

# Finite Elements and Absorbing Boundary Conditions for Scattering Problems on a Parallel Distributed Memory Computer

C. Voltaire, L. Nicolas, A. Nicolas

Ecole Centrale de Lyon - CEGELY - U.P.R.E.S.A C.N.R.S. 5005, B.P.163 - 69131 Ecully cedex - France

**Abstract**—An efficient finite element method is presented to compute time harmonic microwave field in three dimensional configurations. Nodal-based finite elements have been coupled with an absorbing boundary condition to solve open boundary problems. Modeling real devices is made possible using parallel computation. This paper describes first the time harmonic formulation with the absorbing boundary condition and the sequential code. A new algorithm is then proposed to implement this formulation on a cluster of workstations (10 DEC ALPHA 300X). Analysis of efficiency is performed using simple problems. The electromagnetic scattering of a plane wave by a perfect electric conductor cylinder is finally given as example.

## I. INTRODUCTION

Nodal-based Finite Element (FE) method has been previously used to modelize microwave problems [1]. Absorbing Boundary Conditions (ABC) have been introduced to simulate open boundary domains. The time harmonic formulation is written in terms of vector fields. Because nodal-based finite elements are used, a penalty term is added to the formulation in order to avoid spurious reflexions.

This code operates on scalar workstation: only simple problems can be modeled. For instance, a 3 wavelengths side cubic geometry, meshed with 10 nodes by wavelength, leads to 81000 complex unknowns. Actually, only parallel computation enables to model real devices because it reduces the computation time and mainly arranges enough memory. For test considerations, we use a cluster of ten DEC ALPHA workstations (80 Mflops linpack, 64 Mo of RAM, 1 Go of swap per processor) linked by a FDDI ring. This distributed memory computer is a Multi Instruction - Multi Data streams (MIMD) type. Parallel Virtual Machine (PVM) software is used to pass messages between workstations.

The objective of this paper is to describe how the existing code has been modified in order to implement it on such a computer. Algorithms designed to operate on scalar calculators do not fit to parallel distributed memory computers because data are interdependant. It is not sufficient to add directives of messages passing in a program to allow good performances and algorithms have to be deeply modified. To implement this formulation on such a computer, we use an algorithm which distributes data over the different workstations. The developed code is a Single Program Multi Data type (SPMD). This allows to minimize messages passing which step is CPU consuming. Performances are then analyzed for simple problems. Speed-up, Amdahl's law and

data volume switched are presented for the different steps of the program. The answer of a  $6\lambda$ -length cylinder, modeled as a perfect electric conducting (p.e.c.), enlightened with an electromagnetic plane wave, is finally presented.

## II. FE FORMULATION COUPLED WITH ABC

We are dealing with frequency domain open boundary electromagnetic field problems. According to Maxwell equations, the magnetic field  $\mathbf{H}$  and the electric field  $\mathbf{E}$  satisfy the vector wave equations. Following developments are made only for the  $\mathbf{H}$  field formulation. All the steps can be applied to the  $\mathbf{E}$  field formulation as well. The weak Galerkin formulation of the vector wave equation for  $\mathbf{H}$  is given by:

$$\int_V \left[ (\nabla W \times \frac{1}{\epsilon_r} \nabla \times \mathbf{H}) - W k_0^2 \mu_r \mathbf{H} \right] dv - \int_S \left[ \mathbf{n} \times (W \frac{1}{\epsilon_r} \nabla \times \mathbf{H}) \right] ds = 0 \quad (1)$$

A penalty term is added to the formulation to avoid spurious reflexions [1]. It makes the field divergence free. ABC are used to truncate the 3D finite element region. The open boundary is modeled using a second order 3D vector Enquist-Majda ABC [2]. Outer surfaces are then rectangles, which is less mesh-consuming for a large number of geometries:

$$\mathbf{n} \times \nabla \times \mathbf{H} \cong g_{ABC}(\mathbf{H}) = j k_0 \mathbf{H}_t - \frac{j}{2k_0} \nabla_t^2 \mathbf{H}_t \quad (2)$$

