Feldmann and Freund [39, 40] and, almost simultaneously, Gallivan, Grimme, and Van Dooren [59] were the first to introduce the circuit-simulation community to the Lanczos-Padé connection, and to advocate its use instead of the mathematically equivalent, but numerically unstable AWE method. Computing Padé approximants of transfer functions of single-input single-output linear dynamical systems via (72) has become known as the **PVL** (**P**adé **V**ia **L**anczos) method, which is the name proposed in [39, 40]. For details on PVL and further issues, such as the choice of the expansion point s_0 for the Padé approximation, we refer the reader to [40].

Next, we present an example (taken from [39, 40]) that illustrates the potential of Padé-based reduced-order modeling in circuit simulation. The example is the so-called PEEC circuit [98], which has become a popular test circuit for reduced-order modeling techniques. The PEEC circuit has 2100 capacitors, 172 inductors, and 6990 inductive couplings. It

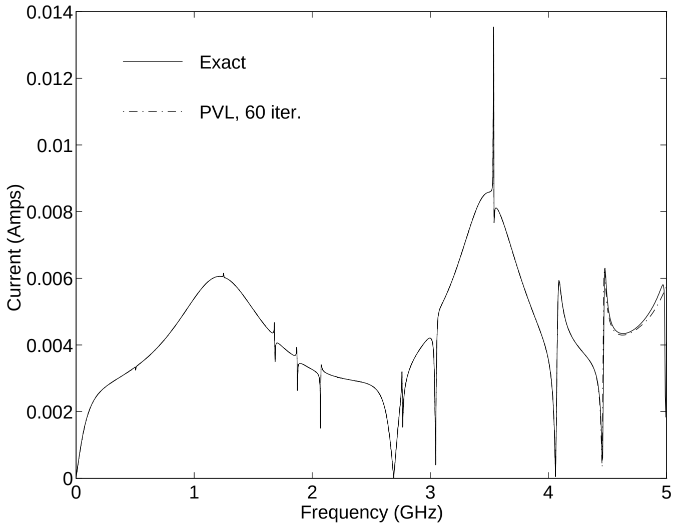

Figure 3: Results for the PEEC circuit, 60 PVL iterations.

turns out that the Padé approximant H_{60} of order $n = 60$ is sufficient to recover the exact transfer function H in the frequency range of interest, $0 \leq \omega \leq 5 \times 10^9$. In Figure 3, we show the approximation $|H_{60}(\mathrm{j}\,\omega)|$ (computed with PVL), together with the exact function $|H(\mathrm{j}\,\omega)|$, both for $0 \leq \omega \leq 5 \times 10^9$.

We conclude this subsection by mentioning some further work related to PVL. In [50], it is shown how PVL can be extended to include the computation of circuit sensitivities. Bai and Ye [10] developed a technique for estimating the approximation error in PVL. Recall from Sections 3.3 and 5.3 that, for RLC circuits, the inherent symmetry of the transfer functions can be exploited by employing a suitable $\mathbf{J}$-symmetric variant of the classical Lanczos algorithm. For single-input single-output RLC circuits, such a symmetric variant of PVL, the SyPVL method, was introduced in [51]. Finally, it is also possible to use PVL in simulations that include distributed circuit elements such as transmission lines. For example, Celik and Cangellaris [21, 22] developed a PVL-based technique for the simulation of multiconductor transmission lines.

6.2 The Multi-Input Multi-Output Case

We now return to the general multi-input multi-output case. Recall that, in this case, $\mathbf{H}\colon \mathbb{C} \mapsto (\mathbb{C} \cup \{\infty\})^{p \times m}$ is a matrix-valued function. Moreover, in view of (11), $\mathbf{H}$ can be characterized by the three matrices $\mathbf{A} \in \mathbb{C}^{N \times N}$, $\mathbf{R} \in \mathbb{C}^{N \times m}$, and $\mathbf{L} \in \mathbb{C}^{N \times p}$.

A trivial approach to using Padé approximation in the multi-input multi-output case would be to compute scalar Padé approximants for each of the $m \cdot p$ components of $\mathbf{H}$ by means of $m \cdot p$ PVL runs. However, such a "scalar" approach is not competitive with reduced-order modeling based on matrix-Padé approximants $\mathbf{H}_n$, as defined by (8) and (9). First, to obtain reduced-order models of comparable accuracy, the scalar approach requires at least $((m + p)/2)$-times more computational work than the "matrix" approach based on

$\mathbf{H}_n$. Second, the scalar approach results in reduced-order models with bigger state-space dimension than those obtained with the matrix approach.

It turns out that matrix-Padé approximants can be computed from the quantities produced by the Lanczos-type process for multiple starting vectors described in Section 5.2. An exact connection between this Lanczos-type process and matrix-Padé approximants $\mathbf{H}_n$, $n \geq 1$, only holds for exact deflation. Therefore, for the remainder of this subsection, we assume that only exact deflation is performed.

First, we introduce some notation that will be needed to state our main result. We denote by m_j (respectively p_k) the number of columns of the matrix $\mathbf{R}_j$ (respectively $\mathbf{L}_k$) in the definition (51) of the deflated block Krylov matrices $\mathbf{K}^{\mathrm{dl}}(\mathbf{A}, \mathbf{R})$ and $\mathbf{K}^{\mathrm{dl}}(\mathbf{A}^{\mathrm{T}}, \mathbf{L})$. We remark that

$$m \geq m_1 \geq m_2 \geq \cdots \geq m_{j_{\max}} \geq 1, \quad p \geq p_1 \geq p_2 \geq \cdots \geq p_{k_{\max}} \geq 1,$$

and that $m_{j-1} > m_j$, respectively $p_{k-1} > p_k$, indicates the occurrence of at least one exact deflation. Let N_0 be the integer defined in (64). For each integer n with $\max\{m_1, p_1\} \leq n \leq N_0$, we define $j = j(n)$ and $k = k(n)$ as the maximal integers such that

$$m_1 + m_2 + \cdots + m_j \leq n \quad \textit{and} \quad p_1 + p_2 + \cdots + p_k \leq n, \tag{73}$$

respectively. Note that if no deflation occurs, then

$$j(n) = \left\lfloor \frac{n}{m} \right\rfloor \quad \text{and} \quad k(n) = \left\lfloor \frac{n}{p} \right\rfloor.$$

We denote by $\mathbf{T}_n \in \mathbb{C}^{n \times n}$, $\boldsymbol{\rho} \in \mathbb{C}^{m_1 \times m}$, and $\boldsymbol{\eta} \in \mathbb{C}^{p_1 \times p}$, the matrices generated by n $\big(\geq \max\{m_1, p_1\}\big)$ steps of the Lanczos-type Algorithm 5.2 (or a look-ahead variant thereof) applied to $\mathbf{A}$, $\mathbf{R}$, and $\mathbf{L}$. Moreover, $\mathbf{D}_n$ $\big(= \mathbf{W}_n^{\mathrm{T}} \mathbf{V}_n\big)$ is the corresponding matrix describing the biorthogonality of the Lanczos vectors. In the case of no look-ahead, $\mathbf{D}_n$ is diagonal and nonsingular. In the case of look-ahead, $\mathbf{D}_n$ is block-diagonal, and it is singular if, and only if, an exact breakdown occurs at step n. After these preliminaries, we can now state the following theorem.

Theorem 6.2 *Let* $\max\{m_1, p_1\} \leq n \leq N_0$. *Assume that* $\mathbf{D}_n$ *is nonsingular.* (*This assumption is always satisfied if no look-ahead occurs, and in the case of look-ahead, it is satisfied if* n *corresponds to the end of a look-ahead step.*) *Let* $j = j(n)$ *and* $k = k(n)$ *be the integers defined in* (73).
Then,

$$\begin{aligned} &\mathbf{H}_n(s) = \boldsymbol{\eta}_n^{\mathrm{T}} \left(\mathbf{I} - (s - s_0)\, \mathbf{T}_n\right)^{-1} \boldsymbol{\rho}_n, \\ &\textit{where} \quad \boldsymbol{\rho}_n := \begin{bmatrix} \boldsymbol{\rho} \\ \mathbf{0}_{(n-m_1)\times m} \end{bmatrix} \quad \textit{and} \quad \boldsymbol{\eta}_n := \mathbf{D}_n^{\mathrm{T}} \begin{bmatrix} \boldsymbol{\eta} \\ \mathbf{0}_{(n-p_1)\times p} \end{bmatrix}, \end{aligned} \tag{74}$$

is an n*-th matrix-Padé approximant of the function* $\mathbf{H}$ *defined in* (11), *and*

$$\mathbf{H}_n(s) = \mathbf{H}(s) + \mathcal{O}\left((s - s_0)^{j+k}\right).$$

Moreover, if $n = m_1 + m_2 + \cdots + m_j$ *or* $n = p_1 + p_2 + \cdots + p_k$, *then* $\mathbf{H}_n$ *is a matrix-Padé approximant of minimal order* n *that matches the first* $j + k$ *moments* (12), $\mathbf{M}_0, \mathbf{M}_1, \ldots, \mathbf{M}_{j+k-1}$, *of* $\mathbf{H}$.

Remark 6.1 This theorem gives an extension of the classical Padé-Lanczos connection to the matrix case. Indeed, for the special case $m = p = 1$, formula (74) reduces to (72). Theorem 6.2 appeared first in [46], and for a proof, we refer the reader to [46, 49]. To the best of our knowledge, Theorem 6.2 is the only connection between matrix-Padé and a Lanczos-type process that has appeared in the literature.

Remark 6.2 The representation (74) of $\mathbf{H}_n$ is of the form (7) with

$$\mathbf{C}_n := -\mathbf{T}_n, \quad \mathbf{G}_n := \mathbf{I} + s_0\,\mathbf{T}_n, \quad \mathbf{B}_n := \boldsymbol{\rho}_n, \quad \mathbf{E}_n := \boldsymbol{\eta}_n. \tag{75}$$

In particular, the linear dynamical system (5) with the matrices $\mathbf{C}_n$, $\mathbf{G}_n$, $\mathbf{B}_n$, and $\mathbf{E}_n$ defined in (75) is a reduced-order model whose transfer function is just $\mathbf{H}_n$.

Remark 6.3 Instead of (74), the matrix-valued rational function $\mathbf{H}_n$ can always be written in terms of a pair of matrix polynomials, in analogy to the representation (38) of H_n in the scalar case $m = p = 1$. However, since matrix polynomials do not commute and also for other reasons, such a representation involves a number of difficulties (see the discussion in [123]) that can be avoided by using (74).

Remark 6.4 The problem of matrix-Padé approximation of transfer functions of linear dynamical systems is mathematically equivalent to the minimal partial realization problem; see, e.g., [14, 15, 47, 63, 115]. In fact, this connection is used in the proof of Theorem 6.2 in [46, 49]. More precisely, $\mathbf{H}_n$ is verified to be a matrix-Padé approximant by checking the criterion [63, Theorem 0.1] for minimality of a given partial realization. Furthermore, Theorem 6.2 can be directly translated into a corresponding result for minimal partial realizations, see [47].

The connection (74) is the basis of the **MPVL** (**M**atrix-**P**adé **V**ia **L**anczos) method that was developed by Freund and Feldmann in [41]. To the best of our knowledge, MPVL was the first algorithm to use a Lanczos-type procedure to compute matrix-Padé-based reduced-order models of general multi-input multi-output linear dynamical systems. For p-terminal RLC circuits, a special MPVL-variant, called SyMPVL, was proposed in [43, 52]. The SyMPVL algorithm exploits the symmetry of the transfer functions of RLC circuits (see Section 3.3) by using the $\mathbf{J}$-symmetric Lanczos-type procedure sketched in Section 5.3, instead of the general Algorithm 5.2. In particular, SyMPVL for RLC circuits does not involve any matrix-vector products with $\mathbf{A}^{\mathrm{T}}$, as MPVL for general circuits does, and as a result, SyMPVL requires only half the computational work of MPVL.

Finally, we would like to mention the earlier work of Meyer and Pal [84] who used Ruhe's Hermitian band-Lanczos method [101] to compute matrix elements of the resolvent of a Hermitian matrix. In some sense, SyMPVL can be viewed as an extension of the algorithm in [84] to the more general $\mathbf{J}$-symmetric case.

