A NEW METHOD FOR ORDERING SPARSE MATRICES AND ITS PERFORMANCE IN CIRCUIT SIMULATION

Gunther Reißig

Otto-von-Guericke-Universität Magdeburg
Chair of Systems Theory (J. Raisch), FEIT-IFAT
PF 4120
D-39016 Magdeburg, Germany

URL: http://www.reiszig.de/gunther/

ABSTRACT

Local algorithms for obtaining a pivot ordering for sparse symmetric coefficient matrices are reviewed together with their mathematical background and appropriate data structures. Recently proposed heuristics as well as improvements to them are discussed, and their performance, mainly in terms of the resulting number of factorization operations, is compared with that of the Minimum Degree and the Minimum Local Fill algorithms. It is demonstrated that a combination of Markowitz' algorithm with these symmetric methods applied to the unsymmetric matrices arising in circuit simulation is capable of accelerating the simulation significantly.

1. INTRODUCTION

Simulating an electrical circuit requires the solution of numerous linear equations of the form

$$Ax = b, \tag{1}$$

where A is a real or complex $n \times n$ matrix [Chua et al., 1987, Feldmann et al., 1992, Nagel, 1975]. Solving (1) is also necessary in other types of analyses, such as DC- and small signal analysis, and the efficiency by which this is done determines the quality of simulation tools as a whole to a great extent.

In many professional simulation tools, including TITAN [Feldmann et al., 1992], (1) is solved directly without pivoting for numerical accuracy [Hajj et al., 1981, Tan, 1986, Nagel, 1975], despite the fact that A is unsymmetric and indefinite. (To investigate the merits of that approach is beyond the scope of this paper.) In fact, one assumes that it is numerically feasible to solve

$$(PAP^T)y = Pb \tag{2}$$

by Gaussian elimination for any $n \times n$ permutation matrix P and then set $x = P^T y$. That is, the diagonal entries may be chosen as pivots, in any order [Hajj et al., 1981, Nagel, 1975].

Usually, A is very large and extremely sparse, i.e., only few of its entries are nonzero. However, new nonzeros are introduced during the elimination phase:

The coefficient matrix PAP^T is successively overwritten by the matrix $L + U - \mathrm{id}_n$, where

$$LU = PAP^T \tag{3}$$

and L and U are factors of PAP^T, L is unit lower triangular, U is upper triangular, and id_n is the $n \times n$ identity matrix [Chua and Lin, 1975, Golub and Van Loan, 1993, Nagel, 1975].

In general, the matrix $L + U - \mathrm{id}_n$ may have nonzeros at positions where the coefficient matrix PAP^T has zeros. Those newly created nonzeros are called *fill-ins*, and their total is called *fill* [Duff et al., 1986, George and Liu, 1989].

As the coefficient matrices of all linear equations to solve usually have the same zero-nonzero pattern, the same permutation matrix P may be used throughout the simulation [Nagel, 1975, Chua and Lin, 1975]. With the amount of fill created, the computational complexity of solving equation (1) heavily depends on P. Thus, a deliberate choice of P is extremely important for the overall performance of circuit simulation software.

Unfortunately, minimizing the fill appears to be harder than solving the linear equations at hand without taking advantage of their sparsity [Yannakakis, 1981]. Therefore, various heuristics have been proposed, which aim at acceptable low fill rather than minimum fill [Duff et al., 1986]. Among them, there are what is called *local* algorithms, which mimic Gaussian elimination based on the zero-nonzero pattern of the coefficient matrix alone, where it is assumed that nonzeros do not accidentally cancel out.

Those local methods always choose a pivot that minimizes some scoring function and thereby determine an overall pivoting order.