Finally, the formulation for scattering problems in term of total field is given by (3). Two types of surface are considered: external surface ( $S_{ext}$ ) and pec surface ( $S_{pec}$ ).

$$\int_V \left[ (\nabla W \times \frac{1}{\epsilon_r} \nabla \times \mathbf{H}) + W k_0^2 \mu_r \mathbf{H} \right] dv - \int_V [(\nabla W)(\nabla \cdot \mathbf{H})] dv + \int_{S_{ext+Spec}} W \mathbf{n} \cdot \nabla \cdot \mathbf{H} ds - \int_{S_{ext}} W g'_{ABC}(\mathbf{H}) ds = \int_{S_{ext}} W [g_{ABC}(\mathbf{H}_i) - \mathbf{n} \times \nabla \times \mathbf{H}_i] ds \quad (3)$$

Where  $\mathbf{H}$  is the total field and  $\mathbf{H}_i$  the incident field.

### III. SEQUENTIAL CODE

Main steps of the existing code are:

- 1) *Create the structure of the global matrix:* symbolic assembling.
- 2) *Assemble the global matrix:* each elementary matrix (related to a volume finite element) is computed and results are then put in the global matrix.
- 3) *Introduce ABC on the external boundaries and Boundary Conditions (BC) on conductors:* only surface elements are sought.
- 4) *Introduce BC on the symmetry planes.*
- 5) *Symmetrization of the matrix:* the ABC step (#3) results in a non-symmetric global matrix. The FE matrix is symmetrized by adding it to its transposed matrix. This approximative method has been shown to give good results
- 6) *Solve the system of equations:* because the matrix system is sparse, iterative solvers have been implemented.

### IV. PARALLEL FE ALGORITHM

#### A. Distributing the data

We have chosen to duplicate the entire data file on each processor. Indeed, the program is a SPMD type performing a parallel reading. An other strategy would consist in storing no more than what is required for one processor at a time and in reading and updating data files during the processing. But this method would result in unacceptable I/O overhead. A compromise is available in the case of distributed memory architectures [3]-[5].

#### B. Assembling in parallel

The assembling step of the global matrix is performed by degree of freedom [6], instead of by element as in the sequential code. It allows to assemble at once the three lines (three coordinates) corresponding to a node. Each element including the considered node are sought and contributions with the related nodes are computed. These three lines are then compressed by storing only non-zero values. Space is left in the compressed storage if the volume element includes a surface element. Same method is used to introduce ABC and BC on conductors but only surface elements are sought.

This method does not require the creation of the structure of the matrix. To solve a problem meshed with N nodes, if W workstations of the cluster are available, each workstation assembles a part of the system including N/W nodes. The load balancing is nearly perfect because of the constant bandwidth of the global matrix. The program is a SPMD type: no message passing is required. Hence, the speedup of the assembling stage is optimal (Fig. 4)

#### C. BC on symmetry planes and conductors

Messages passing are necessary to introduce the BC on conductors (**E** field solving only) and on the symmetry planes because a global modification of the matrix system is required (due to the method used to introduce BC) (Fig. 1). On a symmetry plane the H field verifies (6a) or (6b):

$$\text{On a symmetry plane:} \quad \mathbf{n} \cdot \mathbf{H} = 0 \quad (6a)$$

$$\text{On an antisymmetry plane:} \quad \mathbf{n} \times \mathbf{H} = 0 \quad (6b)$$

The speedup related to this step is very bad due to the large amount of message passing (Fig. 4). However, its effect on the global performances of the code is negligible: the length of this step is less than one percent (**H** field solving) of the total computation duration (Fig. 2).