6.3 Stability and Passivity

It is desirable and often crucial that reduced-order models inherit the essential properties of the original linear dynamical system. One such crucial property is *stability*; see, e.g., [4, Section 3.7]. A linear dynamical system (1) is said to be *stable* if its associated transfer function (4), $\mathbf{H}$, has no poles in

$$\mathbb{C}_+ := \{\, s \in \mathbb{C} \mid \operatorname{Re} s > 0 \,\}$$

(the open right half of the complex plane) and if any purely imaginary poles of $\mathbf{H}$ are simple. Any linear dynamical system that describes an actual physical system, such as a functioning electronic circuit, is stable.

For linear dynamical systems with $m = p$ inputs and outputs, a second important property is *passivity*; see, e.g., [4, 122]. In particular, linear p-terminal RLC circuits (see Section 3.3) are always passive. Roughly speaking, a linear dynamical system is passive if it does not generate energy. In terms of the transfer function $\mathbf{H}$, passivity means that $\mathbf{H}$ is a positive real function. More precisely, a linear dynamical system (1) with $m = p$ is said to be *passive* if its associated transfer function (4), $\mathbf{H}$, satisfies the following three conditions:

(i) $\mathbf{H}(s)$ has no poles in $\mathbb{C}_+$;

(ii) $\mathbf{H}(\bar{s}) = \overline{\mathbf{H}(s)}$ for all $s \in \mathbb{C}$;

(iii) $\operatorname{Re}\left(\mathbf{x}^{\mathrm{H}}\mathbf{H}(s)\,\mathbf{x}\right) \geq 0$ for all $s \in \mathbb{C}_+$ and $\mathbf{x} \in \mathbb{C}^p$.

In view of (i), passive linear dynamical systems are in particular stable. Furthermore, property (iii) implies that $\mathbf{H}$ has no zeros in $\mathbb{C}_+$. The condition that $\mathbf{H}$ has neither poles nor zeros in $\mathbb{C}_+$, however, is only necessary, and in general not sufficient, for the passivity of a linear dynamical system.

A well-known disadvantage of reduced-order modeling based on Padé approximants $\mathbf{H}_n$ to $\mathbf{H}$ is that, in general, $\mathbf{H}_n$ may not be stable even though the original linear dynamical system was stable; see, e.g., [1, 3, 109, 110]. Similarly, for passive systems, passivity of $\mathbf{H}_n$ cannot be guaranteed in general. If the reduced-order model is used only for simulation purposes (for example, to validate the desired frequency response of a circuit design), then the model not being stable or passive is not an issue at all, and thus nothing needs to be done. In other situations, stability or passivity of the reduced-order model are crucial. For example, if the reduced-order model of an RLC circuit needs to be synthesized as a smaller RLC circuit, then such a synthesis is only possible if the reduced-order model is stable and passive. In such cases, it needs to be checked if $\mathbf{H}_n$ is stable and passive, and if it is not, then the reduced-order model needs to be modified to make it stable and passive. In Sections 6.4 and 6.5, we will briefly describe two such modification techniques.

For the important special cases of RC, RL, and LC circuits, stability and passivity of the Padé approximants $\mathbf{H}_n$ can always be guaranteed, at least in theory. However, in order to achieve this stability and passivity even in practice, a proper implementation is required; see Section 6.6. The passivity of Padé-based reduced-order models for RC, RL, and LC circuits was noted by Bracken [16] who also established a connection to network synthesis. The stability and passivity of the SyPVL and SyMPVL methods, when applied to RC, RL, and LC circuits, was proved in [43, 51, 52]. The proof uses the fact that, by (29) and (30), the circuit matrices $\mathbf{C}$ and $\mathbf{G}$ are symmetric positive semi-definite. As a result, the matrix $\mathbf{A}$ to which the symmetric Lanczos or Lanczos-type process is applied is symmetric negative semi-definite. At least in exact arithmetic, this guarantees that the Lanczos matrix $\mathbf{T}_n$ is also symmetric negative semi-definite:

$$-\mathbf{T}_n \geq \mathbf{0}. \tag{76}$$

With (76), it is then readily verified that $\mathbf{H}_n$ is indeed stable and passive for RC, RL, and LC circuits.

For RC circuits, a different Padé-based reduced-order modeling technique, the PACT algorithm, was proposed in [75, 77]. Here, one block of circuit variables is first eliminated in the transfer function. The result is a smaller state-space dimension, but a more complicated function of s that is then replaced by a suitable Padé approximant. Using the concept of congruence transformations, it is proved that PACT generates passive reduced-order models. In [76], an extension to RLC circuits, based on split congruence transformations, is studied.

6.4 PVLπ: Post-Processing of PVL

In [6], Bai, Feldmann, and Freund proposed a post-processing technique for PVL that, if necessary, modifies the n-th Padé approximant H_n to make it stable and, if the original circuit is passive, also passive. The resulting variant of PVL is called PVLπ.

We now briefly describe the main idea behind PVLπ. For simplicity, we consider only the case of PVL based on the classical Lanczos Algorithm 5.1 without look-ahead. Recall that the n-th reduced-order transfer function H_n is given by (70). First, one checks the stability of H_n. This requires the computation of the poles p_i, $i = 1, 2, \ldots, n$, of H_n. By (70), the poles are given by

$$p_i = s_0 + \frac{1}{\lambda_i\left(\mathbf{T}_n\right)}, \quad i = 1, 2, \ldots, n, \tag{77}$$

where the $\lambda_i\left(\mathbf{T}_n\right)$'s are the eigenvalues of $\mathbf{T}_n$. If none of the poles (77) lie in $\mathbb{C}_+$ and if possible purely imaginary poles are simple, then the reduced-order model defined by H_n is stable. Otherwise, some of the poles (77) violate the stability condition; we call these the "unstable" poles. If the original circuit is passive, then we also check if H_n has no zeros in $\mathbb{C}_+$; recall from Section 6.3 that this is a necessary condition for passivity. The zeros of H_n are given by

$$z_i = s_0 + \frac{1}{\lambda_i\left(\mathbf{T}'_n\right)}, \quad i = 1, 2, \ldots, n-1, \tag{78}$$

where $\mathbf{T}'_n$ is the $(n-1) \times (n-1)$ matrix obtained from $\mathbf{T}_n$ by deleting the first row and the first column. If none of the zeros (78) lie in $\mathbb{C}_+$, then the reduced-order model defined by H_n satisfies the necessary condition for passivity, and we proceed to check the necessary and sufficient passivity criterion developed in [8], which requires the solution of a generalized eigenvalue problem of size n. Otherwise, some of the zeros (78) violate the necessary passivity condition; we call these the "unstable" zeros.

Suppose that we found l_1 unstable poles and l_2 unstable zeros, and set $l := l_1 + l_2$. If H_n is a sufficiently good approximation over the frequency range of interest, then typically $l \ll n$. In particular, we now assume that $l \leq n$. By appropriately modifying the last column of the Lanczos matrix $\mathbf{T}_n$ in (70), one can construct a new reduced-order transfer function,

$$\widetilde{H}_n(s) := \mathbf{l}^{\mathrm{T}}\,\mathbf{r}\,\mathbf{e}_1^{\mathrm{T}}\left(\mathbf{I} - (s - s_0)\,\widetilde{\mathbf{T}}_n\right)^{-1}\mathbf{e}_1, \tag{79}$$

that has any set of l_1 prescribed poles and any set of l_2 prescribed zeros, and that at the same time still matches the first $2n - l$ moments of H:

$$\widetilde{H}_n(s) = H(s) + \mathcal{O}\left(s - s_0\right)^{2n-l}. \tag{80}$$

More precisely, $\widetilde{\mathbf{T}}_n$ is obtained from $\mathbf{T}_n$ via a rank-one update of the form

$$\widetilde{\mathbf{T}}_n = \mathbf{T}_n + \mathbf{t}\,\mathbf{e}_n^{\mathrm{T}}, \tag{81}$$

where $\mathbf{t} \in \mathbb{C}^n$ is a vector whose first $n - l$ entries are zero, and $\mathbf{e}_n$ denotes the n-th unit vector of length n. We remark that this modification technique generalizes earlier results by Golub, Elhay, and Kautsky [64, 74] on the computation of certain Gauss-type quadrature rules. Note that, in analogy to the representation (38) of H_n, the function (79) can be expressed in terms of polynomials as follows:

$$\widetilde{H}_n(s) = \frac{\varphi_{n-1-l_2}(s)}{\psi_{n-l_1}(s)}\,\frac{\widetilde{\varphi}_{l_2}(s)}{\widetilde{\psi}_{l_1}(s)}. \tag{82}$$

Here, $\widetilde{\psi}_{l_1}$ and $\widetilde{\varphi}_{l_2}$ are fixed polynomials of degree l_1 and l_2 whose zeros are just the l_1 prescribed poles and l_2 prescribed zeros, respectively, of $\widetilde{H}_n$. Furthermore, φ_{n-1-l_2} and ψ_{n-l_1} are polynomials of degree at most $n - 1 - l_2$ and $n - l_1$, respectively, which are described by a total of $2n - l$ free parameters. In view of (80), these parameters are chosen such that the first $2n - l$ moments of $\widetilde{H}_n$ and H match. A rational function of the form (82) with a total of l prescribed poles and zeros is called an *n-th partial Padé approximant* to H if it satisfies (80); see [17].

The next step then is to select a set of l_1 prescribed poles and a set of l_2 prescribed zeros in the left half of the complex plane, and to determine the vector $\mathbf{t}$ in (81) such that the modified reduced-order transfer function (79) has these l prescribed numbers as poles and zeros. One possible strategy for the choice of these prescribed numbers is to simply reflect the unstable poles and zeros with respect to the imaginary axis. Of course, in general, it cannot be guaranteed that all n poles and all $n - 1$ poles of $\widetilde{H}_n$ are stable, since the transition from H_n to $\widetilde{H}_n$ may have moved some of the stable poles and zeros of H_n into $\mathbb{C}_+$. However, in our experience, this rarely happens. Roughly speaking, the reason is that, in view of (80), the modified transfer function $\widetilde{H}_n$ has almost the same approximation properties as the original Padé approximant H_n, especially when $l \ll n$. As a result, the critical stable poles and zeros close to the imaginary axis are essentially unchanged when H_n is modified to $\widetilde{H}_n$.

Finally, we present one example to illustrate the typical behavior of PVLπ. The example arises in the analysis of a 64-pin package model used for an RF integrated circuit. Only eight of the package pins carry signals, the rest being either unused or carrying supply voltages. The package model is described by approximatively 4000 circuit elements, resistors,

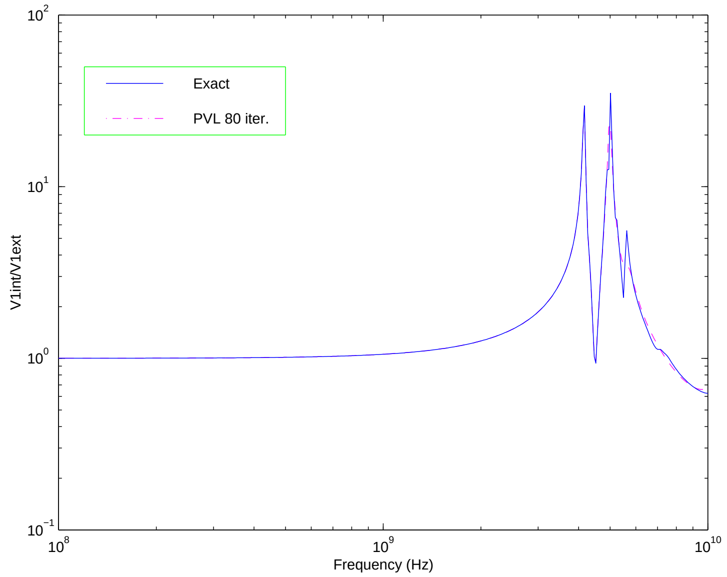

Figure 4: PVL approximation for package example.

capacitors, inductors, and inductive couplings. Here, we compute the input impedance of one of the pins. After executing the PVL algorithm for $n = 80$ iterations, the Padé approximant H_{80} is sufficiently accurate up to 10GHz, but H_{80} has two unstable poles and four unstable zeros. Figure 4 shows the frequency response of the exact transfer function and the approximation H_{80} as computed by PVL. Next, we applied the PVLπ algorithm to move the two unstable poles and four unstable zeros into the left half of the complex plane. Figure 5 shows the frequency response of the exact transfer function and the modified approximation $\widetilde{H}_{80}$ produced by PVLπ. The reduced-order model characterized by $\widetilde{H}_{80}$ is stable and passive. Figures 4 and 5 clearly illustrate that no accuracy has been lost by enforcing stability and passivity by means of our post-processing procedure.

