Error bounds for reduction of multi-port resistor networks

Ugryumova, M.; Rommes, J.; Schilders, W.H.A.

Published: 01/01/2010

Document Version
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the author’s version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 15. Mar. 2018
Error bounds for reduction of multi-port resistor networks

by

M.V. Ugryumova, J. Rommes, W.H.A. Schilders
Error Bounds for Reduction of Multi-Port Resistor Networks

M. V. Ugryumova¹, J. Rommes², W. H. A. Schilders³

¹ Department of Mathematics and Computer Science of Eindhoven University of Technology, Eindhoven, The Netherlands (phone: 31-40-2475546; fax: 31-40-2442489; e-mail: m.v.ugryumova@tue.nl).
² NXP Semiconductors (joost.rommes@nxp.com).

SUMMARY

The interconnect layouts of chips can be modeled by large resistor networks. In order to be able to speed up simulations of such large networks, reduction techniques are applied to reduce the size of the networks. For some class of networks, an existing reduction strategy does not provide sufficient reduction in terms of the number of resistors appearing in the final network. In this paper we propose an approach for obtaining a further reduction in the amount of resistors. The suggested approach improves sparsity of the conductance matrix by neglecting resistors which do not contribute significantly to the behavior of the circuit. Explicit error bounds, which give an opportunity to control the errors due to approximation, have been derived. Numerical examples show that the suggested approach appears promising for multi-terminal resistor networks and, in combination with the existing reduction strategy, leads to better reduction.

KEY WORDS: Resistor networks; model order reduction; generalized eigenvalue problem; Cholesky factorization; singular value

1. INTRODUCTION

The interconnect layouts of chips can be modeled by large resistor networks. Such networks may contain up to millions of resistors, hundreds of thousands of internal nodes and thousands of external nodes and, as a result, simulations of such networks may be very time consuming or not possible. In order to be able to carry out simulations, model order reduction techniques are used. In [1], an exact reduction technique for resistor networks has been suggested. The approach is based on finding a special order in which internal nodes are eliminated. This allows to minimize sparsity of conductance matrix, and, therefore, the number of resistors in the reduced model. However the above approach does not always deliver a good reduction in terms of the number of resistors in the final circuit. For instance, resistor networks with many terminals, extracted by the use of finite element method [2], cannot be reduced efficiently, since any elimination of any internal nodes will lead to hardly any reduction in the amount of resistors.

In this paper we propose an approach for obtaining a reduction in the amount of resistors. The suggested approach improves sparsity of the conductance matrix by neglecting resistors, which do not contribute significantly to the behavior of the circuit. Further we refer to it as simplification of resistor networks. In order to control the quality of approximation, we derive explicit error bounds and analyze them from different perspectives (sharpness, implementation issues). The suggested approach appears promising for multi-terminal resistor networks and, in combination with ReduceR, can improve reduction.

This paper is organized as follows. In section 2, we summarize the modeling of resistor networks. In section 3, we suggest two criteria, which are used to measure approximation of resistor networks. In sections 4 and 5 we derive correspondent error estimations for each criterion. Suggested algorithms for simplification are discussed in section 6. In section 7, we provide numerical examples and discuss the performance of the suggested algorithms. Section 8 concludes.
2. CIRCUIT EQUATIONS

The nodal admittance formulation is based on Kirchhoff current law, which states that the algebraic sum of currents leaving any node is zero. For an \( n \)-port resistor network it can be written as [3]:

\[
Gv = i, \tag{1}
\]

where \( G = \mathbb{R}^{n \times n} \) is symmetric positive semidefinite conductance matrix \( (G \succeq 0) \), \( v \in \mathbb{R}^n \) are the node voltages, and \( i \in \mathbb{R}^n \) are the currents injected in ports (external nodes). Subdividing a set of nodes into external and internal, one can rewrite (1) in a block form:

\[
\begin{pmatrix}
G_{11} & G_{12} \\
G_{12}^T & G_{22}
\end{pmatrix}
\begin{pmatrix}
v_e \\
v_i
\end{pmatrix}
= \begin{pmatrix}
B \\
0
\end{pmatrix}i_e, \tag{2}
\]

where \( v_e \in \mathbb{R}^{n_e} \) and \( v_i \in \mathbb{R}^{n_i} \) are the voltages at external and internal nodes, respectively \((n = n_e + n_i)\), \( i_e \in \mathbb{R}^n \) 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 \( G_{22} = G_{22}^T \in \mathbb{R}^{n_i \times n_i} \).

A \( k \)-th current source between terminals \( a \) and \( b \) with current \( j \) leads to contributions \( B_{a,k} = 1, B_{b,k} = -1, \) and \( l_i(k) = j \). If current is only injected in a terminal \( a \), then \( B_{a,k} = 1 \) and \( l_i(k) = j \). Systems (2) must be grounded, i.e. the equation corresponding to the ground node must be removed from the system.

