Available online at www.sciencedirect.com
ScienceDirect
Mathematics and Computers in Simulation 80 (2010) 1684–1693
MATHEMATICS AND COMPUTERS IN SIMULATION
www.elsevier.com/locate/matcom

Fill-ins number reducing direct solver designed for FIT-type matrix

Michał Dobrzyński *, Jagoda Plata

Numerical Methods Laboratory (LMN), Politehnica University of Bucharest, Splaiul Independentei 313, 060042 Bucharest, Romania

Received 30 October 2008; received in revised form 8 April 2009; accepted 2 May 2009
Available online 15 May 2009

Abstract

This paper investigates the particular problem of matrices appearing during the modeling of Integrated Circuits with Finite Integration Technique (FIT) method. We present the key points of FIT approach followed by an illustration of the structure and the properties of the FIT-type matrix. A novel algorithm SMark is proposed, which focuses on fill-ins number reduction. The main idea of SMark is the concept of a dual architecture—symbolic and numeric factorization.

In order to validate SMark a comparison with other methods was performed. The excellent results confirm that the proposed approach is an adequate solving method for FIT-type matrices, whereas the identified weak points of the algorithm indicate possible directions in the future work.

Keywords: Fill-ins; Direct solver; LU decomposition; Finite Integration Technique

1. Introduction

Direct solvers for matrices resulting from modeling of Integrated Circuits cause high load of memory, extended computation time and a large number of numerical faults. These three main problems can be diminished by improving the patterns of triangular matrices in LU decomposition. The choice of a proper preordering algorithm can be a basic method to reach the satisfying effects.

Nowadays the need of automatic tools with ability to deal with complex Integrated Circuit (IC) modeling has become a must. As a result, many laboratories and industrial teams all over the world have aimed at accurate models in wide range of frequencies (Itrs roadmap, web-page, cited May 26, 2008. http://www.itrs.net). This challenge leads to the major problem of computational efficiency and memory storage. As the machine computational limits have been reached, an essential improvement of solving and modeling algorithms must take place [12].

The research presented in this paper is based on the Chameleon-RF (CHRF) project [8]. The paper presents the bottleneck of the Finite Integration Technique (FIT) and the possible solutions for its elimination. In the FIT method the accuracy increment of the modeled device is obtained by grid condensation. This approach results in larger output matrices [11], which are difficult to be solved. In this paper a new direct solver is proposed.

* Corresponding author.
E-mail addresses: d_michal@lmn.pub.ro (M. Dobrzyński), plata@lmn.pub.ro (J. Plata).

0378-4754/$36.00
doi:10.1016/j.matcom.2009.05.001

Although the process of sparse symmetric matrix decomposition is fairly well understood, the unsymmetric case is still a challenge. Since first proposed in 1957 by Harry M. Markowitz, the Minimum Degree Algorithm (MD) [10], with its further Multiple Minimum Degree (MMD) [9] and Approximate Minimum Degree (AMD) [1] modifications, has become the most popular and robust solution for unsymmetric matrices. The solving packages as Unsymmetric Multifrontal Package (UMFPACK) [4,5] developed by Davis and Duff, Duff and Amestoy's Symmetric Pattern Multifrontal Package (MUPS) [2] or parts of Schenk and Gartner's Parallel Sparse Direct Solver Package (PARDISO) [13] are all based on MD. Although it is hard to beat UMFPACK or PARDISO in general, the particular case improvements are still feasible. In this paper a dedicated solving method is proposed, taking advantage of the special properties of FIT-matrices as: huge sparsity, unsymmetry, and wide numerical value variability [7,12].

This paper is structured as follows: Section 2 is dedicated to the explanation of the IC modeling process using FIT method. The main purpose of the section is to illustrate the structure of an exemplary FIT-type matrix and to introduce the main problem: Linear System of Equations (LSE) needed to be solved. In Section 3 a brief definition of the Gaussian Elimination is given in the context of solving LSE. The research results of Section 4 are the main contribution of the paper. The proposed preordering algorithm SMark is introduced. In order to indicate the strong and the weak sides of the SMark algorithm, Section 5 carries information about UMFPACK package, and the comparison between the two methods. Numerical results are presented based on an exemplary FIT-type matrix. Finally, Section 6 concludes this paper and presents our plans on further work.

2. Problem

This section describes the problem formulation from the physical and mathematical point of view. The purpose of the section is to illustrate the structure and properties of the matrices that will be submitted to the solver.