6.5 Passive Reduced-Order Models from SyMPVL

Recently, Odabasioglu [88] developed an Arnoldi-based reduced-order modeling technique, the PRIMA algorithm; see also [89]. In PRIMA, one first runs a block Arnoldi algorithm (applied to a single matrix), and then projects the circuit matrices onto the subspace spanned by the Arnoldi vectors. In [88, 89], it is proved that, at least in theory, PRIMA produces passive reduced-order models when applied to p-terminal RLC circuits. Furthermore, in [88, 89], it is shown that the associated reduced-order transfer function still matches half as many moments as the n-th Padé approximant $\mathbf{H}_n$, which is the reduced-order transfer function generated by SyMPVL.

It turns out that this moment-matching property of PRIMA is not a characteristic of the underlying block Arnoldi process, and in fact, it is true for any method that generates basis vectors for the corresponding block Krylov subspaces. In particular, in [48], it is shown that the projection idea in PRIMA can also be combined with the Lanczos-based SyMPVL procedure to generate provably passive reduced-order models for p-terminal RLC circuits. The

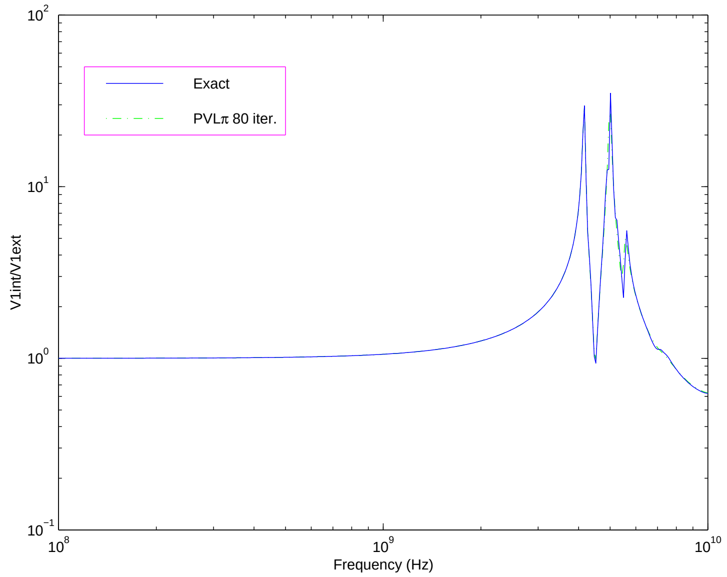

Figure 5: PVLπ approximation for package example.

resulting algorithm, called SyMPVLplus in [48], has the advantage over PRIMA that, for each n, it produces two reduced-order models: the SyMPVL reduced-order model associated with the n-th matrix Padé approximant $\mathbf{H}_n$, and a second, provably passive, reduced-order model associated with a second approximation $\mathbf{H}_n^{(\mathrm{p})}$. Here, $\mathbf{H}_n$ is of maximal accuracy in the sense that $\mathbf{H}_n$ matches the maximal number of moments, and $\mathbf{H}_n^{(\mathrm{p})}$ still matches half as many moments, just as the PRIMA reduced-order model. Furthermore, SyMPVLplus and PRIMA require exactly the same number of matrix-vector products. The remaining computational work is mainly vector operations, and here SyMPVLplus even requires less work than PRIMA due to the fact that, in contrast to the Arnoldi process, the Lanczos-type method involves only short vector recurrences.

Next, we sketch how $\mathbf{H}_n^{(\mathrm{p})}$ is generated in SyMPVLplus; for details, we refer the reader to [48]. Recall that for p-terminal RLC circuits, the transfer function $\mathbf{H}$ is given by (26)–(28). In view of the block structure of the matrices $\mathbf{G}$, $\mathbf{C}$, and $\mathbf{B}$, in (26) and (27), the representation (28) can be rewritten as follows:

$$\begin{aligned} &\mathbf{H}(s) = \mathbf{B}^{\mathrm{T}} \left(\widetilde{\mathbf{G}} + s\,\widetilde{\mathbf{C}}\right)^{-1} \mathbf{B}, \\ &\text{where} \quad \widetilde{\mathbf{G}} = \begin{bmatrix} \mathbf{G}_{11} & \mathbf{G}_{12} \\ -\mathbf{G}_{12}^{\mathrm{T}} & \mathbf{0} \end{bmatrix} \quad \text{and} \quad \widetilde{\mathbf{C}} = \widetilde{\mathbf{C}}^{\mathrm{T}} = \begin{bmatrix} \mathbf{C}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{22} \end{bmatrix}. \end{aligned} \tag{83}$$

Now let $\mathbf{V}_n$ be the matrix of Lanczos vectors generated by the $\mathbf{J}$-symmetric Lanczos-type algorithm underlying SyMPVL. Finally, in analogy to the projection used in PRIMA, one sets

$$\begin{aligned} &\mathbf{H}_n^{(\mathrm{p})}(s) := \mathbf{B}_n^{\mathrm{T}} \left(\mathbf{G}_n + s\,\mathbf{C}_n\right)^{-1} \mathbf{B}_n, \\ &\text{where} \quad \mathbf{B}_n := \mathbf{V}_n^{\mathrm{T}} \mathbf{B}, \quad \mathbf{G}_n := \mathbf{V}_n^{\mathrm{T}} \widetilde{\mathbf{G}}\, \mathbf{V}_n, \quad \mathbf{C}_n := \mathbf{V}_n^{\mathrm{T}} \widetilde{\mathbf{C}}\, \mathbf{V}_n. \end{aligned} \tag{84}$$

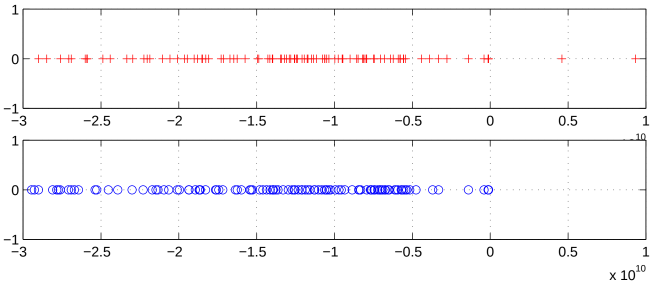

Figure 6: Poles: non-coupled (upper figure) vs. coupled (lower figure).

Using the structure of the matrices $\widetilde{\mathbf{G}}$ and $\widetilde{\mathbf{C}}$ in (83), it is relatively straightforward to verify that the function (84) is positive real. Hence, the reduced-order model associated with (84) is passive. Using similar techniques as in [88, 89] one can also show that $\mathbf{H}_n^{(\mathrm{p})}$ matches half as many moments as $\mathbf{H}_n$; see [48].

6.6 How to Achieve Passivity in Practice

As we mentioned in Section 6.3, for RC, RL, and LC circuits, the reduced-order models generated by SyPVL and SyMPVL are provably stable and passive, at least in exact arithmetic. However, when implemented in finite-precision arithmetic, it may actually occur that the computed model is unstable or non-passive. In the upper part of Figure 6, we show the computed poles that were obtained after $n = 300$ steps of SyMPVL applied to a very large RC circuit consisting of over 200,000 resistors and capacitors. Clearly, a few of the poles are positive and thus the associated reduced-order model is unstable and non-passive.

The reason for unstable and non-passive models is that the positive semi-definiteness (76) of $-\mathbf{T}_n$, which is the crucial property that implies stability and passivity, is not guaranteed in finite-precision arithmetic, at least not when conventional, non-coupled, Lanczos recurrences are employed. The simple remedy is to replace the Lanczos-type algorithm in SyMPVL by a mathematically equivalent procedure that is based on suitably coupled Lanczos recurrences. Such a coupled Lanczos-type algorithm was developed in [9], and combined with SyMPVL in [7]. Using the same compact matrix form as in (65), the coupled recurrences of this Lanczos-type algorithm (for symmetric negative semi-definite matrices $\mathbf{A}$ and one block of p starting vectors $\mathbf{R}$) can be stated as follows:

$$\begin{aligned} \mathbf{A}\,\mathbf{P}_n &= -\mathbf{V}_n\,\mathbf{L}_n + [\,\underbrace{0 \quad \cdots \quad 0}_{n-p_c} \;\; \underbrace{\hat{\mathbf{v}}_{n+1} \quad \cdots \quad \hat{\mathbf{v}}_{n+p_c}}_{p_c}\,] + \mathbf{V}_{\mathrm{defl}}, \\ \mathbf{V}_n &= \mathbf{P}_n\,\mathbf{L}_n^{\mathrm{T}}. \end{aligned} \tag{85}$$

Here, $\mathbf{V}_n$ is the matrix of Lanczos vectors that span $\mathcal{K}_n(\mathbf{A}, \mathbf{R})$, $\mathbf{P}_n$ is a matrix of a second set of vectors that also span $\mathcal{K}_n(\mathbf{A}, \mathbf{R})$, and $\mathbf{L}_n$ is an $n \times n$ lower triangular matrix that contains the coefficients of the coupled recurrences. In the case of exact deflation only, $\mathbf{L}_n$ is banded with bandwidth $p+1$. In terms of $\mathbf{L}_n$, the crucial matrix $-\mathbf{T}_n$, which defines the reduced-order model, is now given by

$$-\mathbf{T}_n := \mathbf{L}_n\,\mathbf{L}_n^{\mathrm{T}}. \tag{86}$$

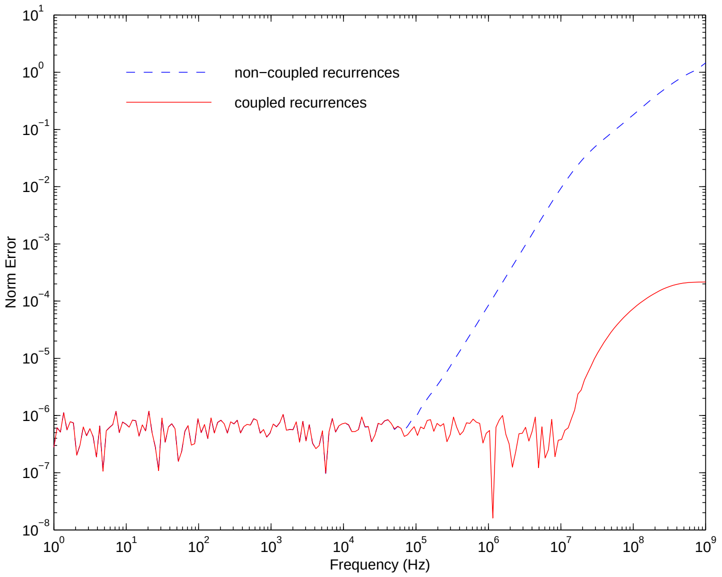

Figure 7: Approximation error: non-coupled vs. coupled.

The point of using coupled recurrences is to actually generate $\mathbf{L}_n$ and to then compute $\mathbf{T}_n$ from $\mathbf{L}_n$ via (86). This guarantees that $-\mathbf{T}_n$ is positive semi-definite.

In the lower part of Figure 6, we show the computed poles that were obtained after $n = 300$ steps of SyMPVL based on the coupled recurrences (85), applied to the same large RC circuit. Clearly, all the computed poles are negative now, and the reduced-order model is stable and passive.

Another desirable by-product of coupled recurrences is that, for the same n, the reduced-order models computed by means of coupled recurrences are typically more accurate for high frequencies than those generated by means of non-coupled recurrences. This point is illustrated in Figure 7, where we compare the error curves $\|\mathbf{H}(\mathrm{j}\,\omega) - \mathbf{H}_{\mathrm{n}}(\mathrm{j}\,\omega)\|$ obtained after the same number of n SyMPVL iterations. Clearly, the reduced-order model computed via coupled recurrences is more accurate for high frequencies.

