Results of Parallelisation of Existing Electromagnetic Finite Element Codes on Some Different Parallel Architectures

M. Dracopoulos, K. Parrot
Oxford Parallel, University of Oxford, OUCL, Parks Rd., Oxford OX1 3QD, UK

G. Molinari, M. Nervi
Dipartimento di Ingegneria Elettrica, Università di Genova
11a, Via Opera Pia, I-16145 Genova, Italy

J. Simkin
Vector Fields ltd., 24 Bankside, Kidlington, Oxford OX5 1JE, UK

Abstract - In this paper the results obtained from the parallelisation of some 3D industrial electromagnetic Finite Element codes within the ESPRIT Europort 2 project PARTEL are presented. The basic guidelines for the parallelisation procedure, based on the Bulk Synchronous Parallel approach, are presented and the encouraging results obtained in terms of speed-up on some selected test cases of practical design significance are outlined and discussed.

I. Introduction

The present work was carried out, within the ESPRIT programme of the European Union, as part of an initiative named Europort, a parallelisation exercise aiming at producing efficient parallel versions of existing commercial software codes on a wide range of parallel architectures, from workstation networks to shared memory supercomputers.

The PARTEL project, included in the Europort 2 cluster, was focused on parallelising the Finite Element electromagnetic analysis codes TOSCA, SCALA and ELEKTRA made available by Vector Fields, the code owner and project partner. Other partners were Oxford Parallel, the parallel application centre of Oxford University, acting as parallel expert, and the Philips Research Laboratories in Eindhoven as well as the Department of Electrical Engineering of the University of Genova acting as end users.

The codes cover a rather wide set of electromagnetic analysis subdomains, ranging from linear and nonlinear static problems to eddy current and particle beam modelling; test cases have been selected by the end users among real and rather different design problems, trying to obtain information on the widest possible range of situations of practical interest.

In this paper the approach selected for the parallelisation is outlined, the structure of the codes and the range of parallel machines used are briefly described and the results of some test cases, prepared and run at the Department of Electrical Engineering of the University of Genova, are summarised.

Manuscript received September 23, 1996.
M. Dracopoulos, e-mail: Mike.Dracopoulos@comlab.ox.ac.uk
G. Molinari, e-mail: molinari@die.unige.it
M. Nervi, e-mail: nervi@die.unige.it
K. Parrott, e-mail: Kevin.Parrott@comlab.ox.ac.uk
J. Simkin, e-mail: js@vector.demon.co.uk

II. The Parallelisation Strategy

As usual for the parallelisation of Finite Element codes, the parallelisation strategy is driven by a geometric decomposition of the application domain, exploiting the intrinsic parallelism in the element-based computations and the assembly of the subdomain stiffness matrices. Since the codes under parallelisation already use iterative solvers for the large, sparse matrices they generate, a straightforward extension was introduced to take advantage of the inherently parallel nature of the solvers within the subdomain partitioning framework.

Another challenging problem was related to the parallel evaluation of the R.H.S., because the quadrature routines evaluating the fields are adaptive, and therefore the computational load can be largely unpredictable, and is not suitable for a simple geometric partitioning. A cyclic data-driven distribution has therefore been used

In order to achieve maximum portability, various parallelisation models and their related libraries were considered. Even though message passing is the dominant model in use in parallel programming (mainly as implemented in PVM and MPI), a radically different approach was adopted for the PARTEL project based on the Bulk Synchronous Parallel (BSP) paradigm, proposed by Valiant [1]. The BSP model provides a general purpose communication protocol covering both distributed and shared memory or virtual-shared-memory multiprocessors. It can be naturally implemented on message passing architectures, but it is not finalised towards message passing semantics. By adopting the general shared-memory semantics, the BSP model offers ease of use, flexibility and simplicity. Furthermore, the model requires only minimal modifications to existing sequential code, assuming that the native structure of the application is data parallel. Moreover, BSP provides means for the performance prediction of an algorithm across a wide range of parallel architectures. The development of the BSP and its related tools is a project currently under development at Oxford University, and is currently going through a standardisation procedure [2].