The process of IC modeling consists of two stages: continuous model description and continuous model discretization. Extraction of the continuous model is based on Maxwell equations for the electromagnetic elements [7] and it is described by Partial Differential Equations (PDE). In order to discretize the continuous model the FIT method was used [15]. The basic concept of the classical FIT method, is the use of Differential Algebraic Equations (DAE) and its spatial discretization into the Maxwell Grid Equations (MGE) (1)–(5). This approach does not carry a discretization error because of its non-metric dimension. The boundary conditions are taken into consideration with (6)–(8). The material properties are modeled by Hodge Eqs. (9)–(11).

$$\operatorname{curl}\mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \quad \Rightarrow \quad \int_{\Gamma} \mathbf{E}\cdot \mathrm{d}\mathbf{r} = -\int\int_{S_\Gamma} \frac{\partial \mathbf{B}}{\partial t}\cdot \mathrm{d}\mathbf{A} \quad \Rightarrow \mathbf{C}\mathbf{u}_{\mathrm{e}} = -\frac{\mathrm{d}\varphi}{\mathrm{d}t} \tag{1}$$

$$\hookrightarrow \operatorname{div}\mathbf{B} = 0 \quad \Rightarrow \quad \int\int_{\Sigma} \mathbf{B}\cdot \mathrm{d}\mathbf{A} = 0 \quad \Rightarrow \quad \mathbf{D}'\varphi = \mathbf{0} \tag{2}$$

$$\operatorname{curl}\mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} \Rightarrow \int_{\Gamma} \mathbf{H}\cdot \mathrm{d}\mathbf{r} = \int\int_{S_\Gamma} \left(\mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}\right)\cdot \mathrm{d}\mathbf{A} \Rightarrow \mathbf{C}'\mathbf{u}_m = \mathbf{i} + \frac{\mathrm{d}\psi}{\mathrm{d}t} \tag{3}$$

$$\hookrightarrow \operatorname{div}\mathbf{D} = \rho \quad \Rightarrow \quad \int\int_{\Sigma} \mathbf{D}\cdot \mathrm{d}\mathbf{A} = \int\int\int_{\mathcal{D}_\Sigma} \rho\, \mathrm{d}v \quad \Rightarrow \mathbf{D}\psi = \mathbf{q} \tag{4}$$

$$\hookrightarrow \operatorname{div}\mathbf{J} = -\frac{\partial \rho}{\partial t} \quad \Rightarrow \quad \int\int_{\Sigma} \mathbf{J}\cdot \mathrm{d}\mathbf{A} = -\int\int\int_{\mathcal{D}_\Sigma} \frac{\partial \rho}{\partial t}\,\mathrm{d}v \quad \Rightarrow \mathbf{D}\mathbf{i} = -\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} \tag{5}$$

$$\mathbf{n}\cdot \operatorname{curl}\mathbf{E}(P,t) = 0, \forall P \in \sum \tag{6}$$

$$\mathbf{n}\cdot \operatorname{curl}\mathbf{H}(P,t) = 0, \forall P \in \sum -\overset{'}{S}_k \tag{7}$$

$$\mathbf{n}\times \mathbf{E}(P,t) = 0, \forall P \in \bigcup \overset{'}{S}_k \tag{8}$$

$$\mathbf{B} = \mu\mathbf{H} \quad \Rightarrow \quad \varphi = \mathbf{G}_{\mathrm{m}}^{-1}\mathbf{u} \tag{9}$$

$$\mathbf{D} = \varepsilon \mathbf{E} \quad \Rightarrow \quad \psi = \mathbf{C}_\mathrm{e}\mathbf{v} \tag{10}$$

$$\mathbf{J} = \sigma \mathbf{E} \quad \Rightarrow \quad \mathbf{i} = \mathbf{G}_\mathrm{e}\mathbf{v} \tag{11}$$

A Linear Time Invariant (LTI) system is obtained from the matrix form of the MGE and Hodge equations [6]. The problem of large matrix decomposition appears in order to solve the LTI system. Moreover, the LTI system has to be solved for each frequency separately (13)–(15), to provide the wide frequency range analysis of the device modeled.

Let $\mathbf{C}$, $\mathbf{G}$, $\mathbf{B}$, $\mathbf{L}$ be semi-state space matrices.

$$\begin{cases} \mathbf{C}\dot{\mathrm{x}} = -\mathbf{G}\cdot\mathbf{x} + \mathbf{B}\cdot\mathbf{u} \\ \mathbf{y} = \mathbf{L}\cdot\mathbf{x} \end{cases}, \quad \mathbf{u} = \begin{bmatrix} \mathbf{v} & \mathbf{i} \end{bmatrix}^T, \quad \mathbf{x} = \begin{bmatrix} \mathbf{v}_\mathrm{e} & \mathbf{v}_\mathrm{m} \end{bmatrix}^T \tag{12}$$

where $\mathbf{v}$ is the electric voltages, $\mathbf{i}$ the electric current, $\mathbf{v}_\mathrm{e}$ the electric voltages defined on the electric grid and $\mathbf{v}_\mathrm{m}$ is the magnetic voltages defined on the electric grid.