MARKOWITZ was the first to propose heuristics for fill-in minimization in Gaussian elimination [Markowitz, 1957]. In his *Minimum Local Fill (MF)* algorithm, the score of a diagonal entry is the number of fill-ins that would be created in the next elimination step if that en-

Proceedings 18th European Simulation Multiconference
Graham Horton (c) SCS Europe, 2004
ISBN 3-936150-35-4 (book) / ISBN 3-936150-36-2 (CD)

try was chosen as the next pivot. The following scoring function was also proposed in [Markowitz, 1957]:

Let c_i and r_i be the number of off-diagonal nonzeros in the ith column and row, respectively, of the zero-nonzero pattern that has been obtained from the preceding eliminations. The product $c_i r_i$ is called the *Markowitz product* and is an upper bound on the number of fill-ins introduced when the ith variable in the current scheme is eliminated next using the ith equation. To chose a pivot with minimum Markowitz product has become known as *Markowitz' algorithm*.

If A is structurally symmetric, i.e., $A_{i,j} \neq 0$ iff $A_{j,i} \neq 0$, Markowitz' algorithm is called *Minimum Degree (MD)* algorithm [George and Liu, 1989].

The computational cost of the MF algorithm may be two orders of magnitude higher than that of a version of the MD algorithm [Ng and Raghavan, 1999], which is a serious drawback. In addition, it has been found especially in circuit simulation that the MF algorithm saves at most 5% in factorization operations compared with other algorithms [Nagel, 1975].

Only recently, it has been found that the MF algorithm as well as newly proposed variants of both the MF and MD algorithms yield significantly better orderings than the MD algorithm on certain test suites of symmetric matrices [Cavers, 1989, Lustig et al., 1992, Mészáros, 1998, Rothberg and Eisenstat, 1998, Ng and Raghavan, 1999]. The running times of some of those algorithms are just in the order of those of the MD algorithm [Cavers, 1989, Mészáros, 1998, Rothberg and Eisenstat, 1998, Ng and Raghavan, 1999].

The savings in fill of those variants heavily depend on the field of application and increase with increasing problem size [Cavers, 1989], but, unfortunately, extensive tests and comparisons of those variants on matrices derived from circuits of a size representing today's simulation tasks are still missing.

A first step towards efficient algorithms for high-quality ordering of sparse, unsymmetric matrices in circuit simulation was presented in [Reißig and Klimpel, 2001] and tested in [Reißig, 2001]:

Initially, diagonal entries with Markowitz product zero are chosen as pivots, as many as possible. The corresponding elimination steps do not create any fill, hence the zero-nonzero pattern obtained after those steps is that of a submatrix $\widetilde{A}$ obtained from A by removing the pivot rows and columns. Now, a symmetric ordering method is applied to $|\widetilde{A}| + |\widetilde{A}|^T$, thereby completing the pivoting order for A. That way, much of the unsymmetry of A is removed by the initial steps of Markowitz' algorithm so that $\widetilde{A}$ will be symmetric or nearly so.

It was shown in [Reißig, 2001] that the orderings obtained from recently proposed ordering heuristics for symmetric matrices combined with the above method from [Reißig and Klimpel, 2001, Reißig, 2001] require up to 31% fewer factorization operations than those obtained from the Minimum Degree algorithm, in some cases at virtually no extra computational cost, when applied to symmetrized Jacobians of modified nodal equations.

The purpose of this paper is to demonstrate that, in terms of the resulting number of factorization operations, the method from [Reißig and Klimpel, 2001] outperforms the MD and Markowitz' algorithms even if applied directly to the unsymmetric matrices that arise in circuit simulation. In fact, our tests on a test suite of 15 matrices extracted from the circuit simulator TITAN of Infineon Technologies shows that the method we propose is capable of saving up to 72% (38% on average) of factorization operations if compared to Markowitz' algorithm.

The remaining of this paper is structured as follows. In section 2, we review PARTER's interpretation of Gaussian elimination in terms of graphs [Parter, 1961] and the data structure underlying local symmetric ordering methods. These concepts have already been the basis for the definition of recently proposed ordering heuristics in [Reißig, 2001]. For the convenience of the reader, we compile them in Section 3.

In Section 4, we compare the performance of the MD and MF algorithms and the algorithms from Section 3 on a test set of 15 unsymmetric Jacobians of modified nodal equations extracted from the circuit simulator TITAN [Feldmann et al., 1992] of Infineon Technologies.

