PRECONDITIONING TECHNIQUES IN LINEAR MODEL ORDER REDUCTION

W.H.A. Schilders[1,2], J. Rommes[2]
[1]Eindhoven University of Technology
[2]NXP Semiconductors
The Netherlands

Corresponding author: W.H.A. Schilders, Eindhoven University of Technology,
Centre for Analysis, Scientific Computing and Applications
Building HG8.02, PO Box 513,
5600 MD Eindhoven, The Netherlands,
w.h.a.schilders@tue.nl

Abstract. Krylov subspace methods have become very popular, not only for solving large scale linear systems, but also in the area of model order reduction. It is also well known that the performance of iterative solution techniques depends crucially on the choice of preconditioning matrix. Within the area of model order reduction, the concept of preconditioning has not yet been introduced. However, recent reduction problems arising in electronics applications clearly demonstrate the need for some kind of preconditioning step prior to the actual model order reduction. In this paper, we present initial ideas and some test results.

1 Introduction

Model order reduction is a field of research that receives increased interest in recent years. A closer look at the techniques being used reveals that there are essentially a number of classes. In [8], these classes are identified, and a state-of-the-art overview is provided. Besides the well known balanced truncation methods originating from systems and control theory, there is also the popular class of Krylov subspace based techniques. Methods from the latter class are closely related to iterative solution methods for linear systems such as ICCG, biCGstab and CGS. Using Krylov subspaces a basis is built up in such a way that the dominant (and most important) part of the solution is found within an acceptable and preferably low number of iterations.

Within the field of numerical linear algebra, it is well known that the number of iterations within iterative solutions is very much dependent on the conditioning of the coefficient matrix. This explains the need for preconditioning techniques, which provide ways to improve the conditioning by multiplying the system by scaling matrices. In fact, one could argue that preconditioning is equivalent to applying a direct solution method (such as Gaussian elimination or Choleski decomposition) in an incomplete or partial sense. Popular choices are the incomplete Choleski decomposition for symmetric systems, and incomplete LU decompositions with thresholding (ILUT) for more general systems.

Since Krylov subspace based linear MOR is very much related to iterative solution methods for linear systems, it seems natural to also employ some kind of preconditioning. In this way, one would hope to reduce the number of basis vectors in the Krylov subspace, and hence obtain a more reduced model. So far, this concept seems not to have been pursued seriously. The main reason is that the transfer function often depends on the product of two matrices, which makes it more difficult to employ preconditioning. This is comparable to the situation encountered in generalized eigenvalue problems.

In this paper we discuss the applicability of preconditioning within the area of model order reduction. In section 2, we first give a brief overview of some concepts in model order reduction that are important for the subsequent discussion. Section 3 contains a discussion on the relation between model order reduction and the solution of linear systems, and an introduction to the concept of preconditioning in MOR. This section contains the main ideas that are currently being developed, but which need further research. Then, in section 4, we discuss a simple first case considering static electronic circuits, more specifically large resistor networks. For this type of problem, the transfer function contains only a single matrix, thereby opening up a host of possible preconditioning strategies. Ordering techniques are an important ingredient, and we use the recently developed BBD algorithm for this purpose in combination with graph theoretical methods. The results are impressive, as is demonstrated in Section 5. We finish the paper with some conclusions.

2 Model order reduction

To place model reduction in a mathematical context, we need to realize that many models developed in computational science consist of a system of partial and/or ordinary differential equations, supplemented with boundary conditions. When partial differential equations are used to describe the behaviour, one often encounters the sit-

uation that the independent variables are space and time. Thus, after (semi-)discretising in space, a system of ordinary differential equations is obtained in time. Therefore, we limit the discussion to ODE's and consider the following explicit finite-dimensional dynamical system (following Antoulas, see [1]):

$$\begin{aligned} \frac{d\mathbf{x}}{dt} &= \mathbf{f}(\mathbf{x},\mathbf{u}) \\ \mathbf{y} &= \mathbf{g}(\mathbf{x},\mathbf{u}). \end{aligned}$$

Here, $\mathbf{u}$ is the input of the system, $\mathbf{y}$ the output, and $\mathbf{x}$ the so-called state variable. The dynamical system can thus be viewed as an input-output system.