Deleting a single conductance \( g \) between two nodes \( a \) and \( b \) from the conductance matrix \( G \) leads to a network with a new conductance matrix \( \tilde{G} \), obtained from \( G \) as

\[
\tilde{G} = G - (e_a - e_b)g(e_a - e_b)^T, \tag{3}
\]

where \( e_a \) and \( e_b \) are the \( a \)-th and \( b \)-th unit vectors, respectively. In this case we say that \( \tilde{G} \) is obtained from \( G \) by a rank-1 correction: rank-1 matrix \((e_a - e_b)^Tg(e_a - e_b)\) is a stamp corresponding to the conductance \( g \). Introducing the notation

\[
(e_a - e_b) = m, \tag{4}
\]

the conductance matrix can be rewritten as a sum of rank-1 corrections:

\[
G = \sum_{i=1}^{N} m_i g_i m_i^T. \tag{5}
\]

3. SIMPLIFICATION OF RESISTOR NETWORKS

In general, networks arising during parasitic extraction contain large resistor subnetworks and nonlinear elements like diodes and transistors. Simulation of such complex networks may be very time consuming or unfeasible. A way to overcome this problem is to use model order reduction techniques for resistor subnetworks which will lead to decreased simulation times of the complex networks. In [1], the problem of reduction of a large resistor network is defined as follows: given a very large resistor network described by (1), find an equivalent network that:

(a) has the same terminals;
(b) has exactly the same path resistances between terminals;
(c) has \( \tilde{n}_i \ll n_i \) internal nodes;
(d) has \( \tilde{N} \ll N \) resistors;
(e) realizable as a netlist.

If the number of external nodes (ports) is large, the full elimination of internal nodes sometimes is not the best option, since it leads to violation of condition (d): the number of resistors \( \tilde{N} = (n_c^2 - n_e)/2 \) in the dense matrix may be larger than the number of resistors, \( N \), in the sparse original
network. The algorithm for exact reduction of resistor networks, ReduceR [1], uses two concepts: graph algorithms and matrix reordering strategy (AMD) which together with node elimination usually lead to a sparser reduced conductance matrix than a straightforward elimination of all internal nodes. However, there are networks for which ReduceR fails. For example, networks which come from substrate extraction based on the Finite Element Method (FEM) usually have a specific quadrilateral structure with large and sparse conductance matrices [2],[4]. The exact reduction of such networks with many terminals is challenging: full or partial elimination of internal nodes may not lead to efficient reduction.

The following small example will demonstrate the problem. In Figure 1, the network has 32 nodes (16 internal nodes, 16 external nodes) and 64 resistors, which correspond to edges of the cubes. Elimination of all internal nodes is not an option here, because the reduced network will give a dense matrix with 120 resistors. Reduction by ReduceR does not help much: the reduced network has 15 internal nodes, 16 external nodes and the same 64 resistors. In fact the best exact reduction for this network is quite poor: 12 internal nodes, 16 external nodes and 64 resistors. Note that the original network has 8 nodes (in the corners) with degree 3, 16 nodes with degree 4 and 8 nodes with degree 5. At this point, elimination of any internal node in the corners does not decrease the number of resistors (edges) and elimination of any non-corner internal node will only increase the number of resistors. Consequently, one cannot reduce the network further.

![Figure 1. Network with 16 external nodes (black dots), 16 internal nodes and 64 resistors (edges of the cubes).](image)

In order to deal with such cases, we suggest a new approach, which we will further call simplification. The idea of simplification is to neglect some resistances which do not affect significantly the behavior of the network. This approach does not deliver exact reduction as in case of ReduceR, however the error of reduction is supposed to be controlled explicitly. The goal of the simplification is to improve sparsity of the conductance matrix before or after reduction. Thus if the number of terminals is large and exact reduction is poor, simplification can be a good alternative to exact reduction. The drawback of simplification is that success will depend on the value of resistances in the network and a given tolerance for approximation.

Before going further we need to choose an error which we want to control under a certain tolerance. The first proposition is to consider a relative error of voltages at all nodes:

\[
Err_v := \frac{||v - \tilde{v}||}{||v||} = \frac{||G^{-1}i - \tilde{G}^{-1}i||}{||Gx^{-1}i||} < \epsilon, \tag{6}
\]

i.e. for a given \(\epsilon \) and \(G \) one has to find a simplified \(\tilde{G} \) such that (6) holds true. It is supposed that \(\tilde{G} \) is obtained from \(G \) by neglecting certain entries. Since in practice, the current, \(i \) is unknown in advance, computing (6) requires the knowledge of an error bound (estimation) which must be independent of \(i \). An error estimation based on the condition number of \(G, \kappa(G) \), can be derived, for
instance, based on the approach presented in [5]:

\[
\frac{||v - \tilde{v}||}{||v||} \leq \kappa(G) \cdot tol = \epsilon,
\]

(7)

where parameter \(tol\) has to be chosen no larger than \(\frac{1}{\kappa(\tilde{G})}\). Moreover \(tol\) satisfies

\[
||G - \tilde{G}|| \leq tol \cdot ||G||.
\]

(8)

Thus for a given \(\epsilon\) resistors satisfying (8) can be deleted. From a practical point of view estimation (7) is not sharp and does not deliver significant improvements in sparsity neither for original nor for reduced resistor networks [6]. In section 4 we will derive a different estimation for (6) which is sharp and cheap to compute.