The BSP approach makes use of two simplifying assumption: the first one is that any parallel computer can be described as a number of processors, each with some local memory; a communication mechanism, for the exchange of data between processors, and a synchronisation mechanism, for synchronising a group of processors at a common barrier point. The second one is that a parallel computation can be organised into a sequence of supersteps; during each superstep the processors can work independently on local data, and initiate requests for the asynchronous reading and writing of non local data to be used during a subsequent superstep. At the end of a single superstep, the processors synchronise and all data exchanges are completed, before proceeding to the next superstep.

The parallel implementation of the Finite Element method here considered can be easily expressed as a BSP algorithm, that is as superstep of local computation and asynchronous remote data access, separated by a global synchronisation. However, as the necessary primitives for doing this are not always portable, particularly in message passing libraries, a compact and portable BSP providing the one-way communication and global synchronisation routines was used.

Efficient implementations were available on a number of different categories of parallel systems, including those used within the project (Meiko CS2, IBM SP2, SGI Power Challenge and DEC Alpha networks).

III. The Partel Codes

A. TOSCA

The TOSCA program is used to model static magnetic, electric and current flow fields in three dimensional problem domains.

In the magnetic case, it solves a generalised form of Poisson's equation based on a combination of total and reduced scalar potentials [3]. The equation is in the form:

$$\nabla \cdot \mu(\mathbf{H})\, \nabla\Phi = -\nabla \cdot \mu(\mathbf{H})\, \mathbf{H}_s + \mu(\mathbf{H})\, \mathbf{H}_c \qquad (1)$$

where $\mathbf{H}$ is the magnetic field intensity, the magnetic scalar potential Φ is defined by the relation $\mathbf{H} = -\nabla\Phi$, $\mathbf{H}_c$ is the coercive field strength of magnetic materials and $\mathbf{H}_s$ is the coil field, obtained from Biot-Savart's equation as:

$$\mathbf{H}_s = \iiint \frac{\mathbf{J} \times \mathbf{r}}{|\mathbf{r}|^3} dv \qquad (2)$$

Tosca solves the problem defined by (1) using the Galerkin weighted residual method. Space is subdivided into a set of contiguous elements and within each element a low order polynomial is used to interpolate the magnetic scalar potential. The weak form of (1) is therefore the following:

$$\iiint \nabla W \cdot \mu(\mathbf{H})\, \nabla \sum_i N_i \Phi_i dv = \\ = -\iint W\mu(\mathbf{H}) \frac{\partial\Phi}{\partial n} ds \; - \; \iiint W\nabla\mu(\mathbf{H})(\mathbf{H}_s - \mathbf{H}_c) dv \qquad (3)$$

The system of algebraic equations obtained by the discretisation of (3), by choosing as weighting functions W the shape functions N used to approximate the potentials is highly and irregularly sparse, symmetric, definite positive and nonlinear. Most of the total solution time is usually spent in the calculation of the R.H.S. and in the solution of the algebraic system of equations.

B. SCALA

The SCALA program is used to model the steady state behaviour of charged particle beam devices in three dimensions [4]. It calculates electrostatic fields in the presence of space charge distributions arising from charged particle beams: in the first phase the electric field, expressed in terms of the electric scalar potential, is calculated through Poisson's equation:

$$\nabla \cdot \varepsilon \nabla V = \rho(x, y, z) \qquad (4)$$

where ε is the permittivity and ρ the space charge density. If desired, a magnetic field can be superimposed on the model, so as to simulate devices like cathode ray tubes, et cetera... SCALA solves the equation (4) using the finite element method, following the same procedures already described for TOSCA. The weak form of the equation becomes then:

$$\iiint \nabla W \cdot \varepsilon \nabla \sum_i N_i V_i dv = -\iint W\varepsilon \frac{\partial V}{\partial n} ds \; - \; \iiint W\rho(x, y, z) dv \qquad (5)$$

As in the case of TOSCA, the resulting matrix is highly and irregularly sparse, symmetric, definite positive and nonlinear. After the calculation of the fields, the program calculates the trajectories of a representative set of particles in the beams. A macroscopic current, determined by the physical law of the emission at the emitting surface, the prevailing electric fields and the area of the emitting surface related to the particle, is then associated with each particle. The set of particles is then intersected with the Finite Element mesh, and the space charge density in the beams is incorporated into the Finite Element model, using the element shape functions to interpolate its spatial variation. The electrostatic potential is then computed again by using an updated ρ field. The iterative procedure continues until the convergence of the potential is reached within a fixed tolerance.