The complexity of the system is characterized by the number of its state variables, i.e. the dimension n of the state space vector $\mathbf{x}$. It should be noted that similar dynamical systems can also be defined in terms of differential algebraic equations, in which case the first set of equations in (1) is replaced by $\mathbf{F}(\frac{d\mathbf{x}}{dt},\mathbf{x},\mathbf{u}) = 0$.

2.1 Essence of MOR

Model order reduction can now be viewed as the task of reducing the dimension of the state space vector, while preserving the character of the input-output relations. In other words, we should find a dynamical system of the form

$$\begin{aligned} \frac{d\hat{\mathbf{x}}}{dt} &= \hat{\mathbf{f}}(\hat{\mathbf{x}},\mathbf{u}), \\ \mathbf{y} &= \hat{\mathbf{g}}(\hat{\mathbf{x}},\mathbf{u}), \end{aligned}$$

where the dimension of $\hat{\mathbf{x}}$ is much smaller than n. In order to provide a good approximation of the original input-output system, a number of conditions should be satisfied:

- the approximation error is small,
- preservation of properties of the original system, such as stability and passivity,
- the reduction procedure should be computationally efficient.

A special case is encountered if the functions $\mathbf{f}$ and $\mathbf{g}$ are linear, in which case the system reads

$$\begin{aligned} \frac{d\mathbf{x}}{dt} &= A\mathbf{x} + B\mathbf{u}, \\ \mathbf{y} &= C^T\mathbf{x} + D\mathbf{u}. \end{aligned}$$

Here, the matrices A, B, C, D can be time-dependent, or time-independent. For linear dynamical systems, model order reduction is equivalent to reducing the matrix A, but retaining the number of columns of B and C.

2.2 Projection methods

Most MOR methods are based on projection, truncating the solution of the original system in an appropriate basis. To illustrate the concept, consider a basis transformation T that maps the original n-dimensional state space vector $\mathbf{x}$ into a vector that we denote by

$$\bar{\mathbf{x}} = \begin{pmatrix} \hat{\mathbf{x}} \\ \tilde{\mathbf{x}} \end{pmatrix},$$

where $\hat{\mathbf{x}}$ is k-dimensional. The basis transformation T can then be written as

$$T = \begin{pmatrix} W^* \\ T_2^* \end{pmatrix},$$

and its inverse as

$$T^{-1} = (V \quad T_1).$$

Since $W^*V = I_k$, we conclude that

$$\Pi = VW^*$$

is an oblique projection along the kernel of W^* onto the k-dimensional subspace that is spanned by the columns of the matrix V.

If we substitute the projection into the dynamical system (1), the first part of the set of equations obtained is

$$\begin{aligned}\frac{d\hat{\mathbf{x}}}{dt} &= W^*\hat{\mathbf{f}}(V\hat{\mathbf{x}}+T_1\tilde{\mathbf{x}},\mathbf{u}),\\ \mathbf{y} &= \hat{\mathbf{g}}(V\hat{\mathbf{x}}+T_1\tilde{\mathbf{x}},\mathbf{u}).\end{aligned}$$

Note that this is an exact expression. The approximation occurs when we would delete the terms involving $\tilde{\mathbf{x}}$, in which case we obtain the reduced system

$$\begin{aligned}\frac{d\hat{\mathbf{x}}}{dt} &= W^*\hat{\mathbf{f}}(V\hat{\mathbf{x}},\mathbf{u}),\\ \mathbf{y} &= \hat{\mathbf{g}}(V\hat{\mathbf{x}},\mathbf{u}).\end{aligned}$$

For this to produce a good approximation to the original system, the neglected term $T_1\tilde{\mathbf{x}}$ must be sufficiently small. This has implications for the choice of the projection Π. In the following sections, various ways of constructing this projection are discussed.

