Numerical Algorithms for Symbolic Analysis of Large Circuits

ZDENĚK KOLKA, MARTIN HORÁK, VIERA BIOLKOVÁ
Department of Radio Electronics
Brno University of Technology
Purkyňova 118, 612 00 Brno
CZECH REPUBLIC

Abstract: - This paper presents an algorithm for fast large-change sensitivity computation for approximate symbolic analysis. The sensitivity is used for identifying negligible elements of circuit model that can be removed in order to simplify symbolic solution. The method is based on combination of the Sherman-Morison formula and sparse matrix techniques. Effectiveness of the method was tested on several benchmark circuits.

Key-Words: - Approximate Symbolic Analysis, Numerical Algorithms, CAD

1 Introduction

One of the most effective methods for generating simplified symbolic expressions in the frequency domain is a method known as SBG (Simplification Before Generation). It is based on a heuristic algorithm consisting in removing elements of circuit model whose influence to a network function is negligible [5]. The process is always based on the knowledge of numerical solution whose changes provide a measure for identification of negligible elements. Any change of circuit model introduces an error. The simplification stops when the maximum allowable error is reached. A symbolic solution is then generated for the simplified circuit model.

The algorithms described perform gradual removing of parameters of network elements from the circuit matrix, i.e. two-terminal elements are replaced by short- or open-circuit, controlled sources are removed entirely or replaced by an ideal operating amplifier, etc.

Usually, large-change sensitivity is used to find a candidate for removing. The algorithm described in [2] performs an effective error prediction using algebraic cofactors. These cofactors can be easily obtained from inverse of the nodal admittance matrix. This paper compares effectiveness of the Sherman-Morison formula [3] for obtaining inverse matrix with sparse matrix techniques to find an optimum strategy.

2 Error Prediction with Inverse Matrix

Any linear circuit can be described in the frequency domain by a system of linear equations

$$\mathbf{A}\mathbf{x} = \mathbf{b} \ , \qquad (1)$$

where **A** is circuit matrix, **b** is vector of sources and **x** is vector of unknowns. Any network function F can be expressed as a ratio of two algebraic cofactors of **A**

$$F = (-1)^{\alpha} \frac{\det(\mathbf{A}_{\mathrm{N}})}{\det(\mathbf{A}_{\mathrm{D}})} \ , \qquad (2)$$

where $\mathbf{A}_{\mathrm{N}}$ and $\mathbf{A}_{\mathrm{D}}$ are submatrices of **A** and coefficient α is an integer. The exact procedure for obtaining $\mathbf{A}_{\mathrm{N}}$, $\mathbf{A}_{\mathrm{D}}$ and α depends on formulation method and required network function. Let a be an element of **A**. Both determinants can be expanded by the element into

$$F = (-1)^{\alpha} \frac{a\Delta_{i:j}^{N} + R_{\mathrm{N}}}{a\Delta_{k:l}^{D} + R_{\mathrm{D}}} \ , \qquad (3)$$

where $\Delta_{i:j}^{N}$ and $\Delta_{k:l}^{D}$ are algebraic cofactors of $\mathbf{A}_{\mathrm{N}}$ and $\mathbf{A}_{\mathrm{D}}$, and R_{N} and R_{D} represents remaining terms in the numerator and denominator. For the nodal formulation method a circuit parameter can appear in one, two or four positions in **A**. Two latter cases are also covered by (3) using the 2nd order cofactors.

It is obvious that large-change sensitivity of F to $a = 0$ or $a \to \infty$ can be simply obtained from (3). To test all elements we need all cofactors of **A**. They can be simply computed by means of numerical inversion of $\mathbf{A}_{\mathrm{N}}$ and $\mathbf{A}_{\mathrm{D}}$ [2]. Matrix of algebraic cofactors of an arbitrary regular matrix **B** is

$$\mathbf{B}_{\Delta} = \det(\mathbf{B})\left(\mathbf{B}^{-1}\right)^{T} \ . \qquad (4)$$

After an element removal we obtain new matrices that should be inverted again. This represents the most demanding part of symbolic simplification.