$$\mathbf{A} = j\omega\mathbf{C} + \mathbf{G} \tag{13}$$

$$\mathbf{b} = \mathbf{B}\cdot\mathbf{u} \tag{14}$$

$$\mathbf{y} = \mathbf{L}\cdot\mathbf{A}\setminus\mathbf{b} \tag{15}$$

Assuming that all inputs are current excited, the state-transition matrix $\mathbf{A}$ – which from now on will be called a FIT-type matrix – has a following structure [3]:

$$\mathbf{A} = \begin{bmatrix} j\omega\mathbf{C}_\mathrm{e} + \mathbf{G}_\mathrm{e} & -\mathbf{T}' \\ \mathbf{T} & j\omega\mathbf{G}_\mathrm{m} \\ j\omega\mathbf{C}_\mathrm{e\text{-}Sl} + \mathbf{G}_\mathrm{e\text{-}Sl} & 0 \\ j\omega\mathbf{C}_\mathrm{e\text{-}T} + \mathbf{G}_\mathrm{e\text{-}T} & 0 \end{bmatrix} \tag{16}$$

where $\mathbf{C}_\mathrm{e}$ is the electric capacitance matrix (edges of the electric grid), $\mathbf{G}_\mathrm{e}$ the electric conductance matrix (edges of the electric grid), $\mathbf{C}_\mathrm{e\text{-}Sl}$ the electric capacitance matrix (edges connected to boundary nodes), $\mathbf{G}_\mathrm{e\text{-}Sl}$ the electric conductance matrix (edges connected to boundary nodes), $\mathbf{G}_\mathrm{m}$ the inverted magnetic reluctance matrix (edges of the magnetic grid), $\mathbf{C}_\mathrm{e\text{-}T}$ the electric capacitance matrix (edges touching terminals), $\mathbf{G}_\mathrm{e\text{-}T}$ the electric conductance matrix (edges touching terminals), and $\mathbf{T}$ is the topological matrix (electric grid; matrix contains only $+1$, 0, -1 elements)

The matrix $\mathbf{A}$ in Eq. (16) has the following special properties: (a) square; (b) sparse (the sparsity rises hyperbolically in the function of matrix dimension); (c) unsymmetric; (d) unstructured; (e) complex.

To solve the electromagnetic simulation problem, the structure of the device has to be spatially discretized by a proper grid division of the computational domain. The geometrical model's mesh is based on a dual, orthogonal grid, one for electric and one for magnetic characteristics of the electromagnetic filed. By grid refinement a better material behavior and a more accurate model's geometry description can be obtained. However, this approach very easily tempts to extend up to computational limits. For this reason, in most cases, another issue becomes important: not only the grid extension, but also the solving method adjustment must be considered.

Assuming the device characteristic is frequency dependent (13), LSE (15) has to be solved for each frequency independently. Nevertheless, the pattern of the matrix (16) remains the same. The above-mentioned facts can be a primary clue to savings in the computational effort.

The idea of this contribution is to develop the proper reordering algorithm, so that the main solving effort should apply only once (i.e. for the first matrix), and the obtained result could be extended for all the next frequencies (matrices) of the same device.

Assuming (13)–(15) we can define the LSE that has to be solved for each frequency ω:

$$\mathbf{A}(\omega)\cdot\mathbf{x} = \mathbf{b}. \tag{17}$$

3. Methodology

Gaussian Elimination (GE) [14] is one of the simplest, best known operations for solving LSE. During the process of GE (18), applied to (17), matrices **L** and **U** are created so that Eq. (19) applies.

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} a_{kj}^{(k)} \tag{18}$$

where i is the row number, j the column number, and k is the step number.

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{U} \tag{19}$$

where **L** is the lower-triangular matrix and **U** is the upper-triangular matrix.

The factorization (19) is known as LU decomposition. As a result, the LSE can be easily solved only by forward-substitution and back-substitution operations: $\mathbf{z} = \mathbf{L} \setminus \mathbf{b}$, $\mathbf{x} = \mathbf{U} \setminus \mathbf{z}$.

This approach reduces the number of operations to a level proportional to the number of arithmetic operations on the non-zero entries of matrices **L** and **U**. The conclusion is, that it is essential to reduce the number of fill-ins[1] for a very large LSE in order to keep not only the memory storage, but also the computation time reasonably low.

Unfortunately, the GE in its pure form is, in general, unstable. Numerical reordering of the matrix, in particular partial (20) or full (21) pivoting, is necessary to provide stability and reduction of the fill-ins number.

$$\mathbf{L} \cdot \mathbf{U} = \mathbf{P} \cdot \mathbf{A} \quad \Rightarrow \quad \mathbf{z} = \mathbf{L} \setminus (\mathbf{P} \cdot \mathbf{b}),\ \mathbf{x} = \mathbf{U} \setminus \mathbf{z} \tag{20}$$

$$\mathbf{L} \cdot \mathbf{U} = \mathbf{P} \cdot \mathbf{A} \cdot \mathbf{Q} \quad \Rightarrow \quad \mathbf{z} = \mathbf{L} \setminus (\mathbf{P} \cdot \mathbf{b}),\ \mathbf{x} = \mathbf{Q} \cdot (\mathbf{U} \setminus \mathbf{z}) \tag{21}$$

where **P** is the row permutation matrix and **Q** is the column permutation matrix.