2.3 Transfer function

In order to illustrate the various concepts related to model order reduction of input-output systems as described in the previous section, we consider the linear time-invariant system

$$\begin{aligned}\frac{d\mathbf{x}}{dt} &= A\mathbf{x}+B\mathbf{u},\\ \mathbf{y} &= C^T\mathbf{x}.\end{aligned}$$

The general solution of this problem is

$$\mathbf{x}(t) = \exp(A(t-t_0))\mathbf{x}_0 + \int_{t_0}^{t}\exp(A(t-\tau))B\mathbf{u}(\tau)d\tau. \tag{1}$$

A common way to solve the differential equation is by transforming it from the time domain to the frequency domain, by means of a Laplace transform defined as

$$\mathscr{L}(f)(s) \equiv \int_0^{\infty} f(t)\exp(-st)dt.$$

If we apply this transform to the system, assuming that $\mathbf{x}(0)=0$, the system is transformed to a purely algebraic system of equations:

$$\begin{aligned}(I_n - sA)\mathbf{X} &= B\mathbf{U},\\ \mathbf{Y} &= C^T\mathbf{X},\end{aligned}$$

where the capital letters indicate the Laplace transforms of the respective lower case quantities. This immediately leads to the following relation:

$$\mathbf{Y}(s) = C^T(I_n - sA)B\mathbf{X}(s). \tag{2}$$

Now define the *transfer function* $H(s)$ as

$$H(s) = C^T(I_n - sA)B. \tag{3}$$

This transfer function represents the direct relation between input and output in the frequency domain, and therefore the behavior of the system in frequency domain. For example, in the case of electronic circuits this function may describe the transfer from currents to voltages, and is then termed impedance. If the transfer is from voltages to currents, then the transfer function corresponds to the admittance. Note that if the system has more than one input or more than one output, then B and C have more than one column. This makes $H(s)$ a matrix function. The i,j entry in $H(s)$ then denotes the transfer from input i to output j.

2.4 Poles and residues

The transfer function can be expanded as follows:

$$H(s) = \sum_{j=1}^{n} \frac{R_j}{s - p_j}, \tag{4}$$

where the p_j are the poles, and R_j are the corresponding residues. The poles are exactly the eigenvalues of the matrix $-A^{-1}$. In fact, if the matrix E of eigenvectors is non-singular, we can write

$$-A^{-1} = E\Lambda E^{-1},$$

where the diagonal matrix Λ contains the eigenvalues λ_j. Substituting this into the expression for the transfer function, we obtain:

$$H(s) = -C^T E(I + s\Lambda)^{-1} E^{-1} A^{-1} B.$$

Hence, if B and C contain only one column (which corresponds to the single input, single output or SISO case), then

$$H(s) = \sum_{j=1}^{n} \frac{l_j^T r_j}{1 + s\lambda_j},$$

where the l_j and r_j are the left and right eigenvectors, respectively.

We see that there is a one-to-one relation between the poles and the eigenvalues of the system. If the original dynamical system originates from a differential algebraic system, then a generalized eigenvalue problem needs to be solved. Since the poles appear directly in the pole-residue formulation of the transfer function, there is also a strong relation between the transfer function and the poles or, stated differently, between the behavior of the system and the poles. If one approximates the system, one should take care to approximate the most important poles. There are several methods that do this. In general, we can say that, since the transfer function is usually plotted for imaginary points $s = \omega i$, the poles that have a small imaginary part dictate the behavior of the transfer function for small values of the frequency ω. Consequently, the poles with a large imaginary part are needed for a good approximation at higher frequencies. Therefore, a successful reduction method aims at capturing the poles with small imaginary part rather, and leaves out poles with a small residue.

