EINDHOVEN UNIVERSITY OF TECHNOLOGY
Department of Mathematics and Computer Science

CASA-Report 10-49
September 2010

Error bounds for reduction of multi-port resistor networks

by

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

Centre for Analysis, Scientific computing and Applications
Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513
5600 MB Eindhoven, The Netherlands
ISSN: 0926-4507

Error Bounds for Reduction of Multi-Port Resistor Networks

M. V. Ugryumova[1], J. Rommes[2], W. H. A. Schilders[3]

[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).*
[3] *Eindhoven University of Technology, and NXP Semiconductors, Eindhoven, The Netherlands.*

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]:

$$G\mathbf{v} = \mathbf{i}, \tag{1}$$

where $G = \in \mathbb{R}^{n\times n}$ is symmetric positive semidefinite conductance matrix ($G \succeq 0$), $\mathbf{v} \in \mathbb{R}^n$ are the node voltages, and $\mathbf{i} \in \mathbb{R}^n$ 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:

$$\begin{pmatrix} G_{11} & G_{12} \\ G_{12}^T & G_{22} \end{pmatrix} \begin{pmatrix} \mathbf{v}_e \\ \mathbf{v}_i \end{pmatrix} = \begin{pmatrix} B \\ 0 \end{pmatrix} \mathbf{i}_e, \tag{2}$$

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 $(n = n_e + n_i)$, $\mathbf{i}_e \in \mathbb{R}^n_e$ 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 $\mathbf{i}_e(k) = j$. If current is only injected in a terminal a, then $B_{a,k} = 1$ and $\mathbf{i}_e(k) = j$. 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

$$\tilde{G} = G - (\mathbf{e}_a - \mathbf{e}_b)g(\mathbf{e}_a - \mathbf{e}_b)^T, \tag{3}$$

where $\mathbf{e}_a$ and $\mathbf{e}_b$ 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 $(\mathbf{e}_a - \mathbf{e}_b)^T g(\mathbf{e}_a - \mathbf{e}_b)$ is a stamp corresponding to the conductance g. Introducing the notation

$$(\mathbf{e}_a - \mathbf{e}_b) = \mathbf{m}, \tag{4}$$

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

$$G = \sum_{i=1}^{N} \mathbf{m}_i g_i \mathbf{m}_i^T. \tag{5}$$

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 $\hat{n}_i \ll n_i$ internal nodes;
(d) has $\hat{N} \ll N$ 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 $\hat{N} = (n_e^2 - n_e)/2$ 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.

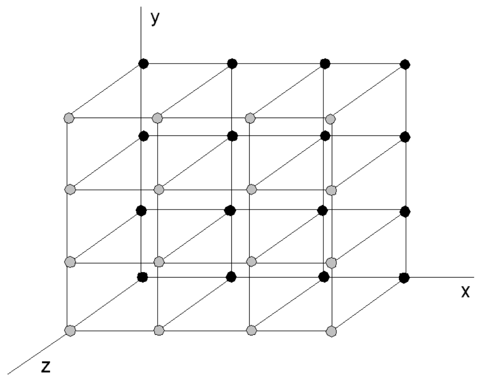


Figure 1. Network with 16 external nodes (black dots), 16 internal nodes and 64 resistors (edges of the cubes).

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:

$$Err_v := \frac{||\mathbf{v} - \tilde{\mathbf{v}}||}{||\mathbf{v}||} = \frac{||G^{-1}\mathbf{i} - \tilde{G}^{-1}\mathbf{i}||}{||G_s^{-1}\mathbf{i}||} < \epsilon, \tag{6}$$

i.e. for a given ϵ 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, $\mathbf{i}$ is unknown in advance, computing (6) requires the knowledge of an error bound (estimation) which must be independent of $\mathbf{i}$. An error estimation based on the condition number of G, $\kappa(G)$, can be derived, for instance, based on the approach presented in [5]:

$$\frac{||\mathbf{v}-\tilde{\mathbf{v}}||}{||\mathbf{v}||} \leq \kappa(G)\cdot tol = \epsilon, \tag{7}$$

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

$$||G-\tilde{G}|| \leq tol \cdot ||G||. \tag{8}$$

Thus for a given ϵ 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.,

$$Err_p := \left|\frac{R_{ij}-\tilde{R}_{ij}}{R_{ij}}\right| < \delta,\ i,j=1,\ldots,n_e, \tag{9}$$