3 Fast Matrix Inversion

The Sherman-Morrison formula [3] can be used for computing a new inverse matrix from the old inverse matrix after changing some elements. It can be expressed by

$$\left(\mathbf{A}+\mathbf{u}\mathbf{v}^{\mathrm{T}}\right)^{-1}=\mathbf{A}^{-1}-\frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^{\mathrm{T}}\mathbf{A}^{-1}}{1+\mathbf{v}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{u}}\ , \tag{5}$$

where and **u** and **v** are arbitrary column vectors whose product defines the matrix change. In case of a single element of **A** $\mathbf{u}=[0..0,1,0..0]^T$ and $\mathbf{v}=[0..0,1,0..0]^T$.

The inverse matrix can also be computed by the modified Gaussian elimination that manipulates with nonzero elements only. Effectiveness of this technique significantly depends on positions of nonzero elements in the matrix. For this reason, the Markowitz criterion [4] for reordering columns and rows of the matrix that minimizes fill-ins during the elimination process was applied.

To find an optimum method, six algorithms using the nodal method were implemented in C++ to achieve the best performance:

a) Algorithm SBG1 uses the Gaussian elimination with full pivoting in all cases.
b) Algorithm SBG2 uses the Gaussian elimination with full pivoting when the eliminated element is substituted by zero impedance and is located on more than one position in the nodal admittance matrix. Otherwise, the Sherman-Morrison formula is used.
c) Algorithm SBG3 is similar to SBG2 but the elimination is performed only if the element is located on more than two positions in the nodal admittance matrix.
d) Algorithm SBG4 uses the Sherman-Morrison formula in all cases.
e) Algorithm SBG5 computes the inverse matrix using sparse techniques in all cases.
f) Algorithm SBG6 computes the inverse matrix using sparse techniques if the eliminated element is substituted by zero impedance and is located on more than one position in the nodal admittance matrix, otherwise the Sherman-Morrison formula is used.

Elements a_i are removed from circuit matrix **A** until the maximum error $\varepsilon_{\max}$ is reached. The error of simplification is usually checked at several frequencies f_j. The algorithm is the same for all methods

```
initialize matrices $\mathbf{A}_\mathrm{N}$ and $\mathbf{A}_\mathrm{D}$
calculate $\mathbf{A}_N^{-1}$ and $\mathbf{A}_D^{-1}$
calculate reference value(s) $F(f_j)$
do (main loop)
    $\varepsilon = \min_{\substack{for\ all\ f_j \\ for\ all\ elements}} (\varepsilon(a_i \to 0), \varepsilon(a_i \to \infty))$
    if $\varepsilon < \varepsilon_{\max}$
        remove element with the lowest $\varepsilon$
        calculate new $\mathbf{A}_N^{-1}$ and $\mathbf{A}_D^{-1}$
while $\varepsilon < \varepsilon_{\max}$
```