3 Preconditioning within MOR

Having established the connection between model order reduction and solving linear systems or computing solutions of eigenvalue problems, it is clear that many concepts and methods from the latter category can be used to find suitable MOR algorithms. Hence, it is not surprising that many reduction techniques are based on the popular Krylov subspace methods, in which the basis for the reduced space is found by constructing a Krylov subspace. The first such method was the Pade-via-Lanczos (PVL) method developed by Freund and Feldmann [2], which meant a small revolution in MOR. Soon after, new methods followed, such as SymPVL, SVD-Laguerre [5], the Laguerre method based on intermediate orthogonalisation [3], and PRIMA [7] which is based on the Arnoldi method.

Although the publication of PVL has sparked many other MOR methods based on Krylov subspace methods, one concept that is indispensable in solving linear systems iteratively appears to have been neglected so far. This is the important concept of preconditioning which, when used appropriately, renders a much improved conditioning of the linear system. Often, this means that the eigenvalue distribution is changed completely, and that the ratio of the largest and the smallest eigenvalue is reduced significantly.

One may wonder why preconditioning has not been considered so far in the area of model order reduction. On the one hand, one may argue that changing the location of the eigenvalues is in contradiction to the aim of model order reduction, namely locating the position of dominant poles. If this is the aim, one should not change the positions of the eigenvalues. This is comparable to the situation of eigenvalue problems, where one also wishes to determine the location of the eigenvalues of the original system in a precise way. This possibly explains the low interest in preconditioning for eigenvalue problems; indeed, when scanning the literature, only very few publications are found [6].

However, the situation in MOR is different from that of solving eigenvalue problems. Model order reduction aims at finding an efficient representation of the transfer function, and this is not conflicting with the concept of preconditioning. In fact, suppose we would like to find a reduced representatiom of the transfer function given in (3). Then we need to solve the linear system

$$(I_n - sA)\mathbf{X} = B\mathbf{U},$$

often at some expansion point $s = s_0$. If we find a suitable preconditioner P, then we would solve (if preconditioned from the left)

$$P(I_n - sA)\mathbf{X} = PB\mathbf{U},$$

and use the Krylov subspace based on the matrix $P(I_n - sA)$. This could then lead to a basis that is substantially reduced as compared to the unpreconditioned system. In turn, this means a more reduced representation of the original system behaviour.

An unexpected example of such preconditioning is found when having a closer look at the Laguerre type methods. As is shown in [4], the Laguerre-SVD method is equivalent to a onesided rational Krylov method with a shift. However, in the presence of finite precision arithmetic, these methods may clearly differ. Thus, even though equivalent theoretically, there may be a significant difference in practice. Laguerre-SVD can, therefore, be considered as a preconditioned version of a Krylov subspace method. For more details, we refer to [4].

Since this concept of preconditioning in model order reduction is new, there is much research to be done in order to assess the usefulness and effectiveness. A complication is that "preconditioning" means many different things, and is certainly not limited to multiplying or scaling the original system matrix. In fact, as preconditioning is equivalent to executing a pre-step using a direct method, aspects like ordering play an important role, or partial elimination.

In order to analyse these aspects in a thorough way, we started with a very simple situation, viz. the static case. In this case, the frequency parameter s does not play a role. In the electronics industry, this means we restrict ourselves to resistive networks. It turns out that this is already an extremely interesting case, that is worthwhile exploring before turning our attention to the more general case. In the remaining part of this paper, we will describe our work for this special problem, and present results that demonstrate the potential of the new methods.

4 MOR for resistive networks

Applying Kirchhoff's Current Law and Ohm's Law for resistors leads to the following system of equations for a resistor network with N resistors (resistor i having resistance r_i) and n nodes:

$$\begin{bmatrix} R & P \\ -P^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{i}_b \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{i}_n \end{bmatrix}, \tag{5}$$

where $R = \mathrm{diag}(r_1, \ldots, r_N) \in \mathbb{R}^{N \times N}$ is the resistor matrix, $P \in \{-1, 0, 1\}^{N \times n}$ is the incidence matrix, $\mathbf{i}_b \in \mathbb{R}^N$ are the resistor currents, $\mathbf{i}_n \in \mathbb{R}^n$ are the injected node currents, and $\mathbf{v} \in \mathbb{R}^n$ are the node voltages. Note that the first blockrow of equations in (5) reflects Ohm's Law, while the second reflects Kirchhoff's Current Law.

