Solving methods for sparse matrices in modeling of EM effects in nano IC

Jagoda Plata and Michał Dobrzyński
Numerical Methods Laboratory,
"Politehnica" University of Bucharest,
Splaiul Independentei 313, Bucharest, ROMANIA
e-mail: plata@lmn.pub.ro, d_michal@lmn.pub.ro

Introduction

The present paper describes problems arose within the development of the automation design tools in the European research project CHAMELEON RF. Comprehensive High-Accuracy Modeling of Electromagnetic Effects in Complete Nanoscale Radio Frequency blocks CHAMELEON RF is a project which aims to deliver the methodologies and prototype tools to enable the analyses of complete RF blocks considered as one entity.

Problem description

Modeling of passive on chip components and, in particular, the stage of solving Partial Differential Equations of the electromagnetic effects at high frequency complex linear systems, leads to large and sparse matrices. Computational methods leading to the solution of $AX=B$, where A is large (i.e. 6.600 x 6.600 up to 101.000 x 101.000) and sparse, tend to effect a large memory usage and an extended CPU time to run the operation. The two problems extremely arise with an enlarged size of the matrices followed by increasing the so called number of DoFs - degrees of freedom, which correspond with a finite number of variables computing electric circuits with distributed parameters elements. The results of time and memory consumption used during computation are not satisfying; moreover very large systems cause a failure of the utilized solver and the solution cannot be found.

In the research existing methods and algorithms of solvers for sparse matrices have been studied in order to improve the solver and to receive better time and memory usage results in its operation as well as to enable its operation for very large matrices at all. However, direct solvers for sparse matrices intent to involve more complicated algorithms than for dense matrices.

Results

An evaluation of existing solvers indicated the direct solver implemented in the UMFPACK solve package as the fastest and most accurate tool for running the operation of finding the solution of an equation $AX = B$. The solver performs sparse LU factorization [2] with row scaling computed by a complex variant of Gaussian Elimination method with partial pivoting, such that:

$$PR^{-1}AQ = LU \tag{1}$$

Then X is computed by solving permuted triangular systems:

$$X = QU^{-1}L^{-1}PB \tag{2}$$

UMFPACK benefits from the algorithm of Markowitz to reduce the number of fill-ins in each stage and preserve the sparsity during the computation. At each stage of decomposition

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \left(\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}\right) a_{kj}^{(k)} \tag{3}$$

the Markowitz measures are created

$$f(a_{ij}^{(k)}) = (r_i - 1)(c_j - 1) \qquad a_{ij}^{(k)} \neq 0 \tag{4}$$

and a set of elements with minimum Markowitz measure are chosen for a pivot. The strategies of UMFPACK include also fill-reducing column pre-ordering and symbolic factorization [1].

The graphical results (Figure1) demonstrate the patterns of matrices L and U obtained by 3 different methods of computing the decomposition of matrix A, size 6.608 x 6.608.

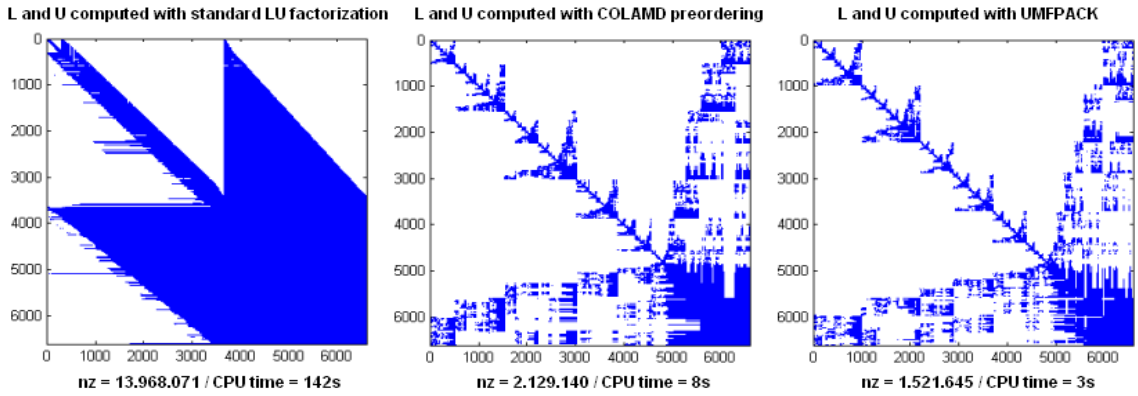

Figure 1: Decomposition matrices for (a) simple LU factorization with partial pivoting; (b) LU factorization with Column Approximate Minimum Degree (COLAMD) permutation pre-ordering; (c) UMFPACK factorization

The CPU time of performing the operation in each case was measured and the number of non zero entries of L and U illustrates the nz-number below the picture.

Conclusions

As the sparse direct solution methods considered in solver application perform the triangular factorization LU, the main problem is to keep the upper U and lower L triangular matrices as sparse as possible. Physically maintaining the sparsity of the matrices can be controlled by efficient minimizing the number of fill-ins in the factors of L and U performing the LU factorization of the original matrix A.

The matrix A analysis reveals its characteristics independent from size (number of DoFs) and frequency sampling. The matrix is always sparse, square, complex, unsymmetrical, unstructured, although has a certain form of a pattern. This implies the possibility of applying the matrix features in the solver code.

Further work

There is no single algorithm that is the best for all types of linear systems and for this reason the researches on the development of a new solver are targeted only for special matrices and not reflect the general case. The solver must be designed based on the characteristics and architectural features of matrix A. The performed studies will be especially useful in the next stage of our further research.

References

[1] Saad Yousef, *Iterative Methods for Sparse Linear Systems.* PWS Publishing Company, Boston, 1996.
[2] Lloyd N. Trefethen, David Bau III, *Numerical linear algebra.* SIAM, Philadelphia, 1997.