Finally, we remark that, in a different context, the benefit of using coupled, instead of non-coupled, Lanczos recurrences was also noted and exploited in [56]. There, a coupled version of the QMR method [55] for the solution of systems of linear equations was proposed.

6.7 Two Other Lanczos-Based Approaches

In this subsection, we briefly describe two further reduced-order modeling techniques that are based on extensions of the Lanczos process.

For the single-input single-output case, Grimme, Sorensen, and Van Dooren [68] proposed to use an *implicitly restarted Lanczos algorithm* to remedy the possible instability of Padé-based reduced-order models. The basic idea here is to adaptively choose a polynomial χ that damps unstable poles of the reduced-order model, and then implicitly run n steps of the Lanczos algorithm applied to $\mathbf{A}$ with modified starting vectors $\chi(\mathbf{A})\,\mathbf{r}$ and $\chi(\mathbf{A}^{\mathrm{T}})\,\mathbf{l}$.

This results in some Padé-type approximation

$$\begin{aligned}
H(s) &= \mathbf{l}^{\mathrm{T}} \left(\mathbf{I} - (s - s_0)\,\mathbf{A}\right)^{-1} \mathbf{r} \\
&\approx \left(\chi(\mathbf{A}^{\mathrm{T}})\,\mathbf{l}\right)^{\mathrm{T}} \left(\mathbf{I} - (s - s_0)\,\mathbf{A}\right)^{-1} \left(\chi(\mathbf{A})\,\mathbf{r}\right) \\
&\approx \mathbf{l}^{\mathrm{T}}\,\mathbf{r}\,\mathbf{e}_1^{\mathrm{T}} \left(\mathbf{I} - (s - s_0)\,\mathbf{T}_n\right)^{-1} \mathbf{e}_1 =: \widehat{H}_n(s),
\end{aligned}$$

where H and $\widehat{H}_n$ match in the $2n$ coefficients

$$\mathbf{l}^{\mathrm{T}}\,\mathbf{A}^i\,(\chi(\mathbf{A}))^2\,\mathbf{r}, \quad i = 0, 1, \ldots, 2n-1. \tag{87}$$

With the same amount of computational effort, PVL would generate a true Padé approximation that matches the $2(n + \deg\chi)$ moments

$$m_i = \mathbf{l}^{\mathrm{T}}\,\mathbf{A}^i\,\mathbf{r}, \quad i = 0, 1, \ldots, 2(l + \deg\chi) - 1. \tag{88}$$

Note that the coefficients (87) contain only part of the information that is contained in the moments (88). It remains unclear if matching only the coefficients (87) still results in a meaningful reduced-order model.

Instead of matching moments only at one expansion point, s_0, so-called *multi-point Padé approximants* (see, e.g., [11]) match moments at several, say M, expansion points, $s_0, s_1, \ldots, s_{M-1}$. A multi-point moment-matching algorithm based on AWE-type computations was proposed in [23]. Gallivan, Grimme, and Van Dooren [60], for the single-input single-output case, advocated the use of multi-point Padé approximation for reduced-order modeling and showed how such approximants can be computed by means of the so-called *rational Lanczos method* [103, 104]. Nguyen and Li [86] developed a multi-point Padé approach for the special case of $m = p$ inputs and outputs.

One clear disadvantage of reduced-order modeling based on multi-point Padé approximation is that now M sparse LU factorizations,

$$\mathbf{G} + s_i\,\mathbf{C}, \quad i = 0, 1, \ldots, M-1,$$

need to be computed, instead of the single factorization (25) for standard Padé or matrix-Padé approximation. For reduced-order modeling in circuit simulation, the cost of the LU factorization (25) usually dominates the other computations. Therefore, having to compute M such factorization increases the computational costs significantly, and it is not obvious that multi-point Padé approximants are so much better than standard Padé approximants to compensate for the $M-1$ additional factorizations. Another nontrivial issue is the choice of the M expansion points. Finally, for the important special case of RC circuits with a tree or tree-like structure, the factorization (25) can often be avoided altogether by choosing $s_0 = 0$ and using (31); see Section 3.3. Clearly, multi-point Padé techniques cannot take advantage of this special property.

7 The Arnoldi Process

Recall from Section 5.1 that the classical Lanczos algorithm generates two sequences of biorthogonal basis vectors for the Krylov subspaces $\mathcal{K}_n(\mathbf{A}, \mathbf{r})$ and $\mathcal{K}(\mathbf{A}^{\mathrm{T}}, \mathbf{l})$ induced by the data $\mathbf{A} \in \mathbb{C}^{N \times N}$ and $\mathbf{r},\ \mathbf{l} \in \mathbb{C}^N$.

Instead of two sequences, the Arnoldi process [5] (applied to $\mathbf{A}$ and $\mathbf{r}$) produces only one sequence of vectors,

$$\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n, \ldots, \mathbf{v}_{n_r}, \tag{89}$$

the so-called *Arnoldi vectors.* In (89), n_r is the integer defined in (36). The first n Arnoldi vectors span the Krylov subspace $\mathcal{K}_n(\mathbf{A}, \mathbf{r})$, i.e., the vectors (89) satisfy (46) for all $n = 1, 2, \ldots, n_r$. Moreover, the Arnoldi vectors are constructed to be orthonormal:

$$\mathbf{v}_i^{\mathrm{H}} \mathbf{v}_n = \begin{cases} 1 & \text{if } i = n, \\ 0 & \text{if } i \neq n, \end{cases} \qquad \text{for all} \quad 1 \leq i, n \leq n_r.$$

From a numerical point of view, orthonormality is a very desirable property, and in this sense, the Arnoldi process produces numerically "optimal" basis vectors for the Krylov subspaces $\mathcal{K}_n(\mathbf{A}, \mathbf{r})$. More precisely, for each n, the matrix $\mathbf{V}_n$ of the first n Arnoldi vectors has optimal Euclidean condition number $\operatorname{cond} \mathbf{V}_n = 1$. Here, in analogy to (56),

$$\mathbf{V}_n := \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \end{bmatrix}. \tag{90}$$

However, this optimality of the basis comes at a price, and in contrast to the Lanczos process, the Arnoli process uses recurrences of increasing length that, in general, involve all previous vectors. Using (90), the recurrences to generate the first n Arnoldi vectors can be stated compactly in matrix form as follows:

$$\mathbf{A} \mathbf{V}_n = \mathbf{V}_n \mathbf{T}_n^{(\mathrm{A})} + [\, \underbrace{\mathbf{0} \ \cdots \ \mathbf{0}}_{n-1} \ \ \hat{\mathbf{v}}_{n+1} \,].$$

Here,

$$\mathbf{T}_n^{(\mathrm{A})} = \begin{bmatrix} t_{11} & t_{12} & \vdots & \cdots & t_{1n} \\ t_{21} & t_{22} & t_{23} & \ddots & \vdots \\ 0 & t_{32} & t_{33} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & t_{n-1,n} \\ 0 & \cdots & 0 & t_{n,n-1} & t_{nn} \end{bmatrix} \tag{91}$$

is an $n \times n$, in general full, upper Hessenberg matrix, and $\hat{\mathbf{v}}_{n+1}$ is a yet unnormalized version of the $(n+1)$-st Arnoldi vector $\mathbf{v}_{n+1}$.

Next, we present a complete statement of the Arnoldi process.

Algorithm 7.1 (Arnoldi process.)

INPUT: A matrix $\mathbf{A} \in \mathbb{C}^{N \times N}$;
A starting vector $\mathbf{r} \in \mathbb{C}^N$.

OUTPUT: An $n \times n$ upper Hessenberg matrix $\mathbf{T}_n^{(\mathrm{A})} = \begin{bmatrix} t_{ij} \end{bmatrix}_{1 \leq i,j \leq n}$,
where n is the iteration index at termination of the algorithm.

(0) *Set* $\hat{\mathbf{v}}_1 = \mathbf{r}$.

For $n = 1, 2, \ldots,$ *do* (Build n-th Arnoldi vector $\mathbf{v}_n$.) :

(1) *Compute* $t_{n,n-1} = \|\hat{\mathbf{v}}_n\|$.
If $t_{n,n-1} = 0$, *then stop.*
(The Krylov subspace $\mathcal{K}_n(\mathbf{A}, \mathbf{r})$ is exhausted.)

(2) *Set* $\mathbf{v}_n = \dfrac{\hat{\mathbf{v}}_n}{t_{n,n-1}}$.

(3) *Set* $\hat{\mathbf{v}}_{n+1} = \mathbf{A}\,\mathbf{v}_n$.

(4) *For* $i = 1, 2, \ldots, n$ *do*:

$$\textit{Set } t_{in} = \mathbf{v}_i^{\mathrm{H}}\,\hat{\mathbf{v}}_{n+1} \textit{ and } \hat{\mathbf{v}}_{n+1} = \hat{\mathbf{v}}_{n+1} - \mathbf{v}_i\,t_{in}.$$

Remark 7.1 For the case of Hermitian matrices $\mathbf{A} = \mathbf{A}^{\mathrm{H}}$, the matrices $\mathbf{T}_n^{(\mathrm{A})}$, $n \geq 1$, can be shown to be tridiagonal, and thus the Arnoldi vectors can be generated by means of three-term recurrences. In fact, these three-term recurrences are just the recursions used in the Lanczos process (applied to Hermitian matrices $\mathbf{A}$ and a single starting vector $\mathbf{r}$). The Arnoldi process can thus be viewed as a generalization of the Hermitian Lanczos process to general non-Hermitian matrices.

Remark 7.2 For the case of a block $\mathbf{R} \in \mathbb{C}^{N \times m}$ of m (> 1) starting vectors, a number of block variants of the Arnoldi process have been proposed; see, e.g., [106, Chapter 6.12] for an overview. However, many of the block Arnoldi algorithms, which appeared in the literature, do not address the issue of deflation. If proper deflation is incorporated, then the vectors produced by the block Arnoldi process (applied to $\mathbf{A}$ and $\mathbf{R}$, and run with exact deflation only) build an orthonormal or block-orthonormal basis of the block Krylov subspaces $\mathcal{K}_n(\mathbf{A}, \mathbf{R})$, $n \geq 1$, defined in Section 4.4.

8 Arnoldi-Based Reduced-Order Modeling

In this section, we survey reduced-order modeling techniques that are based on the Arnoldi process. Mostly, we will consider only the single-input single-output case. Recall that this case is characterized by the data $\mathbf{A} \in \mathbb{C}^{N \times N}$, $\mathbf{r}$, $\mathbf{l} \in \mathbb{C}^N$, and that the transfer function H is given by (33).

In [111, 112], the following Arnoldi-based procedure was proposed. First, the Arnoldi Algorithm 7.1 (applied to $\mathbf{A}$ and $\mathbf{r}$) is run for n steps to generate the matrices $\mathbf{V}_n$ and $\mathbf{T}_n^{(\mathrm{A})}$ defined in (90) and (91). Then, the reduced-order transfer function

$$H_n^{(\mathrm{A})}(s) := \|\mathbf{r}\|\,\mathbf{l}^{\mathrm{T}}\,\mathbf{V}_n\,\left(\mathbf{I} - (s - s_0)\,\mathbf{T}_n^{(\mathrm{A})}\right)^{-1}\mathbf{e}_1 \tag{92}$$

is used as an approximation to H. Furthermore, in [112], it is shown that (92) matches the first n moments of H, i.e.,

$$\begin{aligned} H_n(s) &= H(s) + \mathcal{O}\left((s-s_0)^n\right) \\ &= \sum_{i=0}^{n-1} m_i \, (s-s_0)^i + \mathcal{O}\left((s-s_0)^n\right). \end{aligned} \tag{93}$$

Here, the m_i's are the moments defined in (32). On the other hand, the function (92) is again of the type (38), and thus of the same type as the n-th Padé approximant H_n. Recall from (39) that, except for certain degenerate cases, H_n matches the first $2n$ moments of H. In that sense, the Padé approximant H_n is twice as accurate as the Arnoldi-based approximation $H_n^{(\mathrm{A})}$. In particular, $H_n^{(\mathrm{A})}$ is not a Padé approximant, but only a Padé-type approximation of H. Another key difference between H_n and $H_n^{(\mathrm{A})}$ is the dependency on the representation of the linear dynamical system. Since H_n matches the maximal number of moments, H_n is unique and depends only on the transfer function H, but not on the actual data $\mathbf{A}$, $\mathbf{r}$, and $\mathbf{l}$, which appears in the representation (33) of H. In particular, for linear dynamical systems arising in circuit simulation, the reduced-order transfer function H_n is independent of the chosen method to formulate the circuit equations. On the other hand, the reduced-order transfer function (92) depends crucially on the Arnoldi matrix $\mathbf{T}_n^{(\mathrm{A})}$ that depends only on $\mathbf{A}$ and $\mathbf{r}$, but not on $\mathbf{l}$. If the same transfer function H is represented as

$$H(s) = \tilde{\mathbf{l}}^{\mathrm{T}} \left(\mathbf{I} - (s-s_0)\, \widetilde{\mathbf{A}}\right)^{-1} \tilde{\mathbf{r}}$$

with different data $\widetilde{\mathbf{A}}$, $\tilde{\mathbf{r}}$, and $\tilde{\mathbf{l}}$, the corresponding Arnoldi-based approximate transfer function will, in general, be different from (92). In particular, for linear dynamical system arising in circuit simulation, Arnoldi-based reduced-order models depend on the chosen circuit-formulation method.