We will also make use of an alternative system of equations, derived from (5) by eliminating the resistor currents $\mathbf{i}_b = -R^{-1}P\mathbf{v}$:

$$G\mathbf{v} = \mathbf{i}_n, \tag{6}$$

where $G = P^T R^{-1} P \in \mathbb{R}^{n \times n}$ is symmetric positive semidefinite. Note that this system of equations can also be obtained by applying MNA directly (G is also known as the conductance matrix). Since currents can only be injected in external nodes, and not in internal nodes of the network, system (6) has the following structure:

$$\begin{bmatrix} G_{11} & G_{12} \\ G_{12}^T & G_{22} \end{bmatrix} \begin{bmatrix} \mathbf{v}_e \\ \mathbf{v}_i \end{bmatrix} = \begin{bmatrix} B \\ 0 \end{bmatrix} \mathbf{i}_e, \tag{7}$$

where $\mathbf{v}_e \in \mathbb{R}^{n_e}$ and $\mathbf{v}_i \in \mathbb{R}^{n_i}$ are the voltages at external and internal nodes, respectively ($n = n_e + n_i$), $\mathbf{i}_e \in \mathbb{R}_e^n$ are the currents injected in external nodes, $B \in \{-1, 0, 1\}^{n_e \times n_e}$ is the incidence matrix for the current injections, and $G_{11} = G_{11}^T \in \mathbb{R}^{n_e \times n_e}$, $G_{12} \in \mathbb{R}^{n_e \times n_i}$, and $G_{11} = G_{11}^T \in \mathbb{R}^{n_i \times n_i}$. The block G_{11} is also referred to as the terminal block.

This leads to the problem of reducing large resistor networks, which is defined as follows: given a very large resistor network described by (7), find an equivalent network that

(a) has the same terminals,

(b) has exactly the same path resistances between terminals,

(c) has $\hat{n}_i \ll n_i$ internal nodes,

(d) has $\hat{N} \ll N$ resistors,

(e) is realizable as a netlist so that it can be (re)used in the design flow as subcircuit of large systems.

Important remark: Simply eliminating all internal nodes will lead to an equivalent network that satisfies conditions (a)–(c), but violates (d) and (e): for large numbers n_e of terminals, the number of resistors $\hat{N} = (n_e^2 - n_e)/2$ in the dense reduced network is in general much larger than the number of resistors in the sparse original network (N of $O(n)$), leading to increased memory and CPU requirements.

4.1 Reduction by eliminating internal nodes

Looking at (7), it may seem attractive to eliminate *all* the internal unknowns by computing the Schur complement of G_{22}, resulting in the much smaller, equivalent system

$$(G_{11} - G_{12}G_{22}^{-1}G_{12}^T)\mathbf{v}_e = B\mathbf{i}_e.$$

Since the number n_e of terminals is usually much smaller than the number n_i of internal unknowns, in terms of unknowns this leads in general to a huge reduction. However, in terms of resistors (and hence network size), the reduction, if any, depends on the number of resistors in the original network. Since the original network is in general very sparse (typically three resistors per node on average) and the number of terminals can be large ($O(10^4)$), this approach will lead to network expansion instead of reduction in many cases: the Schur complement will be dense, resulting in $O(n_e^2)$ resistors. Of course, if $n_e \ll n_i$ and $n_e(n_e-1)/2 \ll N$ elimination of all internal nodes may still be attractive (note $n_e(n_e-1)/2$ is the upper bound on the number of resistors in the reduced network), and since this can be computed before doing the actual elimination we always include a check on this. If it is decided to eliminate all internal nodes, the reduced conductance matrix G_k can be computed efficiently using the Cholesky factorization $G_{22} = LL^T$:

$$G_k = G_{11} - QQ^T,$$

where $Q = L^{-1}G_{12}^T$.