The second proposition is to consider an error which is based on neglecting resistors that do not affect much all path resistances, i.e.,

\[
Err_p := \left| \frac{R_{ij} - \tilde{R}_{ij}}{R_{ij}} \right| < \delta, \quad i, j = 1, \ldots, n_e,
\]

(9)

where \(R_{ij}\) is the path resistance between the external nodes \(i\) and \(j\) in the original network, \(\tilde{R}_{ij}\) is the path resistance of the simplified network, and \(\delta\) is a given tolerance. From a practical point of view such choice of error is useful since it helps to control condition (b). If \(n\) is large, then direct computation of (9) is expensive \(O(n^3)\) for each deleted resistor (or group of resistors). Therefore there is a need for the error bound which accurately enough estimates (9) and has less computational cost than the direct computation.

We define the problem of simplification of large resistor networks as follows: given a resistor network described by (1), find a reduced network that satisfy the above conditions (a), (d), (e) and the extra conditions

(f) has the same internal nodes;

(g) for a given tolerance condition (9) or (6) holds true.

Thus the difference between simplification and exact reduction of resistor networks is that after simplification the number of internal nodes, always, stays the same while the number of resistors may decrease and path resistances may differ from the original path resistances. Another issue that is subject to further research is that simplification can be obtained by decreasing the number of nodes in the network. For example, if a resistor between two nodes is relatively small, then current goes through such resistor without obstruction, and as a result, two nodes can be considered as one node. We will not consider this case since in considered practical examples it was not a case.

The idea of simplification (neglecting of resistors) is not new, see, for instance [7, 1, 5], however it has not been deeply developed. In [7], a simple criterion to simplify resistor networks has been suggested. The criterion is based on a physical intuition that large resistors do not affect the behavior of the network and, therefore, can be removed from the network. According to the criterion, a resistor between nodes \(i\) and \(j\) is removed if

\[
\frac{|G_{ij}|}{|G_{ii}|} < tol, \quad \text{and} \quad \frac{|G_{ij}|}{|G_{jj}|} < tol,
\]

(10)

where \(tol\) is a user defined tolerance. The disadvantage of the criterion is that the condition (6) or (9) is not controlled explicitly and, therefore, accuracy of the simplified network is not guaranteed.

We will consider simplification and reduction by ReduceR as totally independent and complementary procedures. We will distinguish different strategies: 1) simplification, 2) reduction, 3) simplification and then reduction, 4) reduction and then simplification. One may expect that simplification before reduction may improve further reduction because neglecting some resistors in the original network changes network topology. Note that the simplification procedure may disconnect the network or even lead to a singular matrix, therefore extra care should be taken to prevent such cases. In this paper we create a framework: when to simplify (before or after reduction) and how to simplify networks according to the criteria (9) and (6) in an efficient way.
4. ERROR ESTIMATION FOR $\frac{||v - \bar{v}||}{||v||}$

In this section we suggest an error estimation for the maximum relative error of vector of voltages which shows to be much more sharp than the error estimation based on the condition number (7). In order to derive it we consider:

$$
\max_{i \in A} \frac{||v - \bar{v}||}{||v||} = \max_{i \in A} \frac{||G^{-1}i - \tilde{G}^{-1}i||}{||G^{-1}i||} = \max_{i \in A} \frac{||(I - \tilde{G}^{-1}G)||}{||f||} (11)
$$

$$
\leq \max_{i \in \mathbb{R}^n, f \neq 0} \frac{||(I - \tilde{G}^{-1}G)||}{||f||} = \sigma_1, (12)
$$

where $f = G^{-1}i_n$, $\sigma_1$ is maximum singular value of $(I - \tilde{G}^{-1}G)$, and

$$
A = \{i \in \mathbb{R}^n | (i)_{k} = 1, (i)_{l} = -1, k, l \in \{1, \ldots, n_e\}, k \neq l \}. (13)
$$

Thus we obtain

$$
Err_v = \frac{||v - \bar{v}||}{||v||} \leq \max_{i \in \mathbb{R}^n, f \neq 0} \frac{||(I - \tilde{G}^{-1}G)||}{||f||} = (\lambda_{max} ((I - \tilde{G}^{-1}G)^T (I - \tilde{G}^{-1}G)))^{\frac{1}{2}} (14)
$$

$$
= \sigma_1 = Err_{vs}, (15)
$$

where $\lambda_{max}$ denotes the largest eigenvalue, and $Err_{vs}$ demonstrates estimation for the relative error $Err_v$. Computation of maximum singular value can be performed, for instance, by implicitly restarted Lanczos bidiagonalization methods [8], or Krylov-Schur method [9], which is used for numerical examples in section 7(C). We note that within the Krylov-Schur method there is no need to compute $(I - \tilde{G}^{-1}G)$ directly, one only requires to compute matrix-vector product of the form $(I - \tilde{G}^{-1}G)x$, which is performed by solving one linear system and one matrix-vector product. This requires $\tilde{G}$ to be nonsingular and therefore the network has to be grounded. If the network is not grounded, one can temporarily ground an arbitrary external node and after simplification unground it, i.e. to insert back the corresponding row and column in $G$. In section 6 we will exploit the estimation (15) for developing simplification algorithms which allow to delete resistors by groups while keeping $Err_v$ under control.

5. ERROR ESTIMATION FOR $\left| \frac{R_{ij} - \bar{R}_{ij}}{R_{ij}} \right|$

The path resistance between two nodes $i$ and $j$ is defined as the ratio of voltage across $i$ and $j$ to the current injected into them. In practice, path resistances from the input of one device to the output of one or more devices are used. Path resistances can be used for example for the analysis of the power dissipation, and in electro static discharge analysis to check whether unintended peak currents are conducted well enough through the resistive protection network to prevent damage to the chip [1].