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 δ 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

$$\frac{|G_{ij}|}{|G_{ii}|} < tol, \text{ and } \frac{|G_{ij}|}{|G_{jj}|} < tol, \tag{10}$$

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 $\frac{||\mathbf{v}-\tilde{\mathbf{v}}||}{||\mathbf{v}||}$

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:

$$\max_{i\in\mathcal{A}} \frac{||\mathbf{v}-\tilde{\mathbf{v}}||_2}{||\mathbf{v}||_2} = \max_{i\in\mathcal{A}} \frac{||G^{-1}\mathbf{i}-\tilde{G}^{-1}\mathbf{i}||_2}{||G^{-1}\mathbf{i}||_2} = \max_{i\in\mathcal{A}} \frac{||(I-\tilde{G}^{-1}G)\mathbf{f}||_2}{||\mathbf{f}||_2} \tag{11}$$

$$\le \max_{i\in\mathbb{R}^n,\mathbf{i}\neq 0} \frac{||(I-\tilde{G}^{-1}G)\mathbf{f}||_2}{||\mathbf{f}||_2} = \sigma_1, \tag{12}$$

where $\mathbf{f} = G^{-1}\mathbf{i}_n$, σ_1 is maximum singular value of $(I-\tilde{G}^{-1}G)$, and

$$\mathcal{A} = \{\mathbf{i}\in\mathbb{R}^n | (\mathbf{i})_k = 1, (\mathbf{i})_l = -1, k,l\in\{1,\dots,n_e\}, k\neq l\}. \tag{13}$$

Thus we obtain

$$Err_v = \frac{||\mathbf{v}-\tilde{\mathbf{v}}||_2}{||\mathbf{v}||_2} \le \max_{\mathbf{i}\in\mathbb{R}^n,\mathbf{i}\neq 0} \frac{||(I-\tilde{G}^{-1}G)\mathbf{f}||_2}{||\mathbf{f}||_2} = \left(\lambda_{max}\left((I-\tilde{G}^{-1}G)^T(I-\tilde{G}^{-1}G)\right)\right)^{\frac{1}{2}} \tag{14}$$

$$= \sigma_1 = Err_{vs}, \tag{15}$$

where λ_{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 $(I-\tilde{G}^{-1}G)\mathbf{x}$, which is perform 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 $\left|\frac{R_{ij}-\tilde{R}_{ij}}{R_{ij}}\right|$

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]

$$R_{ij} = (\mathbf{e}_i-\mathbf{e}_j)^T G^{-1}(\mathbf{e}_i-\mathbf{e}_j), \tag{16}$$

where $\mathbf{e}_i$ and $\mathbf{e}_j$ are the ith and jth 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 $\mathbf{e}_t^T = \begin{pmatrix} \tilde{\mathbf{e}}_t & 0 \end{pmatrix}^T$ then

$$R_{ij} = (\mathbf{e}_i-\mathbf{e}_j)^T G^{-1}(\mathbf{e}_i-\mathbf{e}_j) = (\tilde{\mathbf{e}}_i-\tilde{\mathbf{e}}_j)^T(G_{11}-G_{12}G_{22}^{-1}G_{12}^T)^{-1}(\tilde{\mathbf{e}}_i-\tilde{\mathbf{e}}_j) \tag{17}$$

$$= (\tilde{\mathbf{e}}_i-\tilde{\mathbf{e}}_j)^T G_s^{-1}(\tilde{\mathbf{e}}_i-\tilde{\mathbf{e}}_j), \tag{18}$$

which follows based on the inverse of 2×2 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

$$Err_p = \left| \frac{\mathbf{e}_{ij}^T G^{-1}\mathbf{e}_{ij} - \mathbf{e}_{ij}^T \tilde{G}^{-1}\mathbf{e}_{ij}}{\mathbf{e}_{ij}^T G^{-1}\mathbf{e}_{ij}} \right| < \delta, \tag{19}$$

where $e_{ij} = \mathbf{e}_i - \mathbf{e}_j,\ i, j = 1, \ldots, n_e,$ 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.,

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

where

$$\mathcal{A} = \left\{ \mathbf{e}_{ij} \in \mathbb{R}^n | \mathbf{e}_{ij} = 1, \mathbf{e}_{ij} = -1, i, j \in \{1, \ldots, n_e\}, i \neq j \right\}. \tag{21}$$