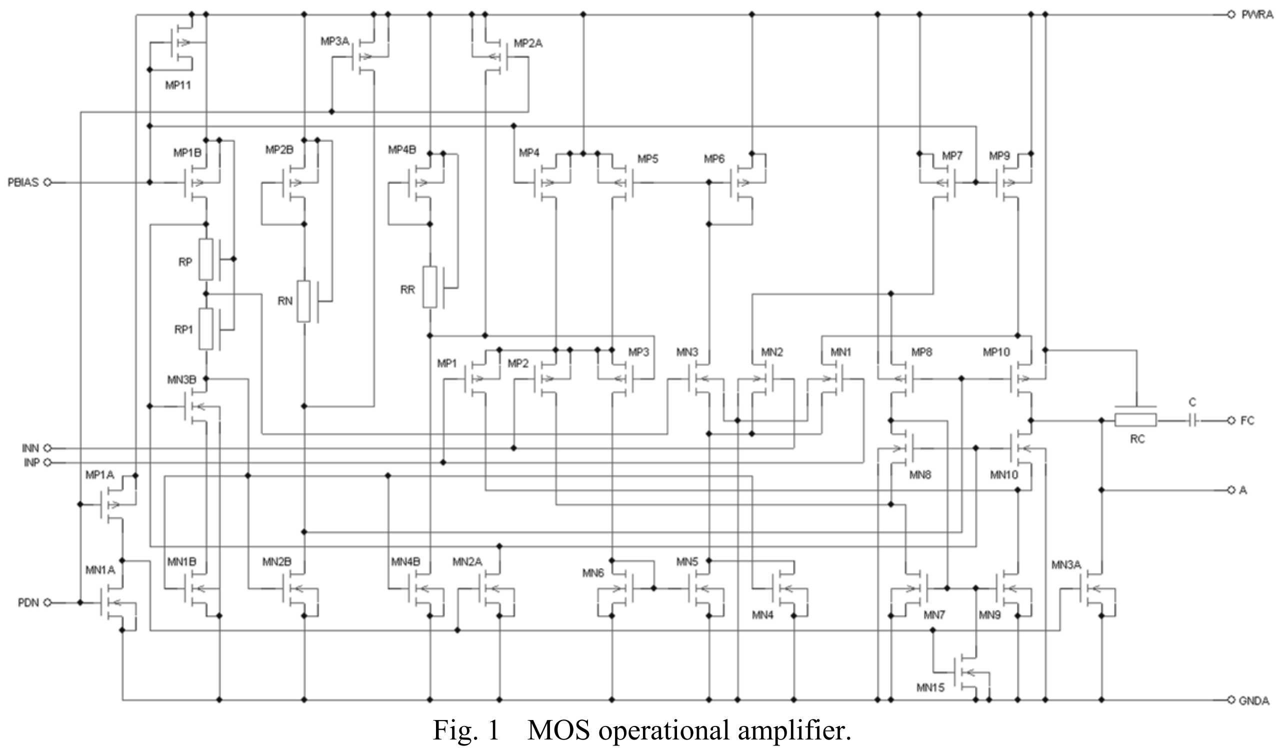
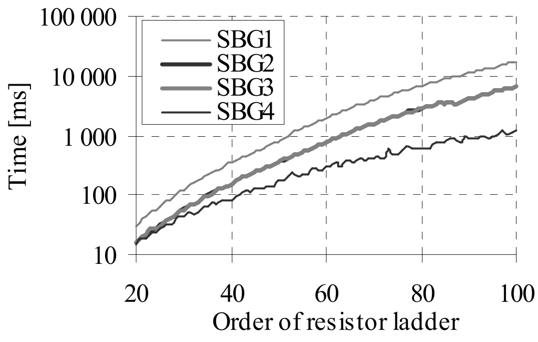


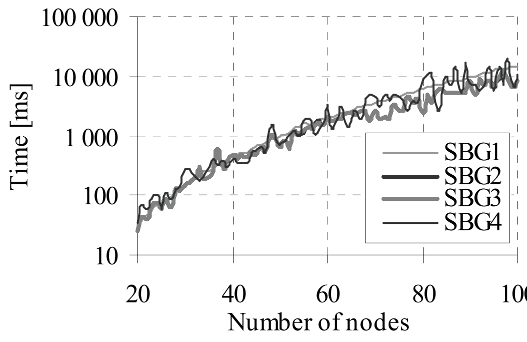
Fig. 1 MOS operational amplifier.

4 Analysis of benchmark circuits

Four benchmark circuits were used: (1) fully connected resistor network of order *n*; (2) *n*-th order resistor ladder; (3) bipolar operational amplifier 741 (741OA); (4) MOS operational amplifier (MOSOA), shown in Fig. 1.



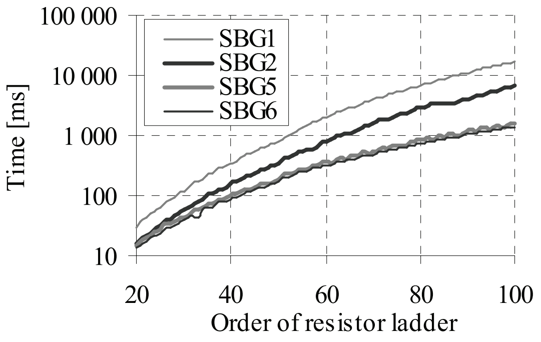
a)



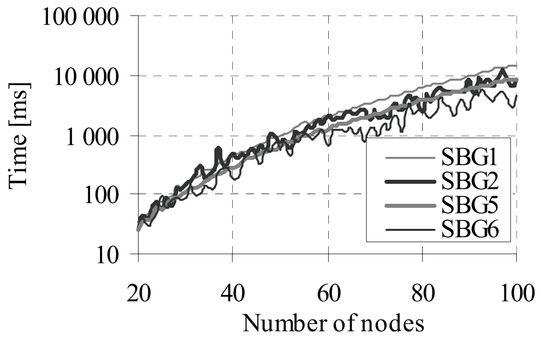
b)

Fig. 2 Results for SBG1-4: a) resistor ladder and b) fully connected resistor network.