In many cases, however, elimination of all internal nodes leads to a dramatic increase in the number of resistors and is hence not advisable. On the other hand, we know that the sparse circuit matrices allow for sparse Cholesky factorizations, provided clever reordering strategies such as Approximate Minimum Degree are used. Our algorithm for reducing large networks heavily relies on reordering of rows and columns to minimize fill-in.

4.2 Effective reduction of resistor networks

The new approach that we will describe next, combines well-established techniques for sparse matrices with graph algorithms. This approach has several advantages:

- the run times do not depend on the number of terminals, and are linear in the number of states,
- both the number of states and number of elements are decreased dramatically,
- the reduced networks are exact (i.e., no approximations are made),
- and the approach can be implemented using existing software libraries.

We use a divide-and-conquer approach to reduce large networks: first the network is partitioned into smaller parts that can be reduced individually. In the second and final phase the partially reduced network is reduced even further by a global approach. In the following subsections we will describe each of the steps in more detail. Finally, we would like to stress that although in this paper we describe and apply the approach for resistor only networks, it can be applied to RC and RCLk networks as well. The key observation, as will become clear in the following, is that internal nodes are identified that partition the network in disconnected parts that are easier to reduce.

In the following we make extensive use of graphs. A graph $\mathscr{G} = (V,E)$ consists of a set V of vertices (or nodes) and a set E of edges between nodes. If all edges in a graph are undirected, the graph is called undirected. It is well known that a circuit can be represented as an undirected graph: the nodes of the circuit are the vertices, and the resistors are the edges. Hence, for resistor networks we have $V = \{0,1,2,3,\ldots,n-1\}$ (there are n nodes, assumed to be numbered consecutively starting at 0) and $E = \{(i,j)|\text{there is a resistor between node } i \text{ and } j\}$. Furthermore, $|V| = n$ (the number of nodes) and $|E| = N$ (the number of resistors).

4.3 Strongly connected components

A graph is called connected if there exists a path from any vertex to any other vertex in the graph. It is easy to see that if a network is not connected, a first simplification can be obtained by considering the connected subgraphs separately. Computing the strongly connected components of an undirected graph is a well-known graph problem and can be solved in $O(|V|+|E|) = O(n+N)$ time using two depth first searches. There are many standard implementations available; in our experiments we have used the function `dmperm` of Matlab.

The presence of connected components is very natural in extracted networks: a piece of interconnect connecting two other elements such as diodes or transistors, for instance, results in an extracted network with two terminals, disconnected from the rest of the extracted circuit.

4.4 Two-connected components

The next phase is to compute the two-connected (or biconnected) components of every connected component. A connected component is called two-connected if it becomes disconnected by removing one vertex. The vertices in a connected component with this property are called articulation nodes. Similar to the classification of connected

components, the two-connected components and articulation nodes can be used to obtain a next reduction. The two-connected components of a connected graph can be computed in $O(|V|+|E|) = O(n+N)$ time.

If a two-connected component is connected to the rest via two articulation nodes and has no terminals, it can be replaced by a single resistor connecting the two articulation nodes. The resistance of this resistor can be computed using the same elimination procedure as before.

If a two-connected component is connected to the rest via exactly one articulation node and has exactly one terminal, it can be replaced by a single resistor connecting the terminal and the articulation node. The resistance of this resistor can be computed using an elimination procedure.

If a two-connected component has multiple articulation nodes and terminals, one can choose to eliminate all non-terminals and non-articulation nodes (if this gives a large enough reduction in number of resistors), to apply a more advanced reduction scheme, or to keep the component for the moment and reduce it in the next phase together with the other components.