In order to choose the best reordering strategy adapted to the FIT-matrix, some established methods were compared. In addition, a new method was proposed. It is based on the concept of hierarchical architecture of dual – symbolic and numeric – decomposition stages in every LU factorization loop.

4. Proposed algorithm: SMark

The matrix $\mathbf{A}(\omega)$ (16) is frequency dependent only and therefore the matrices $\mathbf{A}(\omega_1), \ldots, \mathbf{A}(\omega_{\max})$ have different numerical values, but the same size and the same pattern. The main idea of Simple Markowitz algorithm (SMark) is to find a row permutation matrix **P** and a column permutation matrix **Q** as preordering matrices (21) based on the coefficient matrix **A** for the first simulated frequency ω_1. This preordering should be a sufficient tool to reduce the number of fill-ins and guarantee stability in the LU decomposition process.

In this section attention should be paid to the following aspects: algorithm structure; Markowitz Cost Function implementation; criterion of numerical value selection; possibility of taking fill-in as a pivot; how the results extend to other frequencies.

To manage both – fill-ins number reduction and stability problems – the SMark algorithm was divided into two stages: symbolic and numeric. This means that the first stage is processed only symbolically, i.e. without the numerical values of decomposed matrix **A**. On the contrary, the numeric stage takes into consideration and operates on the actual values of the entries of **A**. We chose the serial architecture with symbolic level before the numeric one (Fig. 1 a), in order to put stress on fill-ins number reduction. The SMark algorithm is structured as follows. In every step of LU decomposition we find the pivot $a_{\max}$ whose row and column indices are stored in the matrices **P** and **Q**. The pivot is searched for among all the non-zero elements of the matrix **A**.

[1] The fill-ins of a matrix are those entries which change from an initial zero to a non-zero value during the execution of an algorithm.

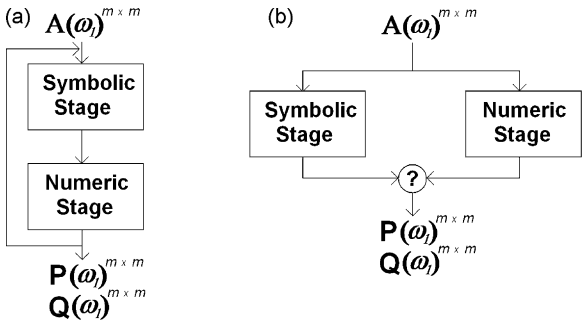

Fig. 1. Algorithm structures: (a) serial symbolic-numeric and (b) parallel symbolic-numeric.

4.1. Symbolic stage

First, for every non-zero element $m = 1, \ldots, n$, where n is a total number of non-zero elements of matrix $\mathbf{A}$, a Markowitz Cost value (MCv) is found. The MCv, denoted v_m, is a result of Markowitz Cost Function (MCF) (22) calculated for each non-zero entry of $\mathbf{A}$ [10]. In other words, a complete set of MCv for $m = 1, \ldots, n$ forms a corresponding matrix $\mathbf{A}_V$ with the similar pattern as $\mathbf{A}$, but with v_m values instead of the actual entries of $\mathbf{A}$.

The MCv is computed as follows. For every non-zero element of $\mathbf{A}$ and in every step m, the product v_m of two components is calculated: $(r_i - 1)$ and $(c_j - 1)$. Here, r_i is the number of non-zero elements in the i-th row (which the element m belongs to), c_j the number of non-zero elements in the j-th column (which the element m belongs to). After the process is complete, the results $v_1, v_2, \ldots, v_n$ are stored in vector v (23).

Next, a potential pivot for the matrix $\mathbf{A}$ is found. Its location is determined by the smallest value of MCv throughout the vector $\mathbf{v}$. This means, that the pivot is chosen only from the non-zero elements of $\mathbf{A}$ and its position is such, that it is placed both in the least populated row and in the least populated column.

The selection of the pivot always points at the most sparse row and the sparsest column; however the pivot must be a non-zero entry of the matrix. This is done simply by finding the smallest value $\mathbf{v}_{\min}$ of vector $\mathbf{v}$ (24). At this stage, please note that $\mathbf{v}_{\min}$ can contain more than one element. It results from the fact, that several entries of matrix $\mathbf{A}$ can be placed in the "equally row- and column- populated" position. If so, the "breaking the ties" technique is used. The approach is as follows: for all the elements of vector $\mathbf{v}_{\min}$ only these one are selected which indicate the position of the pivot in the least populated columns (25).