Next, we illustrate this last point with a circuit example. In [111], it is shown that for RLC circuits, the reduced-order model based on (92) is stable, provided that a specific formulation of the circuit equations is used. We now present a simple RLC circuit and a formulation of the circuit equations for which the reduced-order model $H_1^{(\mathrm{A})}$ is unstable. In view of the stability result in [111], this clearly shows the dependency on the circuit-formulation method.

The example is the RLC circuit shown in Figure 8. It can be described by the equations

$$\begin{aligned} G_1(V-V_1) &= G_2(V_1-V_2) + sCV_1, \\ G_2(V_2-V_1) - I_L &= 0, \\ V_2 &= -sLI_L. \end{aligned} \tag{94}$$

We are interested in the transfer function

$$H(s) = \frac{V_1}{V}.$$

By rewriting (94) in matrix form, it follows that H is given by

$$H(s) = \mathbf{e}_1^{\mathrm{T}} \left(\mathbf{G} + s\,\mathbf{C}\right)^{-1} \left(G_1 \mathbf{e}_1\right), \tag{95}$$

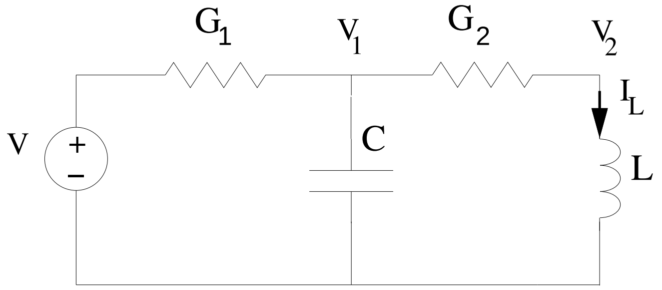

Figure 8: Simple RLC circuit.

where

$$\mathbf{G} := \begin{bmatrix} G_1 + G_2 & -G_2 & 0 \\ -G_2 & G_2 & -1 \\ 0 & -1 & 0 \end{bmatrix}, \quad \mathbf{C} := \begin{bmatrix} C & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -L \end{bmatrix}, \quad \mathbf{e}_1 := \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

From (95), we obtain a representation of H of the form (33) with expansion point $s_0 = 0$ and $\mathbf{A}$, $\mathbf{r}$, and $\mathbf{l}$ given by

$$\mathbf{A} := -\mathbf{G}^{-1}\mathbf{C}, \quad \mathbf{r} := \mathbf{G}^{-1}\left(G_1\mathbf{e}_1\right), \quad \mathbf{l} := \mathbf{e}_1.$$

Applying one step of the Arnoldi process (to $\mathbf{A}$ and $\mathbf{r}$) results in the reduced-order transfer function

$$H_1^{(\mathrm{A})}(s) = \frac{1}{(G_1 + G_2)\left(1 - s\, t_{11}^{(\mathrm{A})}\right)}, \quad \text{where} \quad t_{11}^{(\mathrm{A})} = \frac{\mathbf{r}^T\mathbf{A}\,\mathbf{r}}{\mathbf{r}^T\mathbf{r}}. \tag{96}$$

We now choose the values $G_1 = 1$, $G_2 = 0.1$, $C = 1$, and $L = 1000$. For these values, we get

$$\mathbf{A} = -\frac{1}{11}\begin{bmatrix} 10 & 0 & 1000 \\ 0 & 0 & 11000 \\ -1 & 0 & 1000 \end{bmatrix}, \quad \mathbf{r} = \frac{1}{11}\begin{bmatrix} 10 \\ 0 \\ -1 \end{bmatrix}, \quad \text{and} \quad t_{11}^{(\mathrm{A})} = \frac{7990}{1111}.$$

Hence, the reduced-order transfer function (96) has the unstable pole

$$p_1 = \frac{1}{t_{11}^{(\mathrm{A})}} = \frac{1111}{7990} > 0.$$

In [37], the passivity of Arnoldi-based reduced-order models of the type (92) is investigated. However, in [37], only the necessary condition that a passive system circuit cannot have network zeros in the right half of the complex plane is checked. In general, this condition is not sufficient to guarantee passivity, and one also needs to ensure a positivity condition on the imaginary axis; see condition (iii) of the definition of passivity in Section 6.3. In [8], we developed a simple check of this condition that only requires the solution of a certain eigenvalue problem.

Yet another approach to obtaining Arnoldi-based reduced-order models is the PRIMA algorithm [88, 89], which we already mentioned in Section 6.5. It uses a block Arnoldi method and then projects the original circuit data onto the subspace spanned by the Arnoldi vectors. For p-terminal RLC circuits, the reduced-order models are guaranteed to be passive, but match only half as many moments as the SyMPVL reduced-order models.

It is also possible to obtain the n-th Padé approximant H_n by means of the so-called *two-sided Arnoldi process* [102]. However, this requires two runs of the Arnoldi process: one with $\mathbf{A}$ and $\mathbf{r}$ to generate an orthonormal basis of $\mathcal{K}_n(\mathbf{A}, \mathbf{r})$, and one with $\mathbf{A}^{\mathrm{T}}$ and $\mathbf{l}$ to generate an orthonormal basis of $\mathcal{K}_n(\mathbf{A}^{\mathrm{T}}, \mathbf{l})$. The resulting computational procedure is more expensive than PVL, without having any apparent advantage over PVL.

In [71], reduced-order models characterized by certain Galerkin-type conditions are studied, and it is shown how to compute them via two Arnoldi runs. The resulting approximate transfer functions are just the Padé approximants H_n, although this was not noted in [71]. The techniques in [71] were further refined in [72]. Finally, Jaimoukha [70] uses ideas behind the GMRES algorithm [107] for the solution of systems of linear equations to develop an Arnoldi-based reduced-order technique that generates reduced-order model via minimizing the $\mathcal{L}_\infty$ norm of a certain residual error.

9 Circuit-Noise Computations

In this section, we discuss the use of reduced-order modeling for *circuit-noise computations.* In particular, we show how noise-type transfer functions can be rewritten so that reduced-order modeling techniques for linear dynamical systems can be applied. The material in this section is based on the recent paper [44] by Feldmann and Freund.

9.1 The Problem

Noise in electronic circuits is caused by the stochastical fluctuations in currents and voltages that occur within the devices of the circuit. We refer the reader to [34, Chapter 8] or [119] for an introduction to circuit noise and the main noise mechanisms. Noise-analysis algorithms for circuits in DC steady-state have been available for a long time in traditional circuit simulators such as SPICE [97]. As we will describe now, simulation techniques based on reduced-order modeling, such as PVL and MPVL, can easily be extended to include noise computations.

Noise in circuit devices is modeled by stochastic processes. In time domain, a stochastic process is characterized in terms of statistical averages, such as the mean and autocorrelation, and in the frequency domain, it is described by the spectral power density. The main types of noise in integrated circuits are *thermal noise*, *shot noise*, and *flicker noise.* Thermal and shot noise represent *white noise*, i.e., their *spectral power densities* do not depend on the frequency ω. Flicker noise is modeled by a stochastic process with a spectral power density that is proportional to $(1/\omega)^\beta$ where β is a constant of about one.

Next, we describe the problem of noise computation for circuits with constant excitation in steady-state (DC). As in (20), we assume that the circuit is described by a system of equations of the form

$$\mathbf{f}(\hat{\mathbf{x}}) + \frac{d}{dt}\mathbf{q}(\hat{\mathbf{x}}) = \mathbf{b}_0, \tag{97}$$

where $\mathbf{b}_0$ is now the constant excitation vector. Let $\hat{\mathbf{x}}_0$ be a DC operating point of the circuit, i.e., $\hat{\mathbf{x}}_0$ is a constant vector that satisfies $\mathbf{f}(\hat{\mathbf{x}}_0) = \mathbf{b}_0$. Adding noise sources to (97) gives

$$\mathbf{f}(\hat{\mathbf{x}}_0 + \mathbf{x}) + \frac{d}{dt}\mathbf{q}(\hat{\mathbf{x}} + \mathbf{x}) = \mathbf{b}_0 + \mathbf{B}\,\mathbf{n}(t), \tag{98}$$

where $\mathbf{n}(t)$ is a vector stochastic process of length m that describes the noise sources, $\mathbf{B} \in \mathbb{R}^{N\times m}$ is the noise-sources incidence matrix, and m denotes the number of noise sources. The vector function $\mathbf{x} = \mathbf{x}(t)$ in (98) represents the stochastical deviations of the circuit variables from the DC operating point $\hat{\mathbf{x}}_0$ that are caused by the noise sources. By linearizing (98) about $\hat{\mathbf{x}}_0$ and using the fact that $\mathbf{f}(\hat{\mathbf{x}}_0) = \mathbf{b}_0$, we obtain the following linear system of DAEs:

$$\begin{aligned} \mathbf{C}\,\frac{d\mathbf{x}}{dt} &= -\mathbf{G}\,\mathbf{x} + \mathbf{B}\,\mathbf{n}(t), \\ \mathbf{y}(t) &= \mathbf{E}^{\mathrm{T}}\,\mathbf{x}(t). \end{aligned} \tag{99}$$

Here, $\mathbf{C},\ \mathbf{G} \in \mathbb{R}^{N\times N}$ are the circuit matrices (23), $\mathbf{y}(t)$ is a vector stochastic process of length p describing the stochastical deviations at the outputs of interests due to the noise sources, and $\mathbf{E} \in \mathbb{R}^{N\times p}$ is a constant matrix that selects the outputs of interest. Note that (99) is a linear dynamical system of the form (1) with m inputs and p outputs. Thus we can use MPVL or, if $m = p = 1$, PVL to generate reduced-order models for (99).

For noise computations in frequency domain, the goal is to compute the $p \times p$ cross-spectral power density matrix $\mathbf{S}`_{\mathbf{y}}(\omega)$ of the vector stochastic process $\mathbf{y}$ in (99). It turns out that

$$\mathbf{S}_{\mathbf{y}}(\omega) = \mathbf{E}^{\mathrm{T}}\,(\mathbf{G} + \mathrm{j}\,\omega\,\mathbf{C})^{-1}\,\mathbf{B}\,\mathbf{S}_{\mathbf{n}}(\omega)\,\mathbf{B}^{\mathrm{T}}\,(\mathbf{G} + \mathrm{j}\,\omega\,\mathbf{C})^{-\mathrm{H}}\,\mathbf{E} \tag{100}$$

for all $\omega \geq 0$. Here, ω denotes frequency, and $\mathbf{S}_{\mathbf{n}}(\omega)$ is the given $m \times m$ cross-spectral power density matrix of the noise sources $\mathbf{n}(t)$ in (99). We remark that the diagonal entries of $\mathbf{S}_{\mathbf{n}}(\omega)$ are the spectral power densities of the noise sources, and that nonzero off-diagonal entries of $\mathbf{S}_{\mathbf{n}}(\omega)$ occur only if there is coupling between some of the noise sources. Moreover, if all noise sources are white, then $\mathbf{S}_{\mathbf{n}}$ is a constant matrix.