The path resistance between ports $i$ and $j$ is defined as [10]

$$
R_{ij} = (e_i - e_j)^T G^{-1}(e_i - e_j), (16)
$$

where $e_i$ and $e_j$ are the $i$th and $j$th unit vectors, respectively. It is easy to show that path resistances of the original network, with conductance matrix $G$, are equal to the path resistances of the network obtained after elimination of (all) internal nodes. Indeed, if $G$ has been split into blocks as in (2) and $e_t^T = (\hat{e}_t \ 0)^T$ then

$$
R_{ij} = (e_i - e_j)^T G^{-1}(e_i - e_j) = (e_i - \hat{e}_j)^T (G_{11} - G_{12}G_{22}^{-1}G_{12}^T)^{-1}(e_i - \hat{e}_j) (17)
$$

$$
= (\hat{e}_i - \hat{e}_j)^T G_{ss}^{-1}(\hat{e}_i - \hat{e}_j), (18)
$$
which follows based on the inverse of $2 \times 2$ block matrix [11]. Note that if a network has only positive resistors, then all path resistances in the network are positive.

We will treat simplification of the network from the point of view that the path resistances of the simplified network should not differ much from the path resistances of the original network. Substituting (16) into (9) leads to

$$Err_p = \left| \frac{e_{ij}^T G^{-1} e_{ij} - e_{ij}^T \tilde{G}^{-1} e_{ij}}{e_{ij}^T G^{-1} e_{ij}} \right| < \delta, \quad (19)$$

where $e_{ij} = e_i - e_j$, $i, j = 1, \ldots, n_e$, and $i \neq j$.

We note that computing (19) directly is not an option, especially, if the number of terminals, $n_e$, is large. Therefore it is a good idea to have an estimation for (19), which is sharp enough and can be easily computed.

A grounded network guarantees that $G$ is positive definite ($G \succ 0$). If a network is not grounded, one can temporarily ground an arbitrary external nodes and after simplification to ungrounded it, i.e. to insert back corresponding row and column in $G$. Let $L$ be the Cholesky factor of $G$, i.e. $G = LL^T$, then $Err_p$ for arbitrary $i,j$, $(i,j, = 1, \ldots, n_e)$ is less or equal then the maximum relative error between all path resistances, i.e.,

$$Err_p \leq \max_{e_{ij} \in \mathcal{A}} \left| \frac{e_{ij}^T G^{-1} e_{ij} - e_{ij}^T \tilde{G}^{-1} e_{ij}}{e_{ij}^T G^{-1} e_{ij}} \right| = \max_{e_{ij} \in \mathcal{A}} \left| \frac{e_{ij}^T (G^{-1} - \tilde{G}^{-1}) e_{ij}}{e_{ij}^T L^{-T} L^{-1} e_{ij}} \right|, \quad (20)$$

where

$$\mathcal{A} = \{ e_{ij} \in \mathbb{R}^n | e_{ij} = 1, e_{ij} = -1, i, j = \{1, \ldots, n_e\}, i \neq j \}. \quad (21)$$

Setting up $y = L^{-1} e_{ij}$, one can rewrite (20) as follows

$$Err_p \leq \max_{e_{ij} \in \mathcal{A}} \left| \frac{e_{ij}^T (G^{-1} - \tilde{G}^{-1}) e_{ij}}{e_{ij}^T L^{-T} L^{-1} e_{ij}} \right| \leq \max_{y \in \mathbb{R}^n} \left| \frac{y^T L^{-T} (G^{-1} - \tilde{G}^{-1}) L y}{y^T y} \right| = \max(|\lambda_1|, |\lambda_n|), \quad (22)$$

where $\lambda_1$ and $\lambda_n$ are the largest and the smallest eigenvalues of

$$L^T (G^{-1} - \tilde{G}^{-1}) L. \quad (23)$$

The advantage of (22) is that it is independent of $e_{ij}$, however, it contains inverse matrices. In order to make computations efficient, we would like to get rid of the inverse matrices. First, we consider the following eigenvalue problem:

$$L^T (G^{-1} - \tilde{G}^{-1}) L x = \lambda x, \quad (24)$$

Multiplying (24) on the left side by $L^{-T}$ and setting up $w = Lx$, one obtains:

$$(G^{-1} - \tilde{G}^{-1}) w = \lambda L^{-T} L^{-1} w, \quad (25)$$

or, equivalently,

$$(G^{-1} - \tilde{G}^{-1}) w = \lambda G^{-1} w. \quad (26)$$

Multiplying (26) from the left on $G$ and setting up $z = \tilde{G}^{-1} w$, (26) becomes

$$(\tilde{G} - G) z = \lambda \tilde{G} z. \quad (27)$$

Thus $Err_p$ is approximated as follows:

$$Err_p \leq \max(|\lambda_1|, |\lambda_n|) \equiv Err_{pa}, \quad (28)$$

where $\lambda_1$ and $\lambda_n$ are the largest and the smallest eigenvalues of the generalized eigenvalue problem (27). $Err_{pa}$ is an error bound of $Err_p$ and requires to compute one largest magnitude eigenvalue.
Since not the full spectrum of eigenvalues is required, one can use iterative methods such as the Lanczos method [12], the Arnoldi method [13], and the Jacobi-Davidson method [14] that exploit the sparsity of the system to limit memory and CPU requirements. For numerical examples in section 7 we will use Matlab build-in function eigs which uses the Arnoldi method.