2. DATA STRUCTURE OF LOCAL SYMMETRIC ORDERING METHODS

Throughout this section, A is a structurally symmetric $n \times n$ matrix, i.e., $A_{i,j} \neq 0$ iff $A_{j,i} \neq 0$, for all i and j, $1 \leq i, j \leq n$.

2.1. Gaussian elimination in terms of graphs

A *graph* is an ordered pair (V, E) of a finite set V of *nodes* and a set E of *branches*, $E \subseteq \big\{\{v, w\} \subseteq V \mid v \neq w\big\}$.

Two nodes $v, w \in V$ are *adjacent* in the graph G, $G = (V, E)$, if $\{v, w\} \in E$. For $w \in V$ and $W \subseteq V$, $\operatorname{adj}_G(w)$ and $\operatorname{adj}_G(W)$ denote the *adjacent set* of w and W, respectively, i.e.,

$$\operatorname{adj}_G(w) = \big\{u \in V \mid \{u, w\} \in E\big\},$$
$$\operatorname{adj}_G(W) = \bigcup_{u \in W} \operatorname{adj}_G(u) \setminus W.$$

For $X \in V \cup \mathcal{P}(V)$, we denote the *degree of* X *in* G by $\deg_G(X)$,

$$\deg_G(X) = |\operatorname{adj}_G(X)|,$$

where $|\cdot|$ denotes cardinality, and $\mathcal{P}(V)$ is the *power set* of V.

A node $v \in V$ and an branch $e \in E$ are *incident* if $v \in e$.

The *graph of* A, denoted $\mathcal{G}(A)$, is the graph (V, E) defined by

$$V = \{1, \ldots, n\},$$
$$E = \big\{\{i, j\} \mid A_{i,j} \neq 0, i \neq j\big\}.$$

The *elimination graph* G_v is obtained by *eliminating* $v \in V$ *from* $G = (V, E)$, i.e., by removing v and its incident branches from G, and connecting all nodes previously adjacent to v [Parter, 1961], thereby creating a set $\operatorname{fill}_G(v)$ of new branches or *fill-ins*.

Choosing pivots down the diagonal in PAP^T for some $n \times n$ permutation matrix P corresponds to selecting nodes of $\mathcal{G}(A)$ in the order $\pi(1), \pi(2), \ldots, \pi(n)$ for some bijection $\pi : V \to V$, which we call the *pivoting order*. Thus, local ordering algorithms are equivalent to the selection of nodes as follows:

```
Input:   Graph $G = (V, E)$, scoring function $s$.
Step 1:  $S := \emptyset$.
Step 2:  Pick $v \in V \setminus S$ with $s(v, G) = \min_{w \in V \setminus S} s(w, G)$.
Step 3:  $S := S \cup \{v\}$, $\pi(|S|) := v$, $G := G_v$.
Step 4:  If $S \neq V$, goto Step 2.
Output:  Pivoting order $\pi$.
```

In particular, if the input graph G equals $\mathcal{G}(A)$, the above algorithm is the MD and the MF algorithm for $s(v, G) = \deg_G(v)$ and $s(v, G) = |\operatorname{fill}_G(v)|$, respectively.

2.2. Clique representations of elimination graphs

A *clique* is a graph in which any two distinct nodes are adjacent. The node set of a clique will also be called a clique.

A set $\mathcal{C} \subseteq \mathcal{P}(V)$ is called a *clique representation* of the graph $G = (V, E)$ if $E = \big\{\{v, w\} \subseteq C \mid C \in \mathcal{C}, v \neq w\big\}$. In other words, clique representations of $\mathcal{G}(A)$ correspond to coverings of the nonzeros of A by full symmetric minors. Note also that E is a (trivial) clique representation of G.

If $\mathcal{C}$ is a clique representation of G and $v \in V$, then the set $\mathcal{C}'$,

$$\mathcal{C}' = \{C \in \mathcal{C} \mid v \notin C\} \cup \{\operatorname{adj}_G(v)\} \tag{4}$$

is a clique representation of G_v.