Circuits 1 and 2 represent theoretical limits of complexity [5]. Small-signal model of real circuits 3 and 4 contains almost 100 nodes and 200 basic elements. In all cases, voltage transfer ratio between input and output was examined. The error criterion was set in such way, that all elements in the circuit were eliminated. This means, that the input and output ports became identical. The CPU time required for resistor ladder network and fully connected resistor network is shown in Fig. 2. This analysis was performed on Ahlon64 XP3500+ processor under Microsoft Windows XP.



a)



b)

Fig. 3 Results for SBG1,2,5,6: a) resistor ladder and b) fully connected resistor network.

It is obvious from Fig. 2a that the fastest algorithm is SBG4, which exploits the Sherman-Morrison formula in all cases. However, this program completely failed to simplify the ladder resistor network of order 74 and higher due to numerical instability. Algorithms SBG2 and SBG3 gave almost identical results, which was caused by a special form of the matrix.

Tab. 1: Time required for topological simplification of real circuits

Circuit	Analyzed frequency range	User-defined error		Ref. freq. count	Time required by the SBG method [ms]			
		ε [dB]	ψ [°]		SBG1	SBG2	SBG5	SBG6
MOSOA	0.1Hz – 1kHz	1	5	2	27 122	18 820	3 252	2 991
	0.1Hz – 1MHz	1	5	3	63 029	30 172	6 775	5 738
	0.1Hz – 2MHz	2	10	3	68 286	29 320	7 383	6 653
741OA	0.1Hz	2	5	1	20 533	8 885	2 311	2 152
	1kHz	1	5	1	19 069	8 498	2 165	2 001
	0.1Hz – 1kHz	2	10	2	38 526	17 237	4 438	4 074

All programs that use the Sherman-Morrison formula cause small oscillations in the required CPU time. It is caused by numerical errors in the computed inverse matrix which have significant impact to the elimination order of elements. Additionally, the program SBG4 gave correct result in 92% of runs and program SBG3 in 93%.

Fig. 3 compares algorithms SBG1 and SBG2 with and their sparse matrix versions. It is obvious that the fastest program is SBG6, but the program SBG5 is slower only by 10%. Both programs are almost two times faster than programs SBG1 a SBG2. Fig. 3b confirms conclusions from previous paragraph. In average, the fastest program was SBG6.

The only programs that provided correct results were applied to real circuits. In all cases, the voltage transfer network function was analyzed. Frequency range, design point count, user-defined error and required CPU time for topological simplification are shown in Tab.1.

Tab. 1 shows that the program SBG6 exploiting both the Sherman-Morrison theorem and sparse matrix technique for computing inverse matrix gives the best results. The program SBG5 exploiting sparse matrix technique only, is slower only about 10% than program SBG6. Other programs that use Gaussian elimination are almost five times slower.

5 Conclusions

In this contribution, the sparse matrix technique and the Sherman-Morrison formula for symbolic simplification have been compared. Experimental results show that the combination of the sparse matrix technique and the Sherman-Morrison formula gives the lowest time. A circuit containing almost 250 elements and 100 nodes can be topologically simplified by these techniques in several seconds using a standard personal computer.

6 Acknowledgements

This research has been supported by the Grant Agency of the Czech Republic under contracts No. 102/05/0771 and No. 102/05/0277 and by the Ministry of Education under contract No. MSM0021630513.

References:

[1] Lin, P., M., Symbolic network analysis. Elsevier, Amsterdam, 1991.

[2] Kolka, Z., Pospíšil V., Horák, M., Program for Symbolic Analysis. In: *Proc. of 11th International Conference on Mixed Design of Integrated Circuits and Systems (MIXDES 2004)*. Szczecin, Poland, 2004, p. 666 - 669.

[3] Householder, A., S., The Theory of Matrices in Numerical Analysis. New York, 1964

[4] Markowitz, H., M., The elimination form of the inverse and its application to linear programming. *Management Sci. 3*. p 255-269.

[5] Henning, E., Symbolic approximation and Modelling Techniques for Analysis and Design of Analog Circuits. Doctoral dissertation, Univ. of Kaiserslautern, Shaker Verlag, 2000.