The sparsity of particular column is the data previously computed as c_j and properly stored for this operation. The selected pivot candidates are stored in the vector $\mathbf{v}_{\text{cmin}}$. At this stage, please observe that also $\mathbf{v}_{\text{cmin}}$ can contain more than one element. If so, this would be the case that several entries of matrix $\mathbf{A}$ (candidates to become a pivot) are placed in the "equally populated" columns.

To conclude, as a result of the symbolic stage we obtain vector $\mathbf{v}_{\text{cmin}}$ which contains coordinates of the best pivot candidates with respect to the sparsity. If the size of $\mathbf{v}_{\text{cmin}}$ equals 1, it becomes obvious that its single element becomes a pivot. However, if $\mathbf{v}_{\text{cmin}}$ contains more pivot candidates, the next -numeric- stage of the algorithm has to take place.

For the thorough explanation of the symbolic stage please follow Example 1 below.

$$\forall m \in \mathbf{A};\ v_m = (r_i - 1) \cdot (c_j - 1) \tag{22}$$

$$\mathbf{v} = [v_1, \ldots, v_n] \tag{23}$$

$$\mathbf{v}_{\min} = min(\mathbf{v}) \tag{24}$$

$$\mathbf{v}_{\text{cmin}} = column\_min(\mathbf{v}_{\min}) \tag{25}$$

Example 1. Lets consider an exemplary matrix $A = \begin{bmatrix} 7 & 0 & 0 & 0 \\ 3 & 0 & -5 & 4 \\ 1 & 2 & 0 & 0 \\ -8 & 0 & -9 & 0 \end{bmatrix}$;

then number of non-zero elements $n = 8$;

the number of non-zero elements in rows $= \begin{cases} r_1 = 1 \\ r_2 = 3 \\ r_3 = 2 \\ r_4 = 2 \end{cases}$;

and the number of non-zero elements in columns $= \begin{cases} c_1 = 4 \\ c_2 = 1 \\ c_3 = 2 \\ c_4 = 1 \end{cases}$;

so, the new matrix A_V with MCv (22) instead of the original values can be created. $A_V = \begin{bmatrix} \mathbf{0} & 0 & 0 & 0 \\ \mathbf{6} & 0 & \mathbf{2} & \mathbf{0} \\ \mathbf{3} & \mathbf{0} & 0 & 0 \\ \mathbf{3} & 0 & \mathbf{1} & 0 \end{bmatrix}$;

the values are stored in vector (23)$v = [0\,6\,3\,3\,0\,2\,1\,0]$;.

The smallest elements of vector v (24) are $v_{min} = v(1) = 0 \cup v(5) = 0 \cup v(8) = 0$; what corresponds to the elements of matrix A_V: $A_V(1, 1)$, $A_V(3, 2)$, $A_V(2, 4)$.

Vector v contains more than 1 element, so "breaking the ties" technique was used:

for all elements of vector v the ones with the smallest number of elements in columns are selected (25)$v_{cmin} = v(5) \cup v(8)$.

This happens because of the fact, that the number of elements in column $c_1 = 4$, $c_2 = \mathbf{1}$, $c_4 = \mathbf{1}$.

At this stage the elements of matrix A corresponding to vector v_{cmin} are the potential pivots.

$$v(5) \rightarrow A(3, 2), \quad v(8) \rightarrow A(2, 4).$$

4.2. Numeric stage

The objective of the numeric stage is to make a final choice of the pivot. It may occur, that the symbolic stage has returned the unique and single element—so it becomes a pivot in this "lucky" case. However, in practice, in many cases the algorithm necessitates the second stage of selection to be proceeded. This is the case when vector $\mathbf{v}_{\text{cmin}}$ contains more than one entry and therefore many pivot candidates.

First, for all m entries of vector $\mathbf{v}_{\text{cmin}}$ the related numerical values of the original matrix $\mathbf{A}$ are assigned (26). Again, this operation is done operating only on the non-zero elements of $\mathbf{A}$. In contrast to the symbolic stage, which based only on the sparsity pattern of $\mathbf{A}$ (regardless the numerical values of its entries), the numeric stage assigns to the selected pivot candidates their actual numerical values. The numerical values of the pivot candidates are stored in vector $\mathbf{a}_{\text{m}}$.

Second, the single entry $a_{\max}$ with maximum absolute value of $\mathbf{a}_{\text{m}}$ is found. In this manner, a single pivot is chosen as the largest number among the set of candidates selected in the symbolic stage. This approach improves numerical stability of GE.