Computing the estimations $Err_{pa}$ in (27) and $Err_{vs}$ in (15) involves calculation of eigenvalues which are related to some extend. Multiplying (27) from the left by $\tilde{G}^{-1}$, the estimation $Err_{pa}$ requires computing the largest magnitude eigenvalue of the matrix $A := (I - \tilde{G}^{-1}G)$, while $Err_{vs}$ requires computing the maximum eigenvalue of $A^T A$. In the particular case of a symmetric matrix $A$, one obtains $\sqrt{\lambda_{\max}(A^T A)} = \lambda_{\max}(A)$, or, equivalently,

$$\sigma_{\max}(A) = \lambda_{\max}(A).$$  (29)

Since in our case $A$ is nonsymmetric the relation between $\sigma_{\max}(A)$ and $\lambda_{\max}(A)$ is non-trivial. Note that only those resistors can be considered for deleting which do not make $\tilde{G}$ singular. For example, $\tilde{G}$ becomes singular when network is disconnected. Therefore care should be taken that deleted resistor will not disconnect the network. This can be achieved by using graph algorithm which computes strongly connected components (scc) [15]. If the number of scc’s is larger than 1, then the network is disconnected.

Restriction on $G$ to be positive definite, is important because it allows us to use the Cholesky factorization and to keep the matrix pencil $(\tilde{G} - G, \tilde{G})$ regular. To demonstrate the last proposition, let $R$-network not be grounded, then the matrix pencil $(\tilde{G} - G, \tilde{G})$ is not regular, i.e., $G - G - \gamma \tilde{G}$ is singular for any $\gamma \in \mathbb{C}$:

$$(\tilde{G} - G - \lambda \tilde{G})x = 0.  \tag{30}$$

Suppose a resistor has been deleted, i.e., $\tilde{G}$ has been obtained from $G$ by a rank-one update:

$$\tilde{G} = G - ee^T.  \tag{31}$$

Substituting (31) into (30), one obtains

$$(-\lambda G + (\lambda - 1)ee^T)x = 0.  \tag{32}$$

If the network is not grounded (at least temporarily), then $G$ is singular and consists of stamps. Adding a rank-one matrix (which is also singular) will lead to a singular matrix for any $\lambda$. Therefore (32) will not have a unique solution. If the network is grounded, $G$ becomes positive definite, thus deleting any conductances which do not disconnect the network, will help to keep the pencil $(\tilde{G} - G, \tilde{G})$ regular.

In section 6 we will exploit the estimation (28) for developing simplification algorithms which allow to delete resistors by groups while keeping $Err_{pa}$ under control.

### 6. IMPLEMENTATION ISSUES

In this section we suggest a few algorithms for simplification of resistor networks which exploit computation of the error bounds $Err_{vs}$ in (15) or $Err_{pa}$ in (28). For convenience we will use $Err$ for denoting a generic error estimate, i.e. $Err_{vs}$, $Err_{pa}$ or any other. Each algorithm consists of two phases. In the first phase all conductors are sorted in increasing order. The next phase is selective strategy for deleting resistors. We have considered three alternatives for this.

1. **Take resistors one by one and for each compute $Err$. If computed $Err$ is greater than a bound $ErrMax$ more than $T_{max}$ times, then stop the procedure. Since this turns out to give a slow simplification, we will not consider this option any further.**

2. **Golden search I. Choose $k$, which is less than the number of all resistors. (For instance, $k$ can be chosen as $10\%$ from the whole amount of resistors.) Try to delete at once $k$ resistors and check whether network becomes disconnected. If the network is still connected, then compute $Err$. If $Err < \delta$, then try to delete the next $2k$ resistors, otherwise try to delete $k/2$ resistors. If the network is disconnected, then try to delete $h/2$ resistors. Continue the procedure till $Err > \delta$ and $h = 1$ appear $T_{max}$ times.**
(3) **Golden search 2.** It is the same procedure as "Golden search 1" but with a different stop criterion. Simplification has to be stopped immediately after $Err > \delta$ and $h = 1$ (parameter $T_{\text{max}} = 1$).

The **Golden search 2** is faster than **Golden search 1** though it usually allows to delete fewer resistors. To make the choice between these approaches, one needs to find a compromise between time and amount of resistors to delete.

The above algorithms are just some suggested algorithms for selecting resistors that are candidates to be eliminated. Within these algorithms an error bound, $Err$, is used to determine whether the error is below the prescribed tolerance $\delta$. The set of resistors to be eliminated is adapted for deleting as many resistors as possible, while keeping the error under control.

### 7. NUMERICAL RESULTS

We will show how the suggested above approach for simplification of resistor networks and reduction by ReduceR work for the networks from industry. The networks I, II and IV come from realistic designs of very-large-scale integration chips [1] and the network III comes from handlewafer model which has a specific structure similar to the one in the Fig. 1. The simplification algorithms based on the selective strategies (**Golden search 1** and **Golden search 2**) have been implemented in Matlab 7.5 and have been tested on Core 2 Duo 1.6 GHz PC. Since given resistor networks are not grounded, in order to compute $Err_p$ and $Err_p$, we initially ground the first terminal and after simplification (reduction) plug it back to the networks.