4.5 Elimination of selected internal nodes

This phase is the crucial phase in the algorithm, since here it is determined which internal nodes to eliminate and which not. The paradigm is: eliminate those internal nodes that give the least fill-in. As discussed before, eliminating all internal nodes is not an option. Furthermore, it is well known that determining the optimal order of elimination is NP-complete. Fortunately, there exist efficient algorithms to determine a suboptimal order. In our approach we make use of Approximate Minimum Degree. There is one important difference, however, between the usual application of AMD (determination of a fill-in minimizing row/column order for computing Cholesky or LU factorizations) and the network reduction problem: in our case, the terminals (a subset of all unknowns) cannot be eliminated, and hence they have to be treated in a special way.

We have considered four alternatives for this:

1. Apply AMD to the original matrix and eliminate internal nodes one-by-one in the order computed by AMD. Skip terminals that might have mixed with the internal nodes. Since this turned out to give only a modest reduction in the number of internal nodes and resistors, we will not consider this option any further.
2. Apply AMD to the block corresponding to internal nodes only. Numerical experiments have shown that not taking into account the fill-in caused in (and by) the blocks corresponding to terminals results in a reordering that is far from optimal and hence we will not consider this option any further.
3. Make the terminal block dense before applying AMD. As a result, the terminal nodes will be among the last to eliminate. To find the optimal reduction, eliminate internal nodes one-by-one in the order computed by AMD. The rationale behind this is that we would like the rows corresponding to terminals to be considered as well in the reordering process, but that they should be among the last unknowns to be eliminated. Nevertheless, this keeps the possibility open for internal nodes that cause even more fill-in to be detected by AMD (and hence to be kept in the reduced network). The costs for AMD increase linearly with the number of nodes and resistors, but making the terminal block dense can cost quite some memory if the number of terminals is large (these costs can be alleviated by making the terminal block just a factor more dense than the other rows).
4. Recursive reordering: Apply AMD to the original matrix and eliminate internal nodes one-by-one in the order computed by AMD. As soon as a terminal is encountered, move it to the last row and reorder the partially reduced matrix. Repeat this procedure until all internal nodes are eliminated, while keeping track of the reduced system with least fill-in. The idea behind this recursive approach is that AMD for the original network gives the most optimal reordering. However, the terminals can end up anywhere in this order. By eliminating the internal unknowns up to the first occurrence of a terminal, moving the corresponding row(s) to the last row(s) and then computing a new order for the partially reduced matrix and repeating the procedure we might expect to arrive at a reduced network that is much smaller. The worst-case running time for this approach depends quadratically on the number of nodes and resistors.

We have implemented alternatives 3 and 4, including an option to do both and take the smallest reduced network. Our experience in practice is that alternative 3 is satisfactory in 95% of the cases; in 5% of the cases alternative 4 gives a huge reduction in the number of nodes and resistors, while the additional computational effort is limited. See Section 5 for more details.

4.6 Complexity

The connected components and the two-connected components can be detected with $O(n+N)$ operations. The total costs for eliminating the internal nodes are $O(n^\alpha)$, where $1 < \alpha \leq 2$, since the original conductance matrix is very sparse. The worst case costs for the recursive approach, where AMD is called repetetively, are $O(n^2)$. However, in practice we observed much lower costs (nearly linear complexity) in all cases. Hence, the reduction algorithm is perfectly scalable to very large sparse networks, as is also confirmed by the experiments in the next section.

5 Numerical results

In this section we present results of the algorithms for large resistor networks that originate from realistic designs of VLSI chips. We show results for the reduction of very large resistor networks with many terminals with our implementation. The algorithyms were implemented in Matlab 7.5 (R2007B), relying heavily on the functions `amd` and `symamd`. For the graph algorithms we used the Matlab library MatlabBGL. All experiments were carried out on Linux platforms on Intel Xeon 2.66GHz machines with 16GB RAM. For the circuit simulations we used Spectre (ver. 6.1.1.378) and Pstar (ver. 5.4.2).