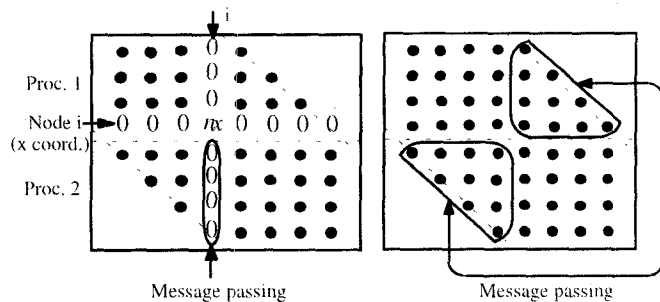


Fig. 1. Message passing for the symmetry planes (left) and for symmetrization (right) - Example with 2 processors.

#### D. Symmetrizing the global matrix

The introduction of ABC and penalty function results in a non-symmetric system matrix. This one is approximately symmetrized by adding it to its transposed matrix. This operation needs a large amount of messages passing (Fig. 1). The speedup related to this operation is also poor (Fig. 4). But, once more, the time needed to perform this step is small compared to the total computation time (Fig. 2).

#### E. Solving in parallel

Quasi Minimum Residu (QMR) and Conjugate Gradient (CG) with diagonal or incomplete Choleski preconditioner have been implemented [7], [8]. Because these methods are iterative, small-scale parallelism is used. Parallel matrix multiplication allows to compute the residual vector of each iteration: each processor computes a partial residual vector and *broadcasts* it to all the others (only non-zero values are sent). Then, each processor adds these partial residual vectors to obtain the final residu (SPMD). An other way to calculate the residual vector is to *send* each partial residual vector to one processor. This one computes the final residu and

*broadcasts* it. This Master / Slave (M/S) method minimizes the number of communications.

The cost of communications in this step is penalizing, because small-scale parallelism is not adapted to distributed memory computer. Some others methods can be found in [9] but they are more memory space consuming.

V. PARALLEL PERFORMANCES

PARAGRAPH is used to analyse the performances of the code. For such a purpose, 2 small problems (scattering by a pec cylinder) have been modeled: the first one is meshed with 4000 nodes, leading to 12000 degrees of freedom, and the second one is meshed with 10000 nodes, leading to 30000 degrees of freedom.

A. CPU time distribution

CPU times to solve both of these problems on 8 processors were, respectively, 236 s and 688 s. Fig. 2 shows the CPU time distribution for the main steps of the code (8 CPU's). The solving step is the most CPU consuming because its speedup is very poor (Fig. 4).

Note that in a sequential computing, the assembling step would spend 60% of the CPU time and the solver only 35%.

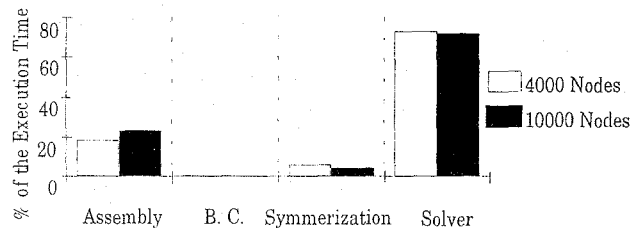


Fig. 2. CPU time distribution (8 CPU's).

B. Analysis of a 4000 nodes problem

For the problem meshed with 4000 nodes, Fig. 3 shows the state of the eight processors (busy, overhead or idle) and Table I shows the data quantity switches. This analysis has been done for both of the programming types of the solver. The M/S method is performed with the processor #1 as Master. The idle time corresponds to synchronisation phases. The overhead time is introduced by the messages passing. From Fig. 3, the SPMD type appears more efficient. However, for large problems, it can imply a saturation of the communication network (Table I). In fact, SPMD type seems to be more adapted to a 'crossbar' network. The data quantity switches remain approximatively the same for both types of programming, but the number of messages decreases for the M/S type. The load balancing is good for both programming types.