2.3. Indistinguishable nodes

The nodes $v, w \in V$ are *indistinguishable* [George and Liu, 1989], $v \underset{G}{\sim} w$, if

$$\operatorname{adj}_G(v) \cup \{v\} = \operatorname{adj}_G(w) \cup \{w\}. \tag{5}$$

If $v \underset{G}{\sim} w$, then $\deg_G(v) = \deg_G(w)$ and $\operatorname{fill}_G(v) = \operatorname{fill}_G(w)$. Moreover, for any reasonable scoring function, it should suffice to determine the score of one of v and w only. Hence, we may maintain the quotient graph $G/\underset{G}{\sim}$ rather than G itself, where $G/\underset{G}{\sim} = (V/\underset{G}{\sim}, E')$ and

$$E' = \big\{\{[v]_{\underset{G}{\sim}}, [w]_{\underset{G}{\sim}}\} \mid \{v, w\} \in E, [v]_{\underset{G}{\sim}} \neq [w]_{\underset{G}{\sim}}\big\}.$$

3. IMPROVED SCORING FUNCTIONS

Let G be the current elimination graph, $G = (V, E)$, $\sim$ be the equivalence relation of indistinguishable nodes, and let $\mathcal{C}$ be a clique representation of $G/\sim$. For simplicity, we denote the class $[v]_\sim$ by $[v]$.

For each node $[v]$ of $G/\sim$, let $\kappa_{[v]}$ be the list of cliques containing $[v]$. Let $c_{[v]}$ be the number of cliques in that list that have been created by eliminating nodes and assume that those created cliques are located at the beginning of the list.

We define scoring functions in terms of $G/\sim$. For $W \subseteq V/\sim$, we define

$$\|W\| = \sum_{[w] \in W} |[w]|.$$

3.1. Bounds on the local fill

The scoring function of the MD algorithm represents an upper bound on the number of fill-ins introduced by eliminating a node from G, since

$$\xi : x \mapsto (x^2 - x)/2$$

is monotonic on the set of nonnegative integers and $|\operatorname{fill}_G(v)| \leq \xi(\deg_G(v))$ for all $v \in V$.

That bound may be improved since some of the $\xi(\deg_G(v))$ potential fill-in branches in G_v are branches of G that are easy to identify:

First, if v is indistinguishable from w in G, $v \neq w$, then eliminating v from G does not create any new branches adjacent to w in G_v. Hence, the *external degree* $\deg_G([v])$ *of* v represents an upper bound on $|\operatorname{fill}_G(v)|$,

$$|\operatorname{fill}_G(v)| \leq \xi(\deg_G([v])). \tag{6}$$

Taking the external degree as scoring function usually produces less overall fill than taking the degree [George and Liu, 1989].

Further, if $C \in \mathcal{C}$ is a clique, then the elimination of $[v]$ from G cannot create any new branch $\{[u], [w]\} \subseteq C$. Even if only those cliques are considered that contain $[v]$, the bound in (6) may be improved in several ways:

The *Approximate Minimum Local Fill (AMF)* algorithm of ROTHBERG and EISENSTAT uses the upper bound s_{AMF_0},

$$s_{AMF_0}([v]) = \xi(\deg_G([v])) - \begin{cases} 0 & \text{if } c_{[v]} = 0 \\ \xi(\|\kappa_{[v]}(1) \setminus \{[v]\}\|) & \text{otherwise} \end{cases}$$

as its scoring function, thereby taking into account the most recently eliminated clique only [Rothberg and Eisenstat, 1998]. We will denote that scoring function that takes in account the largest, rather than the most recently created, clique, by s_{AMF_1}.

NG and RAGHAVAN propose to consider all cliques in $\kappa_{[v]}$ rather than just one [Ng and Raghavan, 1999].