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

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

where λ_1 and λ_n are the largest and the smallest eigenvalues of

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

The advantage of (22) is that it is independent of $\mathbf{e}_{ij}$, 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:

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

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

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

or, equivalently,

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

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

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

Thus Err_p is approximated as follows:

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

where λ_1 and λ_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^TA. In the particular case of a symmetric matrix A, one obtains $\sqrt{\lambda_{max}(A^TA)} = \lambda_{max}(A)$, or, equivalently,

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

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., $\tilde{G} - G - \gamma\tilde{G}$ is singular for any $\gamma \in \mathbb{C}$:

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

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

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

Substituting (31) into (30), one obtains

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

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 λ. 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_p 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 1*. 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 $T_{max} = 1$).

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 δ. 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_{pa} and Err_p, we initially ground the first terminal and after simplification (reduction) plug it back to the networks.

7.1. *Simplification by Err_{pa} 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_{pa} is in comparison to Err_p, we performed each selective strategy with Err_{pa} and Err_p independently. It can be seen that simplification with Err_{pa} works faster than simplification with Err_p. However for the networks I and II, Err_{pa} 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 $Err_{pa} > \delta$ *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_{pa}. This happens due to the costs of computing the matrix vector product of the form $(I - \tilde{G}^{-1}G)\mathbf{x}$ within the Krylov-Schur method [9], which involves solving the system of the form $\tilde{G}\mathbf{y} = (G\mathbf{x})$ for $\mathbf{y}$. 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_{pa}.

Figure (2) shows values of $Err_{vs} \equiv \sigma_{max} < 5\%$ and correspondent $Err_{pa} \equiv \lambda_{max}$ 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 σ_{max} from 53 correspond to $\sigma_{max} < 5\%$. The plot demonstrates noticeable difference in computed values of σ_{max} and λ_{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_{pa} noticeably improves sparsity in reasonable time. Another observation is that with $\delta = 5\%$ simplification by Err_{vs} usually delivers better reduction in the amount of resistors than simplification by Err_{pa}.

7.2. *Simplification by Err_{pa} together with reduction*

We will show how simplification by Err_p and Err_{pa} 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}. $\delta = 5\%$, $Tmax = 5$, $h = 50$

	I	II	III
#res. originally	23222	2476	70006
CPU time Err_p	272 s.	2.7 s.	1185 s.
CPU time Err_{pa}	3.2 s.	0.7 s.	208 s.
#deleted res.Err_p	576	29	9878
#deleted res.Err_{pa}	35	2	9878
#comput. Err_p	74	23	78
#comput. Err_{pa}	14	10	23

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}. $\delta = 5\%$, $Tmax = 5$, $h = 50$

	I	II	III
#res. originally	23222	2476	70006
CPU time Err_p	245 s.	0.8 s.	377 s.
CPU time Err_{pa}	2.1 s.	0.5 s.	57.6 s.
#deleted res.Err_p	570	18	9695
#deleted res.Err_{pa}	35	2	9695
#comput. Err_p	64	6	78
#comput. Err_{pa}	9	5	23

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}. $\delta = 5\%$, $Tmax = 5$, $h = 50$

	I	II	III
#res. originally	23222	2476	70006
CPU time Err_{vs}	23.4 s.	1.7 s.	1385 s.
#deleted res. Err_{vs}	35	2	10222
#comput. Err_{vs}	9	5	53

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

	$Original$	R	$S_1 + R$	$R + S_1$	$R + Sd$	$R + S_2$
# terminals	160	160	160	160	160	160
# int nodes	12661	138	157	138	138	138
# resistors	23222	3315	3315	1359	1244	1187
CPU time	-	23.8 s.	31.2 s.	25.7 s.	229 s.	28 s.