#### 7.1. **Simplification by $Err_p$ and $Err_{vs}$ applied to the original networks**

Table I compares results of simplification (option **Golden search 1**) applied to the original networks. In order to investigate how sharp $Err_p$ is in comparison to $Err_p$, we performed each selective strategy with $Err_p$ and $Err_p$ independently. It can be seen that simplification with $Err_p$ works faster than simplification with $Err_p$. However for the networks I and II, $Err_p$ appears less sharp than $Err_p$, since it allows to delete fewer resistors.

Table II shows a similar experiment but with the use of option **Golden search 2**. Golden Search 2 works faster than **Golden search 1** because it works till the first occurrence $Err_p > \delta$ and $k = 1$, and it allows, in general, to delete fewer resistors.

Table III shows the results obtained after applying simplification with $Err_{vs}$ to the original networks. For $\delta = 5\%$, simplification by $Err_{vs}$ works slower than simplification by $Err_p$. This happens due to the costs of computing the matrix vector product of the form $(I - \hat{G}^{-1}G)x$ within the Krylov-Schur method [9], which involves solving the system of the form $\hat{G}y = (Gx)$ for $y$. For the networks with large scale $G$ (e.g., network III), this step becomes time-consuming. Nevertheless estimation $Err_{vs}$ shows to be sharper than $Err_p$.

Figure (2) shows values of $Err_{vs} \equiv \sigma_{\text{max}} < 5\%$ and correspondent $Err_p \equiv \lambda_{\text{max}}$ computed on the sequence of conductance matrices obtained during **Golden search 2**. For the network III, 10222 resistors from 70006 have been deleted and 34 computations of $\sigma_{\text{max}}$ from 53 correspond to $\sigma_{\text{max}} < 5\%$. The plot demonstrates noticeable difference in computed values of $\sigma_{\text{max}}$ and $\lambda_{\text{max}}$, which is result of nontrivial relation between the error estimations.

From the above we conclude that simplification applied to the original networks is not recommended to be considered as independent reduction since the number of resistors is not decreased significantly. However for the network III simplification by $Err_p$ noticeably improves sparsity in reasonable time. Another observation is that with $\delta = 5\%$ simplification by $Err_{vs} \equiv \sigma_{\text{max}} > 5\%$ usually delivers better reduction in the amount of resistors than simplification by $Err_p$.

#### 7.2. **Simplification by $Err_p$ together with reduction**

We will show how simplification by $Err_p$ and $Err_p$ with **Golden search 1** approach works together with reduction by ReduceR. Tables IV-VII demonstrate results of simplification and reduction.
Table I. Results of simplification by $Err_p$ and $Err_{pa}$ (Golden search 1) applied to three networks (I,II,III). Table includes the original number of resistors, CPU time of simplification, the number of deleted resistors after simplification and the number of computations of $Err_p$ and $Err_{pa}$. $\delta = 5\%$, $T_{max} = 5$, $h = 50$

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>II</th>
<th>III</th>
</tr>
</thead>
<tbody>
<tr>
<td>#res. originally</td>
<td>23222</td>
<td>2476</td>
<td>70006</td>
</tr>
<tr>
<td>CPU time $Err_p$</td>
<td>272 s.</td>
<td>2.7 s.</td>
<td>1185 s.</td>
</tr>
<tr>
<td>CPU time $Err_{pa}$</td>
<td>3.2 s.</td>
<td>0.7 s.</td>
<td>208 s.</td>
</tr>
<tr>
<td>#deleted res.$Err_p$</td>
<td>576</td>
<td>29</td>
<td>9878</td>
</tr>
<tr>
<td>#deleted res.$Err_{pa}$</td>
<td>35</td>
<td>2</td>
<td>9878</td>
</tr>
<tr>
<td>#comput. $Err_p$</td>
<td>74</td>
<td>23</td>
<td>78</td>
</tr>
<tr>
<td>#comput. $Err_{pa}$</td>
<td>14</td>
<td>10</td>
<td>23</td>
</tr>
</tbody>
</table>

Table II. Results of simplification by $Err_p$ and $Err_{pa}$ (Golden search 2) applied to three networks (I,II,III). Table includes the original number of resistors, CPU time of simplification, the number of deleted resistors after simplification and the number of computations of $Err_p$ and $Err_{pa}$. $\delta = 5\%$, $T_{max} = 5$, $h = 50$

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>II</th>
<th>III</th>
</tr>
</thead>
<tbody>
<tr>
<td>#res. originally</td>
<td>23222</td>
<td>2476</td>
<td>70006</td>
</tr>
<tr>
<td>CPU time $Err_p$</td>
<td>245 s.</td>
<td>0.8 s.</td>
<td>377 s.</td>
</tr>
<tr>
<td>CPU time $Err_{pa}$</td>
<td>2.1 s.</td>
<td>0.5 s.</td>
<td>57.6 s.</td>
</tr>
<tr>
<td>#deleted res.$Err_p$</td>
<td>570</td>
<td>18</td>
<td>9695</td>
</tr>
<tr>
<td>#deleted res.$Err_{pa}$</td>
<td>35</td>
<td>2</td>
<td>9695</td>
</tr>
<tr>
<td>#comput. $Err_p$</td>
<td>64</td>
<td>6</td>
<td>78</td>
</tr>
<tr>
<td>#comput. $Err_{pa}$</td>
<td>9</td>
<td>5</td>
<td>23</td>
</tr>
</tbody>
</table>