9.2 Reformulation as a Transfer Function

Clearly, the matrix-valued function (100), $\mathbf{S}_{\mathbf{y}}$, does not have the form of a transfer function (4). Consequently, the reduced-order modeling techniques we discussed so far cannot be applied directly to $\mathbf{S}_{\mathbf{y}}$. However, for the physical relevant values $\omega \geq 0$ and under some mild assumptions on the form of $\mathbf{S}_{\mathbf{n}}$, we can rewrite (100) as a function of the type (4). More precisely, we assume that

$$\mathbf{S}_{\mathbf{n}}(\omega) = (\mathbf{P}(\mathrm{j}\,\omega))^{-1} \quad \text{for all} \quad \omega \geq 0, \tag{101}$$

where

$$\mathbf{P}(s) = \mathbf{P}_0 + \mathbf{P}_1 s + \cdots + \mathbf{P}_M s^M, \quad \mathbf{P}_i \in \mathbb{C}^{m\times m}, \quad 0 \leq i \leq M, \tag{102}$$

is any matrix polynomial of degree M (i.e., $\mathbf{P}_M \neq \mathbf{0}$). In particular, for the important special case that all noise sources are white, as in the case of thermal and shot noise, we have

$$\mathbf{P}(s) = \mathbf{P}_0 = \mathbf{S}_{\mathbf{n}}^{-1} \quad \text{and} \quad M = 0. \tag{103}$$

If $\mathbf{S}_{\mathbf{n}}(\omega)$ does depend on the frequency, as in the case of flicker noise, then the assumption (101) is satisfied at least approximately, see [44].

By inserting (101) into (100) and setting

$$\mathbf{H}(s) := \mathbf{E}^{\mathrm{T}} \left(\mathbf{G} + s\,\mathbf{C}\right)^{-1} \mathbf{B} \left(\mathbf{P}(s)\right)^{-1} \mathbf{B}^{\mathrm{T}} \left(\mathbf{G} + s\,\mathbf{C}\right)^{-\mathrm{H}} \mathbf{E}, \quad s \in \mathbb{C}, \tag{104}$$

it follows that

$$\mathbf{H}(\mathrm{j}\,\omega) = \mathbf{S}_{\mathbf{y}}(\omega) \quad \text{for all} \quad \omega \geq 0. \tag{105}$$

The relation (105) suggests to first generate an approximation $\mathbf{H}_n$ to the function $\mathbf{H}$ in (104) and to then use

$$\mathbf{S}_{\mathbf{y}}(\omega) \approx \mathbf{H}_n(\mathrm{j}\,\omega) \tag{106}$$

as an approximation to $\mathbf{S}_{\mathbf{y}}$. It turns out that the function $\mathbf{H}$ can be rewritten as a transfer function of the type (4), and thus we can employ MPVL (or PVL if $p = 1$) to obtain $\mathbf{H}_n$ as an n-th matrix-Padé approximant to $\mathbf{H}$. More precisely, in [44], it is shown that

$$\mathbf{H}(s) = \widetilde{\mathbf{E}}^{\mathrm{T}} \left(\widetilde{\mathbf{G}} + s\,\widetilde{\mathbf{C}}\right)^{-1} \widetilde{\mathbf{E}} \quad \text{for all} \quad s \in \mathbb{C}. \tag{107}$$

Here, $\widetilde{\mathbf{E}} \in \mathbb{C}^{\tilde{N} \times p}$ and $\widetilde{\mathbf{G}},\ \widetilde{\mathbf{C}} \in \mathbb{C}^{\tilde{N} \times \tilde{N}}$ are matrices given by

$$\begin{gathered}
\widetilde{\mathbf{E}} := \begin{bmatrix} \mathbf{E} \\ \mathbf{0}_{N \times p} \\ \mathbf{0}_{m \times p} \\ \mathbf{0}_{m \times p} \\ \vdots \\ \mathbf{0}_{m \times p} \end{bmatrix}, \quad
\widetilde{\mathbf{G}} := \begin{bmatrix}
\mathbf{0} & \mathbf{G}^{\mathrm{T}} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\
\mathbf{G} & \mathbf{0} & \mathbf{B} & \mathbf{0} & \cdots & \mathbf{0} \\
\mathbf{0} & \mathbf{B}^{\mathrm{T}} & \mathbf{P}_0 & \mathbf{0} & \ddots & \vdots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} & \ddots & \mathbf{0} \\
\vdots & \ddots & \ddots & \ddots & \ddots & \mathbf{0} \\
\mathbf{0} & \cdots & \cdots & \mathbf{0} & \mathbf{0} & -\mathbf{I}
\end{bmatrix}, \\
\widetilde{\mathbf{C}} := \begin{bmatrix}
\mathbf{0} & -\mathbf{C}^{\mathrm{T}} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\
\mathbf{C} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{P}_1 & \mathbf{P}_2 & \cdots & \mathbf{P}_M \\
\mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\
\vdots & \ddots & \ddots & \ddots & \ddots & \mathbf{0} \\
\mathbf{0} & \cdots & \cdots & \mathbf{0} & \mathbf{I} & \mathbf{0}
\end{bmatrix},
\end{gathered} \tag{108}$$

and $\tilde{N} := 2 \cdot N + m \cdot M$.

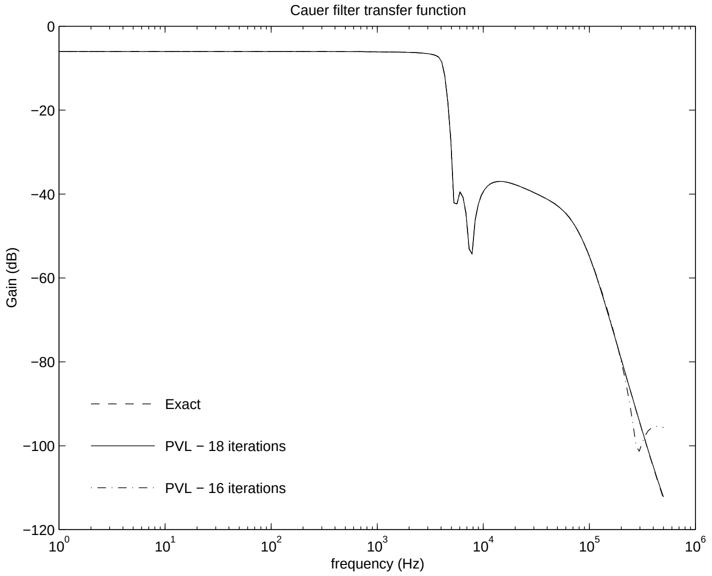

Figure 9: Transfer characteristic of the Cauer filter.

If the matrix polynomial $\mathbf{P}$ is linear, i.e., $M = 1$ in (102), the matrices (108) reduce to

$$\widetilde{\mathbf{E}} := \begin{bmatrix} \mathbf{E} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \quad \widetilde{\mathbf{G}} := \begin{bmatrix} \mathbf{0} & \mathbf{G}^{\mathrm{T}} & \mathbf{0} \\ \mathbf{G} & \mathbf{0} & \mathbf{B} \\ \mathbf{0} & \mathbf{B}^{\mathrm{T}} & \mathbf{P}_0 \end{bmatrix}, \quad \widetilde{\mathbf{C}} := \begin{bmatrix} \mathbf{0} & -\mathbf{C}^{\mathrm{T}} & \mathbf{0} \\ \mathbf{C} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{P}_1 \end{bmatrix}. \tag{109}$$

The important special case (103) of white noise is also covered by (109) with $\mathbf{P}_0 := \mathbf{S}_{\mathbf{n}}^{-1}$ and $\mathbf{P}_1 := \mathbf{0}$. In this case, by eliminating the third block rows and columns in (109), the matrices $\widetilde{\mathbf{E}}$, $\widetilde{\mathbf{G}}$, and $\widetilde{\mathbf{C}}$ can be further reduced to

$$\widetilde{\mathbf{E}} = \begin{bmatrix} \mathbf{E} \\ \mathbf{0} \end{bmatrix}, \quad \widetilde{\mathbf{G}} = \begin{bmatrix} \mathbf{0} & \mathbf{G}^{\mathrm{T}} \\ \mathbf{G} & -\mathbf{B}^{\mathrm{T}} \mathbf{S}_{\mathbf{n}} \mathbf{B} \end{bmatrix}, \quad \widetilde{\mathbf{C}} = \begin{bmatrix} \mathbf{0} & -\mathbf{C}^{\mathrm{T}} \\ \mathbf{C} & \mathbf{0} \end{bmatrix}. \tag{110}$$

9.3 A *PVL* Simulation

We now present results of a typical simulation with the noise-computation algorithm described in Section 9.2.

The example is a 5-th order Cauer filter that uses ten 741 operational amplifiers as building blocks. The total size of the problem is 463 variables. The noise sources are all white. The circuit has a single input and a single output, and we employ PVL to compute an n-th Padé approximant to the transfer function of the circuit. In addition, we also compute the spectral power density of the noise at the single output, by applying PVL to the rewritten noise-type transfer function (107) with matrices $\widetilde{\mathbf{E}}$, $\widetilde{\mathbf{G}}$, and $\widetilde{\mathbf{C}}$ given by (110). Figure 9 shows the exact transfer function of the 5-th order Cauer filter compared to the PVL-computed Padé approximants of order $n = 16$ and $n = 18$. The 18-th order approximation captures the behavior of the circuit almost exactly. Figure 10 shows the spectral power density of the output noise over the same frequency range and for the same

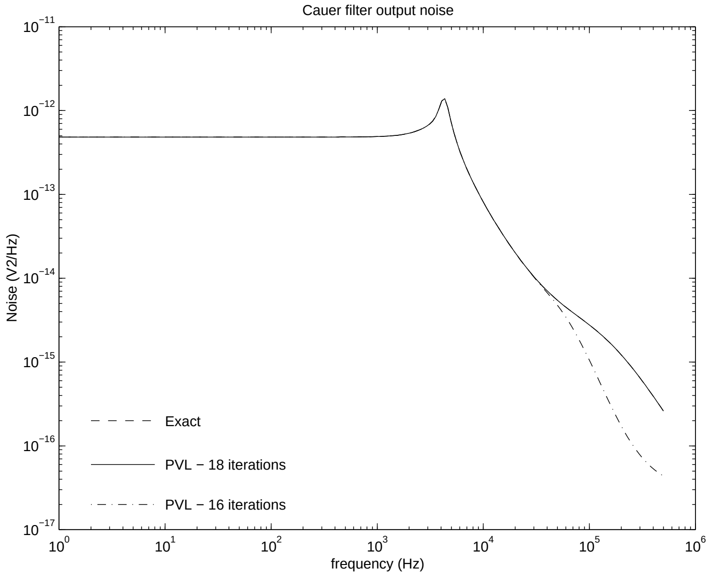

Figure 10: Spectral power density of the Cauer filter.

approximation order $n = 16$ and $n = 18$. Note that the same number n of PVL iterations is needed to obtain an almost perfect match of both the transfer function and the noise spectrum.

10 Concluding Remarks

In the last few years, reduced-order modeling techniques based on Krylov subspaces have become indispensable tools for tackling the very large linear networks that arise in the simulation of electronic circuits. Much of this development was and continues to be driven by the emerging need to accurately simulate the interconnect of electronic circuits. Today, circuit interconnect is typically modeled as very large linear networks that are generated by automatic parasitics-extraction programs. Using reduced-order modeling techniques has become crucial in order to reduce these networks to a size that is manageable for circuit simulators.

Due to ever-decreasing device sizes, the number of circuit elements on a single chip continues to increase. As a result, the use of efficient reduced-order modeling techniques will become even more important in circuit simulation. Furthermore, due to ever-increasing circuit speeds, feature sizes are becoming comparable in size with signal wavelengths. As a result, at least parts of a circuit must be modeled as distributed elements, such as transmission lines. Including distributed elements in the simulation paradigm requires a fusion of traditional lumped circuit simulation and electromagnetic simulation. As we discussed in Section 3, lumped circuit simulation leads to systems of DAEs. Electromagnetic simulation, however, involves systems of partial differential equations (PDEs). Combining lumped circuit simulation with electromagnetic simulation will thus require efficient techniques for the solution of very large systems of DAEs coupled with PDEs. One of the challenges then is to develop reduced-order modeling techniques that allow to replace parts of such coupled systems with much smaller models. Research into and development of such techniques have hardly begun.