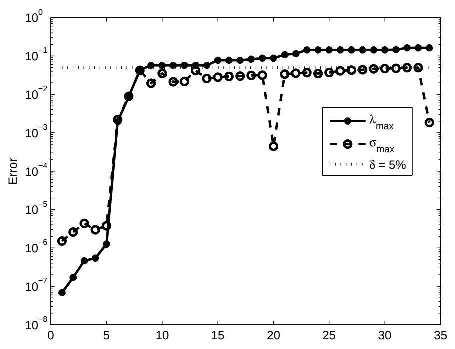


Figure 2. For the network III: values of $\sigma_{max} < 5\%$ and correspondent values of λ_{max} which were computed during performance of *Golden search 2*

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. $\delta = 5\%$, $T_{max} = 5$

	$Original$	R	$S_1 + R$	$R + S_1$	$R + Sd$	$R + S_2$
# terminals	39	39	39	39	39	39
# int nodes	1503	8	8	8	8	8
# resistors	2476	702	702	589	541	510
CPU time	-	1 s.	1.7 s.	1.7 s.	6.2 s.	1.9 s.

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. $\delta = 5\%$ $T_{max} = 5$

	$Original$	R	$S_1 + R$	$R + S_1$	$R + Sd$	$R + S_2$
# terminals	55	55	55	55	55	55
# int nodes	31356	0	0	0	0	0
# resistors	70006	1485	1485	1480	1479	455
CPU time	-	62.4 s.	107 s.	68.8 s.	65.7 s.	79 s.

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. $\delta = 5\%$ $T_{max} = 5$

	$Original$	R	$S_1 + R$	$R + S_1$	$R + Sd$	$R + S_2$
# terminals	76	76	76	76	76	76
# int nodes	1134	383	308	308	308	308
# resistors	1936	1397	1397	1269	1269	1264
CPU time	-	1.12 s.	1.7 s.	2.1 s.	14 s.	2.8 s.

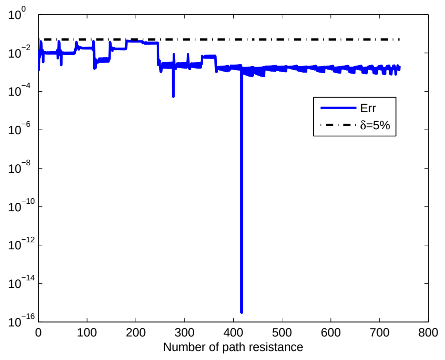
by *ReduceR* applied in different combinations: 1) reduction, 2) simplification by Err_{pa} and then reduction, 3) reduction and then simplification by Err_{pa}, and 4) reduction and then simplification by Err_p.

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 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 Err_{pa}, the relative error of path resistances (9) stays smaller than $\delta = 5\%$.



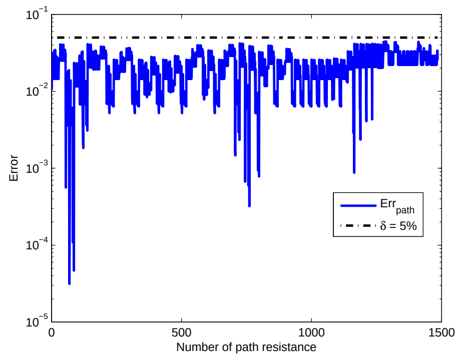
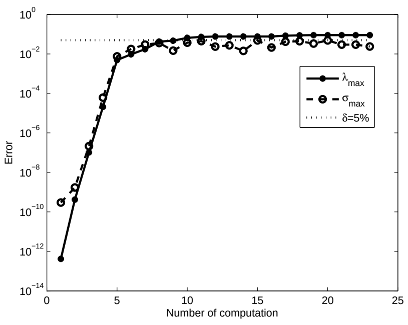


Figure 3. Comparison of computed error Err_p with 5% error for each path resistance for the networks I (left) and III (right).

7.3. *Simplification by Err_{vs} 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 Err_{vs} 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 Err_{pa} and 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 $Err_{vs} \equiv \sigma_{max} < 5\%$ applied after *ReduceR* and correspondent values of $Err_{pa} \equiv \lambda_{max}$ computed on the sequence of conductance matrices obtained during *Golden search 1*. One can observe that λ_{max} increases monotonically. This happens due to the fact that path resistance is a monotonic function [10]. To show it let g and $\tilde{g}$ $(0 \le g \le \tilde{g})$ be conductances and G, $\tilde{G}$ be correspondent conductance matrices. Since $G \preceq \tilde{G}$, then $\mathbf{e}_{ij}^T G^{-1} \mathbf{e}_{ij} \succeq \mathbf{e}_{ij}^T \tilde{G}^{-1} \mathbf{e}_{ij}$, i.e. $R_{ij} \ge \tilde{R}_{ij}$. Therefore deleting each new resistor from G makes the relative error, Err_p, in (19) and its estimation, Err_{pa}, non-decreasing functions.