3.2. Looking ahead

While local ordering algorithms usually consider the fill introduced in the next elimination step only, the concept of indistinguishability provides a simple means to look some steps ahead:
Since the total number of fill-ins created when all nodes in $[v]$ are eliminated from G immediately upon each other is just $|\operatorname{fill}_G(v)|$, ROTHBERG and EISENSTAT consider dividing fill bounds by $|[v]|$ [Rothberg and Eisenstat, 1998]. Their *(Approximate) Minimum Mean Local Fill ((A)MMF)* heuristics are based on the scoring functions s_{MMF^α} and $s_{AMMF_0^\alpha}$,

$$s_{MMF^\alpha}([v]) = |\operatorname{fill}_G(v)|/|[v]|^\alpha,$$
$$s_{AMMF_0^\alpha}([v]) = s_{AMF_0}([v])/|[v]|^\alpha$$

with $\alpha = 1$. According to [Rothberg and Eisenstat, 1998], a version with $\alpha = 1/2$ "produced slightly better results".

We denote by $s_{AMMF_1^\alpha}$ that scoring function that results from application of the above trick to s_{AMF_1}.

3.3. Further variants

Among the various improvements of the MD and MF algorithms that we do not discuss in this paper are the tie breaking techniques of [Cavers, 1989, Mészáros, 1998], the *Modified Multiple Minimum Degree (MMMD)* algorithm of [Ng and Raghavan, 1999], and the *correction terms* of [Rothberg and Eisenstat, 1998, Ng and Raghavan, 1999].

4. COMPUTATIONAL RESULTS

In addition to the techniques described in Section 2, our implementation of the MD and MF algorithms as well as their variants described in Section 3 includes further techniques, such as *element absorption*, *incomplete score update*, and *multiple elimination*, see [George and Liu, 1989].

The code was compiled with the cc compiler (Workshop Compilers 4.2 30 Oct 1996 C 4.2) using the options "-fast -fsimple=2 -xtarget=ultra -xarch=v8ultra" under SunOS Release 5.7 and run on one of the CPUs (sparcv9+vis, 400 MHz clock rate, 4 MB cache, 17.4 SPECint95, 25.7 SPECfp95) of a SUN Enterprise E4500 workstation with 6 Gbytes of memory.

We chose the implementation of Markowitz' algorithm of the circuit simulator TITAN [Feldmann et al., 1992], version $6.1a$, as our reference algorithm.

Our primary measure for comparing ordering algorithms is the number

$$\sum_{i=1}^{n} c_i(1 + r_i) \tag{7}$$

of factorization operations determined by the pivoting orders obtained, which represents divisions and multiplications, where c_i and r_i are the number of off-diagonal nonzeros in column and row i of the factors L and U, respectively, of PAP^T, and P is the permutation matrix corresponding to the pivoting order.

Our test suite of input data consists of 15 matrices extracted from the circuit simulator TITAN of Infineon Technologies. These matrices are listed under the names "bag" through "X" in Tab. 1.

It is well known that ordering heuristics are sensitive to permutations of the rows and columns of the input matrices. Therefore, for the combinations of Markowitz' algorithm with symmetric ordering methods to the problems "bag" through "sei" listed in Tab. 1, we report the arithmetic mean of the number of factorization operations over 11 runs as in the previous paragraph. For Markowitz' algorithm, and for the problems buc through X of Tab. 1, we report the result of one run only.

From the problem data presented in Tab. 1, we see that the initial steps of Markowitz' algorithm only slightly reduce the dimension of the problem, but remove $4\%-43\%$