From a numerical-analysis point of view, there is also a clear need to gain a better understanding of and to further refine existing reduced-order modeling techniques based on Krylov subspaces, especially for the multi-input multi-output case. For example, it would be desirable to have a rigorous analysis of coupled versus non-coupled Lanczos recurrences. Desirable enhancements of existing methods include post-processing techniques for MPVL and SyMPVL to detect and remedy unstable and possibly non-passive reduced-order models, and reliable and easily-computable error bounds for convergence checks.

Finally, there is a need to develop production-quality software for the basic computational modules of Krylov-subspace-based reduced-order modeling techniques. Robust black-box implementations of these algorithms are necessarily quite involved, especially for the multi-input multi-output case, due to the need for deflation and in the case of Lanczos-type algorithms, the need for look-ahead.

Acknowledgments: I am grateful to Peter Feldmann who first introduced me to circuit simulation; many of the results surveyed in this paper are based on joint work with him. I would like to thank Zhaojun Bai for producing the plots in Section 6.6.

References

[1] F. J. Alexandro, Jr. Stable partial Padé approximations for reduced-order transfer functions. *IEEE Trans. Automat. Control*, 29:159–162, 1984.

[2] J. I. Aliaga, D. L. Boley, R. W. Freund, and V. Hernández. A Lanczos-type algorithm for multiple starting vectors. Numerical Analysis Manuscript No. 96–18, Bell Laboratories, Murray Hill, New Jersey, 1996.
Also available online from `http://cm.bell-labs.com/cs/doc/96`.

[3] D. F. Anastasakis, N. Gopal, and L. T. Pillage. Enhancing the stability of asymptotic waveform evaluation for digital interconnect circuit applications. *IEEE Trans. Computer-Aided Design*, 13:729–736, 1994.

[4] B. D. O. Anderson and S. Vongpanitlerd. *Network Analysis and Synthesis*. Prentice-Hall, Englewood Cliffs, New Jersey, 1973.

[5] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9:17–29, 1951.

[6] Z. Bai. P. Feldmann, and R. W. Freund. Stable and passive reduced-order models based on partial Padé approximation via the Lanczos process. Numerical Analysis Manuscript No. 97–3–10, Bell Laboratories, Murray Hill, New Jersey, 1997.
Also available online from `http://cm.bell-labs.com/cs/doc/97`.

[7] Z. Bai. P. Feldmann, and R. W. Freund. How to make theoretically passive reduced-order models passive in practice. In *Proc. IEEE 1998 Custom Integrated Circuits Conference*, pages 207–210. IEEE, 1998.

[8] Z. Bai and R. W. Freund. Eigenvalue-based characterization and test for positive realness of transfer functions. Numerical Analysis Manuscript, Bell Laboratories, Murray Hill, New Jersey, 1999.

[9] Z. Bai and R. W. Freund. A band symmetric Lanczos process based on coupled recurrences with applications in model-order reduction. Numerical Analysis Manuscript, Bell Laboratories, Murray Hill, New Jersey, 1999.

[10] Z. Bai and Q. Ye. Error estimation of the Padé approximation of transfer functions via the Lanczos process. Manuscript, 1997.

[11] G. A. Baker, Jr. and P. Graves-Morris. *Padé Approximants*, Second Edition. Cambridge University Press, New York, New York, 1996.

[12] D. L. Boley. Krylov space methods on state-space control models. *Circuits Systems Signal Process.*, 13:733–758, 1994.

[13] D. L. Boley and G. H. Golub. The nonsymmetric Lanczos algorithm and controllability. *Systems Control Lett.*, 16:97–105, 1991.

[14] O. H. Bosgra. On parametrizations for the minimal realization problem. *Systems Control Lett.*, 3:181–187, 1983.

[15] O. H. Bosgra and A. J. J. Van der Weiden. Input-output invariants for linear multivariable systems. *IEEE Trans. Automat. Control*, 25:20–36, 1980.

[16] J. E. Bracken. Padé approximation of linear circuit responses via passive circuit synthesis. Manuscript, 1995.

[17] C. Brezinski. Partial Padé approximation. *J. Approx. Theory*, 54:210–233, 1988.

[18] A. Bultheel and M. Van Barel. Padé techniques for model reduction in linear system theory: a survey. *J. Comput. Appl. Math.*, 14:401–438, 1986.

[19] P. K. Chan. Comments on "Asymptotic waveform evaluation for timing analysis". *IEEE Trans. Computer-Aided Design*, 10:1078–1079, 1991.

[20] R. H. Chan and M. K. Ng. Conjugate gradient methods for Toeplitz systems. *SIAM Rev.*, 38:427–482, 1996.

[21] M. Celik and A. C. Cangellaris. Simulation of dispersive multiconductor transmission lines by Padé approximation via the Lanczos process. *IEEE Trans. Microwave Theory Tech.*, 44:2525–2535, 1996.

[22] M. Celik and A. C. Cangellaris. Simulation of multiconductor transmission lines using Krylov subspace order-reduction techniques. *IEEE Trans. Computer-Aided Design*, 16:485–496, 1997.

[23] M. Celik, O. Ocali, M. A. Tan, and A. Atalar. Pole-zero computation in microwave circuits using multipoint Padé approximation. *IEEE Trans. Circuits and Systems—I: Fundamental Theory and Applications*, 42:6–13, 1995.

[24] C.-T. Chen. *Linear System Theory and Design*. Oxford University Press, New York, New York, 1984.

[25] E. Chiprout and M. S. Nakhla. *Asymptotic Waveform Evaluation*. Kluwer Academic Publishers, Norwell, Massachusetts, 1994.

[26] R. R. Craig, Jr. and A. L. Hale. Block-Krylov component synthesis method for structural model reduction. *J. Guidance Control Dynamics*, 11:562–570, 1988.

[27] R. R. Craig, Jr. and T.-J. Su. A review of model reduction methods for structural control design. In C. L. Kirk and J. L. Junkins, editors, *Dynamics of Flexible Structures in Space*, pages 121–134. Computational Mechanics Publications, Southampton, UK, 1990.

[28] J. K. Cullum. Arnoldi versus nonsymmetric Lanczos algorithms for solving matrix eigenvalue problems. *BIT*, 36:470–493, 1996.

[29] J. K. Cullum and W. E. Donath. 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*, pages 505–509. IEEE Press, New York, New York, 1974.

[30] J. K. Cullum and R. A. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Volume 1, Theory*. Birkhäuser, Basel, Switzerland, 1985.

[31] J. K. Cullum and R. A. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Volume 2, Programs*. Birkhäuser, Basel, Switzerland, 1985.

[32] J. K. Cullum and R. A. Willoughby. A practical procedure for computing eigenvalues of large sparse nonsymmetric matrices. In J. K. Cullum and R. A. Willoughby, editors, *Large Scale Eigenvalue Problems*, pages 193–240. North-Holland, Amsterdam, The Netherlands, 1986.

[33] L. Dai. *Singular Control Systems*. Lecture Notes in Control and Information Sciences, vol. 118. Springer-Verlag, Berlin, Germany, 1989.

[34] J. Davidse. *Analog Electronic Circuit Design*. Prentice Hall, New York, New York, 1991.

[35] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Oxford University Press, Oxford, UK, 1989.

[36] A. E. Dunlop, W. J. Evans, and L. A. Rigge. Managing complexity in IC design—past, present, and future. *Bell Labs Technical Journal*, 2:103–125, 1997.

[37] I. M. Elfadel and D. D. Ling. Zeros and passivity of Arnoldi-reduced-order models for interconnect networks. In *Proc. 34th ACM/IEEE Design Automation Conf.*, pages 28–33. Association for Computing Machinery, Inc., New York, New York, 1997.

[38] I. M. Elfadel and D. D. Ling. A block rational Arnoldi algorithm for multipoint passive model-order reduction of multiport RLC networks. In *Technical Digest of the 1997 IEEE/ACM International Conference on Computer-Aided Design*, pages 66–71. IEEE Computer Society Press, Los Alamitos, California, 1997.

[39] P. Feldmann and R. W. Freund. Efficient linear circuit analysis by Padé approximation via the Lanczos process. In *Proceedings of EURO-DAC '94 with EURO-VHDL '94*, pages 170–175. IEEE Computer Society Press, Los Alamitos, California, 1994.

[40] P. Feldmann and R. W. Freund. Efficient linear circuit analysis by Padé approximation via the Lanczos process. *IEEE Trans. Computer-Aided Design*, 14:639–649, 1995.

[41] P. Feldmann and R. W. Freund. Reduced-order modeling of large linear subcircuits via a block Lanczos algorithm. In *Proc. 32nd ACM/IEEE Design Automation Conf.*, pages 474–479. Association for Computing Machinery, Inc., New York, New York, 1995.

[42] P. Feldmann and R. W. Freund. *Numerical Simulation of Electronic Circuits: State-of-the-Art Techniques and Challenges.* Course Notes, 1995.
Available online from `http://cm.bell-labs.com/who/freund`.

[43] P. Feldmann and R. W. Freund. Interconnect-delay computation and signal-integrity verification using the SyMPVL algorithm. In *Proc. 1997 European Conference on Circuit Theory and Design*, pages 408–413. European Circuit Society, 1997.

[44] P. Feldmann and R. W. Freund. Circuit noise evaluation by Padé approximation based model-reduction techniques. In *Technical Digest of the 1997 IEEE/ACM Int. Conf. on Computer-Aided Design*, pages 132–138. IEEE Computer Society Press, Los Alamitos, California, 1997.

[45] L. Fortuna, G. Nunnari, and A. Gallo. *Model Order Reduction Techniques with Applications in Electrical Engineering.* Springer-Verlag, London, UK, 1992.

[46] R. W. Freund. Computation of matrix Padé approximations of transfer functions via a Lanczos-type process. In C. K. Chui and L. L. Schumaker, editors, *Approximation Theory VIII, Vol. 1: Approximation and Interpolation*, pages 215–222. World Scientific Publishing Co., Inc., Singapore, 1995.

[47] R. W. Freund. Computing minimal partial realizations via a Lanczos-type algorithm for multiple starting vectors. In *Proc. 36th IEEE Conference on Decision and Control*, pages 4394–4399. IEEE, Piscataway, New Jersey, 1997.

[48] R. W. Freund. Passive reduced-order models for interconnect simulation and their computation via Krylov-subspace algorithms. Numerical Analysis Manuscript No. 98–3–06, Bell Laboratories, Murray Hill, New Jersey, 1998.
Also available online from `http://cm.bell-labs.com/cs/doc/98`.

[49] R. W. Freund. An extension of the Lanczos-Padé connection to the matrix case. Numerical Analysis Manuscript, Bell Laboratories, Murray Hill, New Jersey, 1999.

[50] R. W. Freund and P. Feldmann. Small-signal circuit analysis and sensitivity computations with the PVL algorithm. *IEEE Trans. Circuits and Systems—II: Analog and Digital Signal Processing*, 43:577–585, 1996.

[51] R. W. Freund and P. Feldmann. Reduced-order modeling of large passive linear circuits by means of the SyPVL algorithm. In *Technical Digest of the 1996 IEEE/ACM Int. Conf. on Computer-Aided Design*, pages 280–287. IEEE Computer Society Press, Los Alamitos, California, 1996.

[52] R. W. Freund and P. Feldmann. Reduced-order modeling of large linear passive multi-terminal circuits using matrix-Padé approximation. In *Proceedings of DATE'98*, pages 530–537. IEEE Computer Society Press, Los Alamitos, California, 1998.

[53] R. W. Freund, G. H. Golub, and N. M. Nachtigal. Iterative solution of linear systems. *Acta Numerica*, 1:57–100, 1992.

[54] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal. An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. *SIAM J. Sci. Comput.*, 14:137–158, 1993.

[55] R. W. Freund and N. M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.*, 60:315–339, 1991.

[56] R. W. Freund and N. M. Nachtigal. An implementation of the QMR method based on coupled two-term recurrences. *SIAM J. Sci. Comput.*, 15:313–337, 1994.

[57] R. W. Freund and N. M. Nachtigal. Software for simplified Lanczos and QMR algorithms. *Appl. Numer. Math.*, 19:319–341, 1995.