For the previous Example 1 the corresponding values of the numeric stage are shown below (Example 2).

$$\forall m \in \mathbf{v}_{\text{cmin}}; \quad \mathbf{a}_{\text{m}} = \mathbf{A}(m) \tag{26}$$

$$a_{\max} = \max(|\mathbf{a}_{\text{m}}|) \tag{27}$$

Example 2. According to (26)$a_m = A(3,2) \cup A(2,4)$; $a_m = [2,4]$; following (27)$a_{\max} = \max[2,4]$; $\Rightarrow a_{\max} = 4 \Rightarrow pivot = A(2,4)$

The pivot position is saved in the permuted Identity matrices **P** and **Q**, by changing pivot row with row in matrix **P** and pivot column with column in matrix **Q**. Next, the pivot is placed as the first element of the matrix **A** by switching the pivot column with the first column and the pivot row with the first row $a_{11} \leftrightarrow a_{\max}$ (Example 3). One step of the process of GE is then carried out (18), and the size of matrices **A**, **P** and **Q** is reduced by 1 by cutting out its first row and first column.

Example 3. Following Examples 1 and 2 we have an examplary matrix A, with the $pivot = A(2,4)$ chosen ($A(2,4) \leftrightarrow A(1,1)$).

$$A = \begin{bmatrix} 7 & 0 & 0 & 0 \\ 3 & 0 & -5 & \mathbf{4} \\ 1 & 2 & 0 & 0 \\ -8 & 0 & -9 & 0 \end{bmatrix}; \rightarrow P \cdot A = \begin{bmatrix} 3 & 0 & -5 & \mathbf{4} \\ 7 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ -8 & 0 & -9 & 0 \end{bmatrix}; \rightarrow P \cdot A \cdot Q = \begin{bmatrix} \mathbf{4} & 0 & -5 & 3 \\ 0 & 0 & 0 & 7 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & -9 & -8 \end{bmatrix};$$

$$r2 \leftrightarrow r1 \quad P = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \quad c4 \leftrightarrow c1 \quad Q = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix};$$

This process applies continuously until the size of the matrix **A** reduces to 1 or matrix **A** becomes a full matrix. When matrix **A** is full, obviously the choice of the pivot based upon the sparsity feature becomes pointless. In this case a standard partial pivoting [14] is proceeded.

It is important to note, that after each loop, the first step of MCF evaluation will also count the new entries that appeared in the previous step of GE (fill-ins). This approach is essential for the best choice of the pivot, among all the candidates available. The extracted matrices **P** and **Q** for frequency ω_1 are the final preordering matrices and they will be applied to all the next matrices describing higher frequencies $\mathbf{A}_{\omega_2}, \ldots, \mathbf{A}_{\omega_{\max}}$ of the same device. These matrices will be solved using pure form of GE without pivoting, which is simple, fast and robust.

5. Results

This section presents the numerical results obtained from the examination of an exemplary FIT-type matrix. Two different matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, describing the same model, were generated in order to demonstrate how the pattern of the matrix evolves with the grid condensation. The pattern of the matrix $\mathbf{A}_1$ (size 47×47, 168 nze), generated with the grid $2 \times 3 \times 3$ is presented in Fig. 2 a. However, to obtain rationally accurate results, grid extension up to $6 \times 6 \times 8$ is

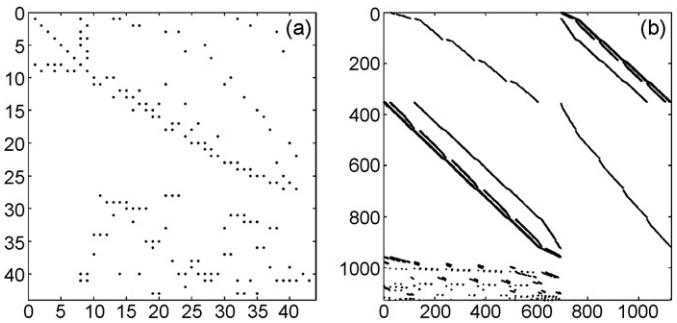

Fig. 2. FIT-type matrix: (a) $\mathbf{A}_1$ size 47×47 and (b) $\mathbf{A}_2$ size 1126×1126.

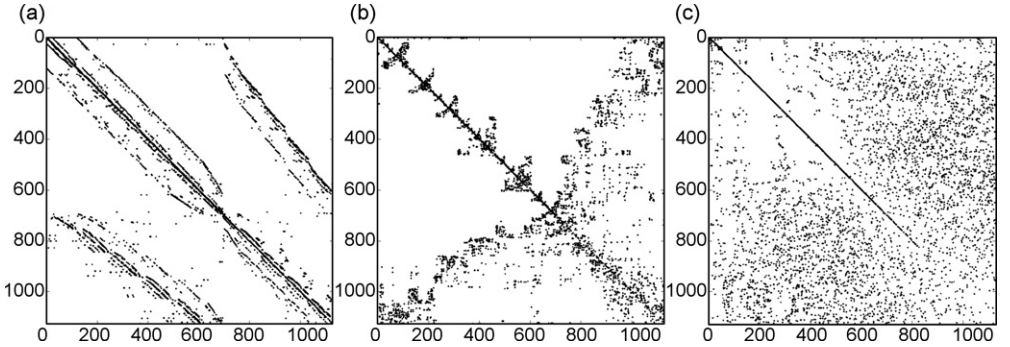

Fig. 3. Matrix $\mathbf{A}_2$ permuted by method: (a) Partial pivoting and (b) UMFPACK, (c) SMark.