Table 1 shows results for resistor networks that result from ESD analysis of realistic chip designs and interconnect layouts. Networks I–III contain diodes as well, and one is interested in the path resistances between certain terminals. Since simulation of the original networks is time consuming or not possible at all, reduced networks are needed. Network IV also contains other nonlinear elements such as transistors. Here the original network could not be simulated.

The reduction factors for the number of internal nodes vary from 5x for network III to 50x for network II. The reduction factors for the number of resistors vary from 2.5x for network IV to 4.5x for network III. As a result, the computing time for calculating path resistances, including the computation of the reduced network, is more than 10 times smaller.

Table 1: Results for the reduction algorithm reduceR.

	Network I		Network II		Network III		Network IV	
	Original	ROM	Original	ROM	Original	ROM	Original	ROM
#terminals	3260		1978		15299		8000	
#int nodes	99k	8k	101k	1888	1M	180k	46k	6k
#resistors	161k	56k	164k	39k	1.5M	376k	67k	26k
#other elements	1874		1188		8250		29k	
#other unknowns	0		0		0		11k	
CPU time reduction	130 s		140 s		1250 s		75 s	
CPU time simulation	67 hrs	6 hrs	20 hrs	2 hrs	–	120h	–	392s
Speed up	10x		12x		∞		∞	

Although the times for computing the reduced network do not depend on the number of terminals, the obtained reduction rates do: network I and network II are quite similar, except for the number terminals (3260 vs. 1978). This difference in terminals is also visible in their respective connected components. Following the reasoning in Section 4.1, reducing a component with many terminals is in general harder than reducing a component with only a few terminals: the more terminals, the more fill-in, or, equivalently, the fewer options for eliminating internal nodes. This explains why the reduction for network II is much better.

6 Conclusions

In this paper, we presented ideas for adding a preconditioning step to methods used in model order reduction. In our opinion, this is an essential step in making the reduction even more attractive. As is the case in the solution of linear systems, it is suggested that the iterative procedure for obtaining a reduced basis is preceded by a direct method. Aspects to be considered are ordering and degree of elimination, and it has been shown that graph theory plays an important role in makiing such decisions.

As an illustrative example, we presented an algorithm for efficient reduction of large resistor networks. These reduced networks can be (re)used in full circuit simulation and since the reduced networks are exactly equivalent, no approximation error is made. The reduction rates in the number of nodes and resistors vary from 80% to 98% and from 60% to 80%, respectively. As a result, full circuit simulations, including the time needed for reducing the networks, were more than 10 times faster, and in many cases only possible using the reduced networks.

7 References

[1] A.C. Antoulas,*Approximation of Large-Scale Dynamical Systems*, SIAM series on Advances in Design and Control (2005)

[2] P. Feldmann and R. Freund, Efficient linear circuit analysis by Padé approximation via the Lanczos process, *IEEE Trans. Computer-Aided Design*, volume 14, pp. 137-158 (1993)

[3] P.J. Heres, *Robust and efficient Krylov subspace methods for Model Order Reduction*, PhD Thesis, TU Eindhoven, The Netherlands (2005)

[4] M.E. Hochstenbach, J. Rommes and W.H.A. Schilders, *A note on Laguerre-SVD model order reduction* (in-preparation)

[5] L. Knockaert and D. De Zutter, Laguerre-SVD Reduced-Order Modelling, *IEEE Trans. Microwave Theory and Techn.*, volume 48(9), pp. 1469-1475 (2000)

[6] K. Meerbergen and Z. Bai, *The Lanczos method for parameterized symmetric linear systems with multiple right-hand sides*, Report TW527, KU Leuven (2008)
[7] A. Odabasioglu and M. Celik, PRIMA: passive reduced-order interconnect macromodeling algorithm, *IEEE Trans. Computer-Aided Design*, volume 17(8), pp. 645-654 (1998)
[8] W.H.A. Schilders, H.A. van der Vorst and J. Rommes, *Model Order Reduction: Theory, Research Aspects and Applications*, Mathematics in Industry series, volume 13, Springer, Berlin (2008)