[58] R. W. Freund and N. M. Nachtigal. QMRPACK: a package of QMR algorithms. *ACM Trans. Math. Software*, 22:46–77, 1996.

[59] K. Gallivan, E. J. Grimme, and P. Van Dooren. Asymptotic waveform evaluation via a Lanczos method. *Appl. Math. Lett.*, 7:75–80, 1994.

[60] K. Gallivan, E. J. Grimme, and P. Van Dooren. A rational Lanczos algorithm for model reduction. *Numer. Algorithms*, 12:33–63, 1996.

[61] T. Geerts. Invariant subspaces and invertibility properties for singular systems: the general case. *Linear Algebra Appl.*, 183:61–88, 1993.

[62] K. Glover. All optimal Hankel-norm approximations of linear multivariable systems and their L^{∞}-error bounds. *Internat. J. Control*, 39:1115–1193, 1984.

[63] I. Gohberg, M. A. Kaashoek, and L. Lerer. On minimality in the partial realization problem. *Systems Control Lett.*, 9:97–104, 1987.

[64] G. H. Golub and J. Kautsky. Calculation of Gauss quadratures with multiple free and fixed knots. *Numer. Math.*, 41:147–163, 1983.

[65] G. H. Golub and C. F. Van Loan. *Matrix Computations*, Third Edition. The Johns Hopkins University Press, Baltimore, Maryland, 1996.

[66] W. B. Gragg. Matrix interpretations and applications of the continued fraction algorithm. *Rocky Mountain J. Math.*, 4:213–225, 1974.

[67] W. B. Gragg and A. Lindquist. On the partial realization problem. *Linear Algebra Appl.*, 50:277–319, 1983.

[68] E. J. Grimme, D. C. Sorensen, and P. Van Dooren. Model reduction of state space systems via an implicitly restarted Lanczos method. *Numer. Algorithms*, 12:1–31, 1996.

[69] X. Huang. Padé approximation of linear(ized) circuit responses. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1990.

[70] I. M. Jaimoukha. A general minimal residual Krylov subspace method for large-scale model reduction. *IEEE Trans. Automat. Control*, 42:1422–1427, 1997.

[71] I. M. Jaimoukha and E. M. Kasenally. Oblique projection methods for large scale model reduction. *SIAM J. Matrix Anal. Appl.*, 16:602–627, 1995.

[72] I. M. Jaimoukha and E. M. Kasenally. Implicitly restarted Krylov subspace methods for stable partial realizations. *SIAM J. Matrix Anal. Appl.*, 18:633–652, 1997.

[73] T. Kailath. *Linear Systems*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1980.

[74] J. Kautsky and S. Elhay. Gauss quadratures and Jacobi matrices for weight functions not of one sign. *Math. Comp.*, 43:543–550, 1984.

[75] K. J. Kerns, I. L. Wemple, and A. T. Yang. Stable and efficient reduction of substrate model networks using congruence transformations. In *Technical Digest of the 1995 IEEE/ACM International Conference on Computer-Aided Design*, pages 207–214. IEEE Computer Society Press, Los Alamitos, California, 1995.

[76] K. J. Kerns and A. T. Yang. Preservation of passivity during RLC network reduction via split congruence transformations. In *Proc. 34th ACM/IEEE Design Automation Conf.*, pages 34–39. Association for Computing Machinery, Inc., New York, New York, 1997.

[77] K. J. Kerns and A. T. Yang. Stable and efficient reduction of large, multiport RC networks by pole analysis via congruence transformations. *IEEE Trans. Computer-Aided Design*, 16:734–744, 1997.

[78] H. M. Kim and R. R. Craig, Jr. Structural dynamics analysis using an unsymmetric block Lanczos algorithm. *Internat. J. Numer. Methods Engrg.*, 26:2305–2318, 1988.

[79] H. M. Kim and R. R. Craig, Jr. Computational enhancement of an unsymmetric block Lanczos algorithm. *Internat. J. Numer. Methods Engrg.*, 30:1083–1089, 1990.

[80] S.-Y. Kim, N. Gopal, and L. T. Pillage. Time-domain macromodels for VLSI interconnect analysis. *IEEE Trans. Computer-Aided Design*, 13:1257–1270, 1994.

[81] C. Lanczos. 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.

[82] C. Lanczos. Solution of systems of linear equations by minimized iterations. *J. Res. Nat. Bur. Standards*, 49:33–53, 1952.

[83] V. L. Mehrmann. *The Autonomous Linear Quadratic Control Problem.* Lecture Notes in Control and Information Sciences, vol. 163. Springer-Verlag, Berlin, Germany, 1991.

[84] H.-D. Meyer and S. Pal. A band-Lanczos method for computing matrix elements of a resolvent. *J. Chem. Phys.* 91:6195–6204, 1989.

[85] B. C. Moore. Principal component analysis in linear systems: controllability, observability, and model reduction. *IEEE Trans. Automat. Control*, 26:17–31, 1981.

[86] T. V. Nguyen and J. Li. Multipoint Padé approximation using a rational block Lanczos algorithm. In *Technical Digest of the 1997 IEEE/ACM International Conference on Computer-Aided Design*, pages 72–75. IEEE Computer Society Press, Los Alamitos, California, 1997.

[87] A. A. Nikishin and A. Yu. Yeremin. Variable block CG algorithms for solving large sparse symmetric positive definite linear systems on parallel computers, I: general iterative scheme. *SIAM J. Matrix Anal. Appl.*, 16:1135–1153, 1995.

[88] A. Odabasioglu. Provably passive RLC circuit reduction. M.S. thesis, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1996.

[89] A. Odabasioglu, M. Celik, and L. T. Pileggi. PRIMA: passive reduced-order interconnect macromodeling algorithm. In *Technical Digest of the 1997 IEEE/ACM International Conference on Computer-Aided Design*, pages 58–65. IEEE Computer Society Press, Los Alamitos, California, 1997.

[90] D. P. O'Leary. The block conjugate gradient algorithm and related methods. *Linear Algebra Appl.*, 29:293–322, 1980.

[91] B. N. Parlett, D. R. Taylor, and Z. A. Liu. A look-ahead Lanczos algorithm for unsymmetric matrices. *Math. Comp.*, 44:105–124, 1985.

[92] L. T. Pileggi. Coping with RC(L) interconnect design headaches. In *Technical Digest of the 1995 IEEE/ACM International Conference on Computer-Aided Design*, pages 246–253. IEEE Computer Society Press, Los Alamitos, California, 1995.

[93] L. T. Pillage and R. A. Rohrer. Asymptotic waveform evaluation for timing analysis. *IEEE Trans. Computer-Aided Design*, 9:352–366, 1990.

[94] V. Raghavan, J. E. Bracken, and R. A. Rohrer. AWESpice: A general tool for the accurate and efficient simulation of interconnect problems. In *Proc. 29th ACM/IEEE Design Automation Conf.*, pages 87–92. Association for Computing Machinery, Inc., New York, New York, 1992.

[95] V. Raghavan, R. A. Rohrer, L. T. Pillage, J. Y. Lee, J. E. Bracken, and M. M. Alaybeyi. AWE–inspired. In *Proc. IEEE Custom Integrated Circuits Conference*, 1993.

[96] C. L. Ratzlaff and L. T. Pillage. RICE: rapid interconnect circuit evaluation using AWE. *IEEE Trans. Computer-Aided Design*, 13:763–776, 1994.

[97] R. A. Rohrer, L. Nagel, R. Meyer, and L. Weber. Computationally efficient electronic-circuit noise calculations. *IEEE J. Solid-State Circuits*, 6:204–213, 1971.

[98] A. E. Ruehli. Equivalent circuit models for three-dimensional multiconductor systems. *IEEE Trans. Microwave Theory Tech.*, 22:216–221, 1974.

[99] A. E. Ruehli, editor. *Circuit Analysis, Simulation, and Design Part 1: General Aspects of Circuit Analysis and Design.* North-Holland, Amsterdam, The Netherlands, 1986.

[100] A. E. Ruehli, editor. *Circuit Analysis, Simulation, and Design Part 2: VLSI Circuit Analysis and Simulation.* North-Holland, Amsterdam, The Netherlands, 1987.

[101] A. Ruhe. Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices. *Math. Comp.*, 33:680–687, 1979.

[102] A. Ruhe. The two-sided Arnoldi algorithm for nonsymmetric eigenvalue problems. In B. Kågström and A. Ruhe, editors, *Proceedings, Pite Havsbad,* Lecture Notes in Mathematics, vol. 973, pages 104–120. Springer-Verlag, Berlin, Germany, 1983,

[103] A. Ruhe. Rational Krylov sequence methods for eigenvalue computation. *Linear Algebra Appl.*, 58:391–405, 1984.

[104] A. Ruhe. Rational Krylov algorithms for nonsymmetric eigenvalue problems. II: matrix pairs. *Linear Algebra Appl.*, 197/198:283–295, 1994.

[105] Y. Saad. *Numerical Methods for Large Eigenvalue Problems.* Halstead Press, New York, New York, 1992.

[106] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Company, Boston, Massachusetts, 1996.

[107] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7:856–869, 1986.

[108] A. L. Sangiovanni-Vincentelli. Circuit simulation. In P. Antognetti, D. O. Pederson, and D. de Man, editors, *Computer Design Aids for VLSI Circuits*, pages 19–112. Sijthoff & Noordhoff International Publishers B.V., Alphen aan den Rijn, The Netherlands, 1981.

[109] Y. Shamash. Stable reduced-order models using Padé-type approximations. *IEEE Trans. Automat. Control*, 19:615–616, 1974.

[110] Y. Shamash. Model reduction using the Routh stability criterion and the Padé approximation technique. *Internat. J. Control*, 21:475–484, 1975.

[111] L. M. Silveira, M. Kamon, I. Elfadel, and J. White. A coordinate-transformed Arnoldi algorithm for generating guaranteed stable reduced-order models of RLC circuits. In *Technical Digest of the 1996 IEEE/ACM International Conference on Computer-Aided Design*, pages 288–294. IEEE Computer Society Press, Los Alamitos, California, 1996.

[112] L. M. Silveira, M. Kamon, and J. White. Efficient reduced-order modeling of frequency-dependent coupling inductances associated with 3-D interconnect structures. In *Proc. 1995 European Design and Test Conference*, pages 534–538. IEEE Computer Society Press, Los Alamitos, California, 1995.

[113] T.-J. Su. A decentralized linear quadratic control design method for flexible structures. Ph.D. thesis, The University of Texas at Austin, Austin, Texas, 1989.

[114] T.-J. Su and R. R. Craig, Jr. Model reduction and control of flexible structures using Krylov vectors. *J. Guidance Control Dynamics*, 14:260–267, 1991.

[115] A. J. Tether. Construction of minimal linear state-variable models from finite input-output data. *IEEE Trans. Automat. Control*, 15:427–436,1970.

[116] N. P. van der Meijs and T. Smedes. Accurate interconnect modeling: towards multi-million transistor chips as microwave circuits. In *Technical Digest of the 1996 IEEE/ACM Int. Conf. on Computer-Aided Design*, pages 244–251. IEEE Computer Society Press, Los Alamitos, California, 1996.

[117] H. A. van der Vorst and G. H. Golub. 150 years old and still alive: eigenproblems. In I. S. Duff and G. A. Watson, editors, *The State of the Art in Numerical Analysis*, pages 93–119. Clarendon Press, Oxford, UK, 1997.

[118] A. J. J. van der Weiden and O. H. Bosgra. The determination of structural properties of a linear multivariable system by operations of system similarity. *Internat. J. Control*, 32:489–537, 1980.

[119] A. van der Ziel. *Noise in Solid State Devices and Circuits.* John Wiley & Sons, Inc., New York, New York, 1986.

[120] C. de Villemagne and R. E. Skelton. Model reductions using a projection formulation. *Internat. J. Control*, 46:2141–2169, 1987.

[121] J. Vlach and K. Singhal. *Computer Methods for Circuit Analysis and Design*, Second Edition. Van Nostrand Reinhold, New York, New York, 1993.

[122] M. R. Wohlers. *Lumped and Distributed Passive Networks.* Academic Press, New York, New York, 1969.

[123] G.-L. Xu and A. Bultheel. Matrix Padé approximation: definitions and properties. *Linear Algebra Appl.*, 137/138:67–136, 1990.