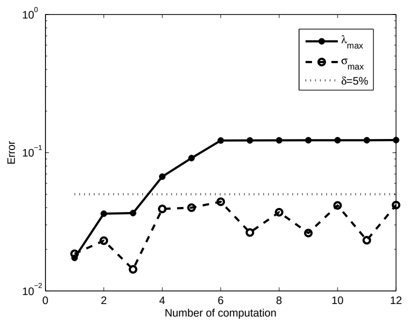


Figure 4. For the network I (left) and II (right): values of $\sigma_{max} < 5\%$ and correspondent values of λ_{max} which were computed during performance of *Golden search 1*

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

1. Rommes J, Schilders WHA. *Efficient methods for large resistor networks.* IEEE Trans. on CAD Circ. Syst. 2010; *29*(1):28–39.
2. Genderen AJ van. *Reduced Models for the Behavior of VLSI Circuits.* Ph.D dissertation, Delft University of Technology: Delft, 1991.
3. Vlach J, Singhal K. *Computer methods for Circuit Analysis and Design* (2nd edn). Van Nostrand Reinhold: New York; 1994; 32–39.
4. Schrik E, Meijs NP van der. *Combined BEM/FEM substrate resistance modeling.* Proceedings of the 39th conference on Design automation, June 10-14, 2002, New Orleans, Louisiana, USA.
5. Yang F, Zeng Y, Su Y, Zhou D. *RLC equivalent circuit synthesis method for structure-preserved reduced-order model of interconnect in VLSI.* Commun. comput. Phys. 2008; *3*(2):376–396.
6. Ugryumova M, Rommes J, Schilders WHA. *On approximate reduction of multi-port resistor networks.* To appear in SCEE 2010 Book of abstracts.
7. Lenaers P. *Model order reduction for large resistive networks. Final Report.* Technische Universiteit Eindhoven: Eindhoven, 2008.
8. Baglama J, Reichel L. *Augmented implicitly restarted Lanczos bidiagonalization methods.* SIAM J. Sci. Comp. 2005; *27*:19–42.
9. Stoll M. *A Krylov-Schur approach to the truncated SVD.* preprint submitted to Elsevier; 2010.
10. Ghosh A, Boyd S, Saberi A. *Minimizing effective resistance of a graph.* SIAM Rev. 2008; *50*(1):37–66.
11. Horn RA, Johnson CR. *Matrix Analysis* Cambridge University Press: Cambridge, 1985.
12. Parlett B. *The symmetric eigenvalue problem* Ser. Classics in Applied Mathematics. SIAM, 1998.
13. Lehoucq R, Sorensen D. *Deflation techniques within an implicitly restarted Arnoldi iteration.* SIAM J. Matrix Anal. Appl. 1996; *17*:789–821.
14. Fokkema DR, Sleijpen GLG, van der Vorst HA. *Jacobi-Davidson style QR and QZ algorithms for the reduction of matrix pensils.* SIAM J. Sci. Comp. 1998; *20*(1):94–125.
15. Gleich D. *MatlabBGL: A Matlab Library.* http://www.stanford.edu/ dgleich/programs/matlab-bgl [12 April 2010].

PREVIOUS PUBLICATIONS IN THIS SERIES:

Number	Author(s)	Title	Month
10-45	M.A.T. van Hinsberg J.H.M. ten Thije Boonkkamp H.J.H. Clercx	An efficient, second order method for the approximation of the Basset history force	July '10
10-46	S.S. Antman D. Bourne	Rotational symmetry vs. axisymmetry in Shell theory	Sept. '10
10-47	M. Pisarenco J.M.L. Maubach I. Setija R.M.M. Mattheij	A numerical method for the solution of time-harmonic Maxwell equations for two-dimensional scatterers	Sept. '10
10-48	L.M.J. Florack E. Balmashnova L.J. Astola E.J.L. Brunenberg	A new tensorial framework for single-shell high angular resolution diffusion imaging	Sept. '10
10-49	M.V. Ugryumova J. Rommes W.H.A. Schilders	Error bounds for reduction of multi-port resistor networks	Sept. '10

Ontwerp: de Tantes,
Tobias Baanders, CWI