TABLE I  
DATA SWITCHES (4000 NODES)

processor number	number of messages		Mb received		Mb sent	
	M/S	SPMD	M/S	SPMD	M/S	SPMD
1	3171	3171	99.2	99.2	643	81
2	471	3171	93.2	95.7	16	106
3	471	3171	93.3	96.2	15.8	102
4	471	3171	93.28	95.8	16.3	105
5	471	3171	93.31	97.1	15.8	96
6	471	3171	93.2	97.3	14.7	94.6
7	471	3171	93.24	97	15	95
8	471	3171	93.46	97.3	13.7	94.7

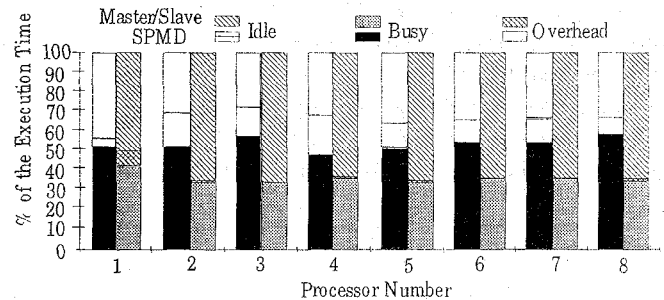


Fig. 3. Processors states for a 4000 nodes problem.

C. Speedup versus number of processors

For a given problem, the sequential ratio of the code remains constant but the cost of communications increases with the number of processors. Parallel ratio given by the Amdahl's law increases slightly with the number of processors (Table II and Fig. 4).

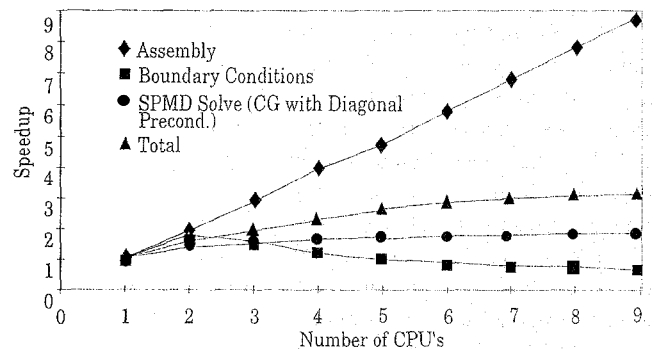


Fig. 4. Speedups for a 10000 nodes problem.

TABLE II  
PROBLEM MESCHED WITH 10000 NODES

CPU's	2	3	4	5	6	7	8	9
Speedup	1.5	1.9	2.2	2.5	2.8	3	3.17	3.21
Amdahl's law	66%	71%	72%	75%	76%	77%	78%	77%

#### D. Speedup versus number of nodes

From Table III, it is obvious that the efficiency of the method increases with the size of the computed problem: this is due to the communication times which become negligible compared to computation times.

When computing large problems, it becomes difficult to measure the code performances because these problems cannot be computed on only one processor (the speedup can not be calculated !). A theoretical variation of the CPU time on 1 processor depending on the number of mesh nodes ( $n$ ) is then used. This theoretical variation is calculated from the actual variation obtained for problems meshed with 10000 nodes at most, together with the knowledge that solving time varies as  $n^2$  (Table IV).

TABLE III  
RESULTS FOR 8 CPU'S

number of nodes	total speedup	solving speedup	Amdahl law (total speedup)
3000	2.35	1.2	65 %
4000	2.62	1.56	70 %
10000	3.17	1.76	78 %
15000	3.61	1.81	82.6%

TABLE IV  
RESULTS FOR 8 CPU'S  
(CPU TIME ON 1 PROCESSOR HAS BEEN ESTIMATED)

number of nodes	total speedup	Amdahl law
22000	3.9	84.9%
38000	4.2	87%
52000	4.5	88.8%
65000	4.9	90%

#### VI. MODELING THE SCATTERING BY A $6\lambda$ CYLINDER

In this section, we present the example of a cylinder enlightened with an electromagnetic plane wave (Fig. 5). The length of the cylinder is  $6\lambda$ , and the radius is  $0.5\lambda$ . The frequency of the incident wave is 3 GHz. Problem has been modeled with 1 symmetry. To obtain accurate results, 10 nodes per wavelength are used [1], leading to a mesh made of 50000 nodes. Note that the space memory available on the cluster of stations allows to solve problems meshed at most with 100000 nodes.