Problem characteristics						Markowitz' A.	MMD	AMMF$_1^{1/2}$	MMF$^{1/2}$
Matrix	$\frac{n_0}{10^3}$	$\frac{m_0}{10^3}$	$\frac{m_1}{m_0}$ in %	$\frac{u_0}{m_0}$ in %	$\frac{u_1}{m_1}$ in %	o	o	o	o
bag	144	995	93	7.3	0	$3.5 \cdot 10^7$	0.95	0.81	0.68
cor	123	892	74	26	0	$1.8 \cdot 10^6$	0.87	0.87	0.87
eng	5	42	80	20	0	$1.1 \cdot 10^6$	0.96	0.97	0.88
jac	1	22	83	17	0	$1.0 \cdot 10^6$	1.12	1.28	0.92
m8	11	237	83	17	0	$3.9 \cdot 10^7$	0.77	0.56	0.43
m14	16	418	84	16	0	$7.1 \cdot 10^7$	0.79	0.64	0.55
m24	26	710	86	13	0	$4.1 \cdot 10^8$	0.94	0.72	0.49
m40	156	2039	88	12	0	$4.2 \cdot 10^9$	0.81	0.61	0.48
sei	4	292	96	3.7	0	$2.5 \cdot 10^8$	0.61	0.34	0.28
buc	10	78	58	51	17	$8.4 \cdot 10^5$	0.92	0.86	0.69
gue	89	549	69	32	2.3	$1.6 \cdot 10^7$	0.82	0.88	0.43
te	60	398	64	36	0.032	$7.6 \cdot 10^5$	1.00	1.01	0.99
tei	175	1206	67	33	0.0099	$4.7 \cdot 10^6$	1.08	1.04	0.86
xch	11	84	57	52	17	$8.3 \cdot 10^5$	1.02	0.82	0.73
X	16	170	67	40	11	$2.8 \cdot 10^6$	0.86	0.76	0.63
geometric mean							0.89	0.78	0.62

Table 1: Circuit matrices and performance of Markowitz' algorithm and its combination with the MMD, AMMF$_1^{1/2}$ and MMF$^{1/2}$ algorithms defined in section 3.2. n_0, m_0, and u_0 denote the number of rows, nonzeros, and structurally unsymmetric nonzeros, respectively. m_1 and u_1 denote the number of nonzeros and structurally unsymmetric nonzeros, respectively, in the matrices remaining after the initial steps of Markowitz' algorithm. o denotes the number of factorization operations (7), for the combinations divided by the corresponding value for Markowitz' algorithm. The values in the row "geometric mean" are geometric means of the ratios reported in the respective column. The values of n_0 and m_0 are rounded to multiples of 1000, all ratios are rounded to a precision of 10^{-2}, and the remaining quantities are rounded to two decimal digits.

of the nonzeros. Moreover, while $3.7\% - 52\%$ of the nonzeros of the original matrices are structurally unsymmetric, those initial steps remove most or all of them. In particular, the remaining matrices for the problems “bag” through “sei” are symmetric, and only about $0.01\% - 17\%$ of the nonzeros of the other remaining matrices are structurally unsymmetric.

For problems leading to symmetric remaining matrices, it is obvious that the advantages of the symmetric ordering methods investigated over the basic MD heuristic carry over to the combination of those methods with Markowitz’ algorithm. The results presented in Tab. 1 show that these advantages carry over to the combination even if the remaining matrices are structurally unsymmetric.

After all, the combination of Markowitz’ algorithm with the MMD, $\mathrm{AMMF}_1^{1/2}$, and $\mathrm{MMF}^{1/2}$ algorithms leads to 11%, 22%, and 38% fewer factorization operations than Markowitz’ algorithm alone, although for some of the problems from Tab. 1, MARKOWITZ’ algorithm yields better orderings than both MMD and $\mathrm{AMMF}_1^{1/2}$.

Furthermore, it is evident that the running time of the above combination with the MMD algorithm should never exceed that of an analogous but unsymmetric implementation of Markowitz’ algorithm. In fact, the code of Markowitz’ algorithm we used was always much slower than its combination with both the MMD and the $\mathrm{AMMF}_1^{1/2}$ heuristic.

In general, permuting coefficient matrices to block triangular form [Duff et al., 1986], of which removal of pivots with zero Markowitz product – the technique we applied – is a first step, followed by ordering and factoring the diagonal blocks can speed up the solution of linear equations even further. However, we found that the effect of that improvement is insignificant if applied to circuit matrices: For the circuit problems bag through sei from Tab. 1, removal of pivots with zero Markowitz product leads to irreducible matrices in all cases except m40. For those problems from Tab. 1 that lead to reducible matrices, the dimension and the number of nonzeros, respectively, of the largest diagonal block would always be greater or equal to 97.6% and 98.7%, respectively, of the corresponding numbers for the whole matrix. Furthermore, the number of structurally unsymmetric nonzeros would be reduced in two cases only and by approximately 0.1%.