Table III. Results of simplification by $Err_{vs}$ (Golden search 2) applied to three networks (I,II,III). Table includes the original number of resistors, CPU time of simplification, the number of deleted resistors after simplification, the number of computations of $Err_{vs}$. $\delta = 5\%$, $T_{max} = 5$, $h = 50$

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>II</th>
<th>III</th>
</tr>
</thead>
<tbody>
<tr>
<td>#res. originally</td>
<td>23222</td>
<td>2476</td>
<td>70006</td>
</tr>
<tr>
<td>CPU time $Err_{vs}$</td>
<td>23.4 s.</td>
<td>1.7 s.</td>
<td>1385 s.</td>
</tr>
<tr>
<td>#deleted res. $Err_{vs}$</td>
<td>35</td>
<td>2</td>
<td>10222</td>
</tr>
<tr>
<td>#comput. $Err_{vs}$</td>
<td>9</td>
<td>5</td>
<td>53</td>
</tr>
</tbody>
</table>

Table IV. Results of simplification by $Err_{pa}$ ($S_1$), $Err_p$ ($S_d$), $Err_{vs}$ ($S_2$) and reduction by ReduceR (R) of the network I. $\delta = 5\%$, $T_{max} = 5$

<table>
<thead>
<tr>
<th></th>
<th>Original</th>
<th>$R$</th>
<th>$S_1 + R$</th>
<th>$R + S_1$</th>
<th>$R + S_d$</th>
<th>$R + S_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td># terminals</td>
<td>160</td>
<td>160</td>
<td>160</td>
<td>160</td>
<td>160</td>
<td>160</td>
</tr>
<tr>
<td># int nodes</td>
<td>12661</td>
<td>138</td>
<td>157</td>
<td>138</td>
<td>138</td>
<td>138</td>
</tr>
<tr>
<td># resistors</td>
<td>23222</td>
<td>3315</td>
<td>3315</td>
<td>1359</td>
<td>1244</td>
<td>1187</td>
</tr>
<tr>
<td>CPU time</td>
<td>-</td>
<td>23.8 s.</td>
<td>31.2 s.</td>
<td>25.7 s.</td>
<td>229 s.</td>
<td>28 s.</td>
</tr>
</tbody>
</table>
Figure 2. For the network III: values of $\sigma_{max} < 5\%$ and correspondent values of $\lambda_{max}$ which were computed during performance of Golden search 2

Table V. Results of simplification by $Err_{pa} (S_1)$, $Err_p (Sd)$, $Err_{vs} (S_2)$ and reduction by $ReduceR$ (R) of the network II. $\delta = 5\%$, $T_{max} = 5$

<table>
<thead>
<tr>
<th></th>
<th>Original</th>
<th>$R$</th>
<th>$S_1 + R$</th>
<th>$R + S_1$</th>
<th>$R + Sd$</th>
<th>$R + S_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td># terminals</td>
<td>39</td>
<td>39</td>
<td>39</td>
<td>39</td>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td># int nodes</td>
<td>1503</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td># resistors</td>
<td>2476</td>
<td>702</td>
<td>702</td>
<td>589</td>
<td>541</td>
<td>510</td>
</tr>
<tr>
<td>CPU time</td>
<td>-</td>
<td>1 s.</td>
<td>1.7 s.</td>
<td>1.7 s.</td>
<td>6.2 s.</td>
<td>1.9 s.</td>
</tr>
</tbody>
</table>

Table VI. Results of simplification by $Err_{pa} (S_1)$, $Err_p (Sd)$, $Err_{vs} (S_2)$ and reduction by $ReduceR$ (R) of the network III. $\delta = 5\%$, $T_{max} = 5$

<table>
<thead>
<tr>
<th></th>
<th>Original</th>
<th>$R$</th>
<th>$S_1 + R$</th>
<th>$R + S_1$</th>
<th>$R + Sd$</th>
<th>$R + S_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td># terminals</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td># int nodes</td>
<td>31356</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td># resistors</td>
<td>70006</td>
<td>1485</td>
<td>1485</td>
<td>1480</td>
<td>1479</td>
<td>455</td>
</tr>
<tr>
<td>CPU time</td>
<td>-</td>
<td>62.4 s.</td>
<td>107 s.</td>
<td>68.8 s.</td>
<td>65.7 s.</td>
<td>79 s.</td>
</tr>
</tbody>
</table>

Table VII. Results of simplification by $Err_{pa} (S_1)$, $Err_p (Sd)$, $Err_{vs} (S_2)$ and reduction by $ReduceR$ (R) of the network IV. $\delta = 5\%$, $T_{max} = 5$