#### VII. CONCLUSION

We have presented in this paper the implementation of an electromagnetic scattering code on a parallel memory computer. Parallel performances obtained when building the FE matrix are satisfying. On an other hand, the cost of communications in the solver is penalizing, because small-

scale parallelism is not adapted to distributed memory computer.

The modeling of realistic problem is not possible on such a cluster because of the memory available. As example, the study of an 15 m-length airplane, even simplified as a perfect electric conducting body, enlightened by a 3 GHz plane wave, would lead immediately to a problem meshed with more than  $10^6$  nodes.

On an other hand, this code is immediately implementable on a computer such as the CRAY T3D. This parallel distributed memory computer is a MIMD type. Given its characteristics (128 processors and 8 Go of memory), it should allow to arrange enough memory to modelize realistic devices.

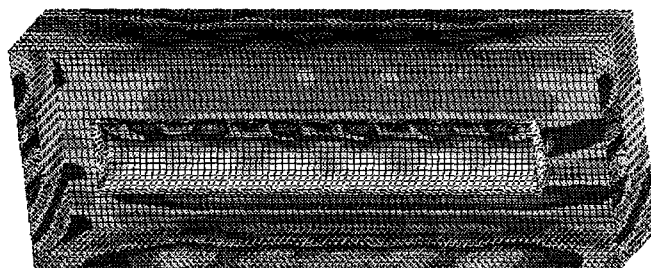


Fig. 5. P.E.C. cylinder meshed with 50000 nodes - Magnitude of the magnetic field.

#### VIII. REFERENCES

- [1] L. Nicolas, K. A. Connor, S. J. Salon, B. G. Ruth, and L. F. Libelo, "Three dimensional FE analysis of high power microwave devices," *IEEE Trans. Mag.*, no. 2, pp. 1642-1645, March 1993.
- [2] J. L. Yaobi, L. Nicolas, and A. Nicolas, "Vector absorbing boundary conditions for nodal or mixed based finite elements," to be published in *IEEE Trans. Mag.*, May 1996.
- [3] G. J. Bendzsak, and T. W. Ma, "Parallel computation of 3-D electric and magnetic fields," *IEEE Trans. Mag.*, vol. 27, no. 5, pp. 4205-4209, September 1991.
- [4] S. Ratnajeevan, and H. Hoole, "Finite element electromagnetic field computation on the sequent symmetry 81 parallel computer," *IEEE Trans. Mag.*, vol. 26, no. 2, pp. 837-840, March 1990.
- [5] M.L.Barton, "Three-dimensional magnetic field computation on a distributed memory parallel processor," *IEEE Trans. on Mag.*, vol. 26, no. 2, pp. 834-836, March 1990.
- [6] D. Zois "Parallel processing techniques for FE analysis: stiffnesses, loads and stresses evaluation," *Comp. & St.*, vol. 34, no. 3, pp. 355-374, 1990.
- [7] R. W. Freund, "Conjugate gradient type methods for linear systems with complex symmetric coefficient matrices," *SIAM J. Stat. Comput.*, vol. 13, no. 1, January 1992.
- [8] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal, "An implementation of a look-ahead Lanczos algorithm for non-hermitian matrices. Part I," *Tech. Rep. 90.45, RIACS, NASA, Ames Research Center*, November 1990.
- [9] R. F. Lucas, T. Blank, and J. J. Tiemann, "A parallel solution for large sparse systems of equations," *IEEE Trans. Comp. Aided Des.*, vol. 6, no. 6, pp. 981-991, November 1987.