A final decision on which of those heuristics is best would not only depend on the kind of circuits to be simulated, but also on the computer architecture and the specific numerical factorization algorithm used [Lustig et al., 1992] and is beyond the scope of this paper. However, let us show by an example that the savings in factorization operations of the $\mathrm{MMF}^{1/2}$ over the $\mathrm{AMMF}_1^{1/2}$ heuristic may very well reduce the overall simulation time:

The largest CPU time for the $\mathrm{MMF}^{1/2}$ algorithm was 2h43min, achieved for the “m40” example for which application of the $\mathrm{AMMF}_1^{1/2}$ algorithm took only 47sec.. However, a transient simulation of that example with TITAN [Feldmann et al., 1992], version $6.1a$, using the pivot ordering obtained from Markowitz’ algorithm, spent over 138h on factorizations, so that the savings of the $\mathrm{MMF}^{1/2}$ algorithm in factorization operations of 21% relative to the $\mathrm{AMMF}_1^{1/2}$ heuristic should outweigh the larger CPU time of the former.

5. CONCLUSIONS

We have reviewed recently proposed local symmetric ordering methods, i.e., methods for obtaining pivot orderings for sparse symmetric matrices, as well as some straightforward improvements to them. We have shown that for the purpose of circuit simulation, a combination of Markowitz’ algorithm with symmetric ordering methods yields pivot orderings significantly better than those obtained from Markowitz’ algorithm alone, in some cases at virtually no extra computational cost and that that combination is capable of accelerating circuit simulation significantly.

We could not make a final decision on what ordering algorithm is best. That decision would not only depend on the kind of circuits to be simulated, but also on the computer architecture and the specific numerical factorization algorithm used and is beyond the scope of this paper.

However, we think that a further improvement of the running times of those symmetric methods that are based on exact local fill counts, for example, through a combination of multiple elimination with the fill updating method of WING and HUANG [Wing and Huang, 1975, Vlach and Singhal, 1983], would make them superior to any other local ordering method known today when combined with Markowitz’ algorithm as described in [Reißig and Klimpel, 2001] and applied in this paper.

6. ACKNOWLEDGMENT

I thank P. I. Barton (Cambridge, MA), G. Denk (München), and Uwe Feldmann (München), for their valuable hints and comments.

7. REFERENCES

[Cavers, 1989] Cavers, I. A. (1989). Using deficiency measure for tiebreaking the minimum degree algorithm. Technical Report 89-2, Dept. Comp. Sci., The Univ. Of British Columbia, Vancouver, B. C., Canada V6T 1W5.

[Chua et al., 1987] Chua, L. O., Desoer, C. A., and Kuh, E. S. (1987). *Linear and Nonlinear Circuits*. McGraw–Hill.

[Chua and Lin, 1975] Chua, L. O. and Lin, P.-M. (1975). *Computer–Aided Analysis of Electronic Circuits*. Prentice–Hall, Englewood Cliffs, NJ.

[Duff et al., 1986] Duff, I. S., Erisman, A. M., and Reid, J. K. (1986). *Direct methods for sparse matrices*. Oxford University Press.

[Feldmann et al., 1992] Feldmann, U., Wever, U. A., Zheng, Q., Schultz, R., and Wriedt, H. (1992). Algorithms for modern circuit simulation. *Archiv für Elektronik und Übertragungstechnik (AEÜ)*, 46(4):274–285.

[George and Liu, 1989] George, A. and Liu, J. W. (1989). The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31(1):1–19.

[Golub and Van Loan, 1993] Golub, G. H. and Van Loan, C. F. (1993). *Matrix Computations*. The John Hopkins Univ. Press, 2 edition.