<table>
<thead>
<tr>
<th></th>
<th>Original</th>
<th>$R$</th>
<th>$S_1 + R$</th>
<th>$R + S_1$</th>
<th>$R + Sd$</th>
<th>$R + S_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td># terminals</td>
<td>76</td>
<td>76</td>
<td>76</td>
<td>76</td>
<td>76</td>
<td>76</td>
</tr>
<tr>
<td># int nodes</td>
<td>1134</td>
<td>383</td>
<td>308</td>
<td>308</td>
<td>308</td>
<td>308</td>
</tr>
<tr>
<td># resistors</td>
<td>1936</td>
<td>1397</td>
<td>1397</td>
<td>1269</td>
<td>1269</td>
<td>1264</td>
</tr>
<tr>
<td>CPU time</td>
<td>-</td>
<td>1.12 s.</td>
<td>1.7 s.</td>
<td>2.1 s.</td>
<td>14 s.</td>
<td>2.8 s.</td>
</tr>
</tbody>
</table>

by $ReduceR$ applied in different combinations: 1) reduction, 2) simplification by $Err_{pa}$ and then reduction, 3) reduction and then simplification by $Err_{pa}$, and 4) reduction and then simplification by $Err_p$. 
Simplification applied after reduction works better than before reduction. This happens because simplification does not make crucial changes in the topology of the networks which can be recognized by ReduceR. Thus combinations 3 and 4 are, in general, better in the amount of resistors than the combination 2. When a reduced network has many internal nodes and terminals (see tables IV, VII), combination 3 shows better compromise between the amount of deleted resistors and time than combination 4. In this case direct computation of $\text{Err}_p$ becomes very time-consuming. If the number of terminals and internal nodes is not large (tables V, VI), then combination 4 is preferable since it allows to delete more resistors in small extra time.

Simplification of a reduced network with many internal nodes is, in general, more efficient than simplification of the reduced network with only a few internal nodes: the more internal nodes, the more options for neglecting resistors which do not affect path resistances. This explains why the simplification after reduction (in the amount of resistors) of the networks I and IV is better than in the case of the networks II and III.

The larger parameter $T_{max}$, the more resistors can be deleted and more time is required. For the network IV (table VII), we used $T_{max} = 5$. This allowed us to delete after reduction 128 resistors in 0.9 sec., while with $T_{max} = 80$ one could delete 153 resistors in 4 sec.

Fig. 3 confirms that after simplification by $\text{Err}_{pa}$, the relative error of path resistances (9) stays smaller than $\delta = 5\%$.

![Figure 3. Comparison of computed error $\text{Err}_p$ with 5\% error for each path resistance for the networks I (left) and III (right).](image)

7.3. Simplification by $\text{Err}_{us}$ together with reduction

In the previous section we have shown that simplification applied after reduction leads to fewer resistors than simplification applied before reduction. Therefore in this section we consider only applying simplification by $\text{Err}_{us}$ after reduction by ReduceR. Tables IV-VII demonstrate that simplification applied after reduction by ReduceR allows to delete more resistors in reasonable extra time than simplification by $\text{Err}_{pa}$ and $\text{Err}_p$ applied after reduction. Noticeable reduction has been achieved for the networks I and III, where the number of resistors has been decreased by 65\% and 70\% respectively.

For the purpose of illustration, Figure (4) shows values of $\text{Err}_{us} \equiv \sigma_{max} < 5\%$ applied after ReduceR and correspondent values of $\text{Err}_{pa} \equiv \lambda_{max}$ computed on the sequence of conductance matrices obtained during Golden search 1. One can observe that $\lambda_{max}$ increases monotonically. This happens due to the fact that path resistance is a monotonic function [10]. To show it let $g$ and $\tilde{g}$ ($0 \leq g \leq \tilde{g}$) be conductances and $G, \tilde{G}$ be correspondent conductance matrices. Since $G \preceq \tilde{G}$, then $e_i^T G^{-1} e_j \geq e_i^T \tilde{G}^{-1} e_j$, i.e. $R_{ij} \geq \tilde{R}_{ij}$. Therefore deleting each new resistor from $G$ makes the relative error, $\text{Err}_p$, in (19) and its estimation, $\text{Err}_{pa}$, non-decreasing functions.
8. CONCLUSIONS

In this paper we have considered approach for reduction of resistors in resistor networks. The suggested approach, so called simplification, can improve the sparsity of the original and/or reduced conductance matrix by neglecting resistors which do not contribute significantly to the behavior of the circuit. Two criteria for measuring the quality of approximation have been suggested and corresponding error bounds have been derived. Obtained error bounds demand less computational effort than the direct error computations and, thus, constitutes the base for simplification algorithms. The considered simplification algorithms, applied after reduction by ReduceR, improved total reduction by 70%. Since the success of simplification depends on the values of conductances in resistor networks, simplification can be considered as a complementary procedure to existing exact reduction techniques.

ACKNOWLEDGEMENTS

The authors would like to thank M. Hochstenbach for sharing the Krylov-Schur SVD code. The first author also wants to thank L. Reichel and P.I. Rosen Esquivel for the useful discussions.

REFERENCES

### Previous Publications in This Series:

<table>
<thead>
<tr>
<th>Number</th>
<th>Author(s)</th>
<th>Title</th>
<th>Month</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-46</td>
<td>S.S. Antman, D. Bourne</td>
<td>Rotational symmetry vs. axisymmetry in Shell theory</td>
<td>Sept. '10</td>
</tr>
</tbody>
</table>