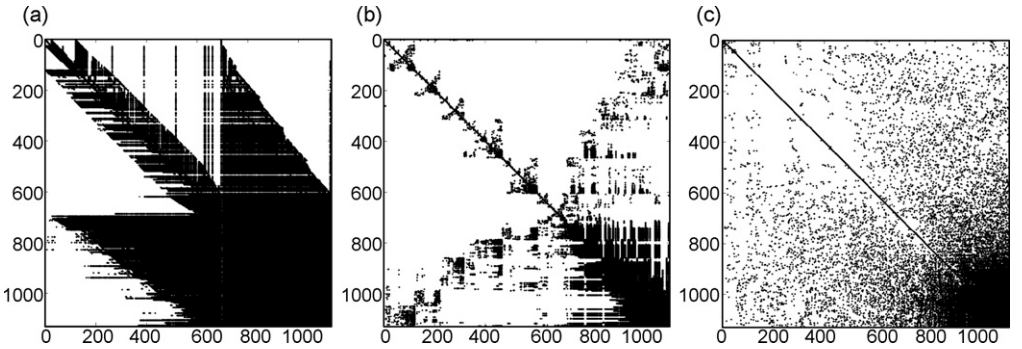

Fig. 4. The sum of the matrices **L** and **U** after GE performed on matrix $\mathbf{A}_2$ permuted by method: (a) partial pivoting, (b) UMFPACK and (c) SMark.

required. This effects in matrix $\mathbf{A}_2$ (size 1126 × 1126, 5366 nze) illustrated in Fig. 2b. In order to show the results of the real application problem we chose three different (low, average and high) operating frequencies of IC: f_1=1 GHz, f_2=10 GHz and f_3 = 60 GHz. Although by imposing the proposed frequencies to the matrix $\mathbf{A}_2$ the pattern remains the same, the numerical values dramatically change, what may disturb the stability of the Gaussian Elimination process.

First, the LU decomposition with Partial Pivoting (PP) was carried out in order to show the results of this simple approach. Next, to validate the quality of the SMark algorithm we made a comparison with currently the most popular and the most efficient method for unstructured sparse matrices—UMFPACK. Likewise via UMFPACK the comparison with the AMD algorithm has been done (for details see [5] pp. 140–141).

As shown in Fig. 3 the various reordering strategies result in significant differences in the $\mathbf{P} \cdot \mathbf{A} \cdot \mathbf{Q}$ pattern. This obviously influences the patterns of matrices **L** and **U** (Fig. 4), the nnz($\mathbf{L} + \mathbf{U}$) and the residual δ calculated as

$$\delta = \max(|\mathbf{LU} - \mathbf{PAQ}|)$$

.

The numerical values are presented in Table 1. Please notice, that from all the investigated algorithms SMark demonstrates the lowest number of elements in matrices **L** and **U**. The relative sparsity of the obtained solution was measured as the ratio of nnz in matrices **L** and **U** to the nnz in matrix **A**, so $g = ((nnz(\mathbf{L} + \mathbf{U}))/nnz(\mathbf{A})) \times 100\%$. Hence, the sparsity of SMark solution (758%) was larger compared with UMFPACK (1295%) and PP (6875%). Regarding

Table 1
Partial pivoting, UMFPACK and SMark comparison on matrix $\mathbf{A}_2$ size 1126 × 1126 with 5366 non-zero elements.

Preordering method	nnz(**L** +**U**)	g [%]	$\delta(f_1)$	$\delta(f_2)$	$\delta(f_3)$
Partial pivoting	368,475	6875	1e −15	1e −15	1e −15
UMFPACK	68,760	1295	1e −17	1e −17	1e −16
SMark	40,649	**758**	1e −9	1e −9	1e −10

the stability issue, all the algorithms in all frequencies were stable ($\delta \leq 1e-9$). However the fill-in reducing SMark algorithm was loaded with the maximal $\delta = 1e-9$ error on the border of stability (in the frequencies f_1 and f_2).

Unfortunately, in some cases SMark algorithm does not give reliable results and $\delta > 1e-9$ appears. This behavior intensifies with the matrix enlargement. We see two main reasons for the possible unstable result. The first and main issue is the fact, that symbolic factorization part appears as the first in the hierarchical algorithm scheme. Therefore, it may happen, that the vector of elements selected in symbolic part does not carry values able to guarantee the stability of the factorization process. The second reason is, that for large matrices a growth factor may cause a loss of precision. The explanation is given in [14] pp.163–171.