[Hajj et al., 1981] Hajj, I. N., Yang, P., and Trick, T. N. (1981). Avoiding zero pivots in the modified nodal approach. *IEEE Transactions on Circuits and Systems*, 28(4):271–278.

[Lustig et al., 1992] Lustig, I. J., Marsten, R. E., and Shanno, D. F. (1992). The interaction of algorithms and architectures for interior point methods. In Pardalos, P. M., editor, *Advances in optimization and parallel computing*, pages 190–204. North-Holland.

[Markowitz, 1957] Markowitz, H. M. (1957). The elimination form of the inverse and its application to linear programming. *Management Science*, 3:255–269.

[Mészáros, 1998] Mészáros, C. (1998). Ordering heuristics in interior point LP methods. In Giannessi, F., Komlósi, S., and Rapcśak, T., editors, *New trends in mathematical programming*, pages 203–221. Kluwer Acad. Publ.

[Nagel, 1975] Nagel, L. W. (1975). SPICE 2: A computer program to simulate semiconductor circuits. Technical Report ERL-M520, Univ. of Calif. Berkeley, Electronic Res. Lab., Berkeley, CA.

[Ng and Raghavan, 1999] Ng, E. G. and Raghavan, P. (1999). Performance of greedy ordering heuristics for sparse Cholesky factorization. *SIAM Journal on Matrix Analysis and Applications*, 20(4):902–914.

[Parter, 1961] Parter, S. V. (1961). The use of linear graphs in Gauss elimination. *SIAM Review*, 3:119–130.

[Reißig, 2001] Reißig, G. (2001). On methods for ordering sparse matrices in circuit simulation. In *Proc. 2001 IEEE Int. Symp. on Circuits and Systems (ISCAS), Sydney, Australia, May 6-9*, volume 5, pages 315–318.

[Reißig and Klimpel, 2001] Reißig, G. and Klimpel, T. (2001). Verfahren zum computergestützten Vorhersagen des Verhaltens eines durch Differentialgleichungen beschreibbaren Systems. Patent DE 101 03 793 A 1. (“Method for computer-aided prediction of the behavior of a system described by differential equations”, in German).

[Rothberg and Eisenstat, 1998] Rothberg, E. and Eisenstat, S. C. (1998). Node selection strategies for bottom-up sparse matrix ordering. *SIAM Journal on Matrix Analysis and Applications*, 19(3):682–695.

[Tan, 1986] Tan, G.-L. (1986). An algorithms for avoiding zero pivots in the modified nodal approach. *IEEE Transactions on Circuits and Systems*, 33(4):431–434.

[Vlach and Singhal, 1983] Vlach, J. and Singhal, K. (1983). *Computer methods for circuit analysis and design*. Van Nostrand Rheinhold.

[Wing and Huang, 1975] Wing, O. and Huang, J. (1975). SCAP - a sparse matrix circuit analysis program. In *Proc. 1975 IEEE Int. Symp. on Circuits and Systems (ISCAS)*, pages 213–215.

[Yannakakis, 1981] Yannakakis, M. (1981). Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 2(1):77–79.

8. AUTHOR BIOGRAPHIES

GUNTHER REIßIG received his Dipl.-Ing. (EE), Dipl.-Math. and Dr.-Ing. (EE) degrees from Technische Universität Dresden, Dresden, Germany, in 1991, 1995, and 1998, respectively. He was awarded the “Förderpreis” of the Information Technology Society, Germany, for his Dr.-Ing. thesis. From 1998 to 2000, he was with SIEMENS AG (München, Germany) and Infineon Technologies (München, Germany), where he contributed to the improvement of the diagnosis, VHDL, and sparse matrix capabilities of the circuit simulator TITAN. From 2000 to 2002, he was with the Department of Chemical Engineering of the Massachusetts Institute of Technology as a Postdoctoral Research Fellow. He is now with the chair of systems theory at the Department of Electrical Engineering of the Otto-von-Guericke Universität Magdeburg (Magdeburg, Germany) as an assistant. At present, he is particularly interested in the structural approach to systems analysis as well as discrete-continuous systems.