6. Conclusions

To conclude, it seems to be obvious that PP is not an appropriate method for sparse matrices and should not be used at all. The existing sparse matrix dedicated algorithms as SMark or UMFPACK are much more efficient.

The use of MCF is very beneficial in fill-ins number reduction appearing in the LU factorization process. Therefore it is essential to investigate the possibilities of its application and further modification.

The most important advantage of the SMark algorithm is that it takes into consideration the values of the fill-ins appearing in the process of GE. This guarantees the possibility of pivot choice not necessarily among the elements of the original matrix **A**, but also among the appearing fill-ins. Consequently, the SMark demonstrates a high fill-ins number reduction in the process of LU decomposition of FIT-type matrices.

On the contrary, in a general case, the stability-ensuring stage of the algorithm seems not to be sufficient. A series joint of symbolic-numeric architecture always gives priority only to one of the stages. However, a proper symbolic-numeric balance could improve the stability and guarantee $\delta \leq 1e-9$. The parallel algorithm architecture (Fig. 1b) can be considered as a promising direction to obtain a proper "fill-ins number reduction" — "stability" balance.

In future we plan to solve the stability problem by introducing a row scaling matrix **R**. Parallel symbolic-numeric algorithm architecture will be investigated. In order to obtain a good robust algorithm, a wide interdisciplinary cooperation with other institutes should be accomplished.

Acknowledgment

The authors would like to acknowledge the financial support of the European Commission under Marie Curie Fellowship program FP6/EST3.

References

[1] P. Amestoy, T. Davis, I. Duff, An approximate minimum degree ordering algorithm, SIAM Journal on Matrix Analysis and Applications 17 (1996) 886–905.
[2] P. Amestoy, I. Duff, Vectorization of a multiprocessor multifrontal code, International Journal of Supercomputer Applications 7 (1993) 64–82.
[3] G. Ciuprina, D. Ioan, D. Niculae, J. Villena, L. Silveira, Parametric models based on sensitivity analysis for passive components, studies in computational intelligence, in: Studies in Computational Intelligence, vol. 119, Springer, Berlin, Heidelberg, New York, 2008, pp. 17–22.
[4] T. Davis, Algorithm 832: Umfpack, an unsymmetric-pattern multifrontal method, ACM Transactions on Mathematical Software 30 (2004) 196–199.
[5] T. Davis, Direct Methods for Sparse Linear Systems, SIAM, Philadelphia, 2006.
[6] D. Ioan, G. Ciuprina, Parametric Models for Electromagnetic Field Systems Related to Passive Integrated Components, in: Proc. PIERS 2007, Beijing, China, pp. 1964–1968.
[7] D. Ioan, G. Ciuprina, Reduced order models of on-chip passive components and interconnects workbench and test structures, in: W. Schilders, H. Vorst, J. Rommes (Eds.), Model Order Reduction, vol. 13 of Mathematics in Industry, Springer, Berlin, Heidelberg, New York, 2008, pp. 447–467.
[8] H. Janssen, J. Niehof, W. Schilders, Accurate modeling of coupled functional rf blocks: Chameleon rf, in: G. Ciuprina, D. Ioan (Eds.), Scientific Computing in Electrical Engineering SCEE, vol. 11 of Mathematics in Industry, Springer, Berlin, Heidelberg, New York, 2007, pp. 81–87.
[9] J. Liu, Modification of the minimum degree algorithm by multiple elimination, ACM Transactions on Mathematical Software 11 (1985) 141–153.
[10] H. Markowitz, The elimination form of the inverse and its application to linear programming, Management Science 3 (1957) 255–269.
[11] I. Munteanu, T. Weiland, Rf & microwave simulation with the finite integration technique from component to system design, in: G. Ciuprina, D. Ioan (Eds.), Scientific Computing in Electrical Engineering SCEE, vol. 11 of Mathematics in Industry, Springer, Berlin, Heidelberg, New York, 2007, pp. 247–260.

[12] J. Plata, M. Dobrzynski, Solving methods for sparse matrices in modeling of em effects in nano ic, in: Proc. IPMS'08, Literatur Yaymcilik Ltd., 2008.
[13] O. Schenk, K. Gartner, Solving unsymmetric sparse systems of linear equations with pardiso, Journal of Future Generation Computer Systems 20 (2004) 475–487.
[14] L.D.B. Trefethen III, Numerical Linear Algebra, SIAM, Philadelphia, 1997.
[15] T. Weiland, A discretisation method for the solution of Maxwell's equations for six-component fields, International Journal of Electronics and Communication AEU 31 (1977) 116–120.