The total solution time is dominated in most cases by the calculation of particle beams.

C. ELEKTRA

The ELEKTRA program is used to model quasi-static time-varying magnetic fields in three dimensional space. It solves the problem written in terms of a formulation that in non-conducting space is the same of TOSCA, and in conducting regions is based on the magnetic vector potential and on the electric scalar potential [5], in the form:

$$\nabla \cdot \mu(\mathbf{H})\, \nabla\Phi = -\nabla \cdot \mu(\mathbf{H})\, \mathbf{H}_s + \mu(\mathbf{H})\, \mathbf{H}_c \qquad (6)$$

$$\nabla \times \frac{1}{\mu} \times \mathbf{A} = -\sigma \frac{\partial \mathbf{A}}{\partial t} - \sigma \nabla V \qquad (7)$$

where $\mathbf{H}$ is the magnetic field intensity, $\mathbf{H}_s$ is the field due to the sources obtained through the integration of the Biot-Savart equation, $\mathbf{H}_c$ is the coercive field strength of magnetic materials, $\mathbf{B}$ is the magnetic induction, $\mathbf{A}$ is the magnetic vector potential defined by the equation $\mathbf{B} = \nabla \times \mathbf{A}$ and Φ is the magnetic scalar potential defined by the relationship $\mathbf{H} = -\nabla\Phi$.

For the solution, the Galerkin version of the F.E.M. is used also in this case. Very schematically, the weak form of the system of partial differential equations becomes of the kind:

$$\iiint \nabla W \cdot \mu(\mathbf{H})\, \nabla \sum_i N_i \Phi_i dv = \\ = -\iint W\mu(\mathbf{H}) \frac{\partial \Phi}{\partial n} ds \; - \; \iiint W \nabla \mu(\mathbf{H})(\mathbf{H}_s - \mathbf{H}_c) dv \tag{8}$$

$$\iiint \nabla \times \mathbf{W} \cdot \frac{1}{\mu(\mathbf{B})} \nabla \times \sum_i \mathrm{N}_i \mathbf{A}_i \mathrm{dv} = \\ = \iint \mathbf{W} \cdot \frac{1}{\mu(\mathbf{B})} \left(\nabla \times \sum_i \mathrm{N}_i \mathbf{A}_i \times \mathbf{n} \right) \mathrm{ds} - \\ - \iiint \mathbf{W}\sigma \left(\frac{\mathrm{d} \sum_i \mathrm{N}_i \mathbf{A}_i}{\mathrm{dt}} + \nabla \sum_i \mathrm{N}_i \mathrm{V}_i \right) \mathrm{dv} \tag{9}$$

As for the previous codes, the shape functions are such that a highly sparse coefficient matrix is obtained. However, as a function on the type of analysis, the implementation can be very different, and the code is in fact a cluster of three different solvers, SS, TR and VL, that can generate coefficient matrices quite different from each other. In particular, ELEKTRASS is used to model steady state a.c. problems through the complex time harmonic substitution, and its coefficient matrix is then complex and symmetric; ELEKTRATR is used to model general, transient, non-periodic problems by means of a time stepping integration procedure, and its coefficient matrices is real, symmetric and indefinite; ELEKTRAVL is used to model problems where the currents are induced by the motion of a conducting body within a magnetic field, and gives rise to real, non-symmetric matrices. Even if the situation may vary significantly according to the specific situation, most of the total solution time is generally spent in the calculation of the R.H.S. and/or in the solution of the algebraic system of equations, especially in the case of ELEKTRATR, for which the solution of the algebraic system is performed at every time step, necessarily in a serial way.

IV. Parallel Solution and Geometric Decomposition

As previously mentioned, a typical parallel implementation of the Finite Element method is based on a geometric partitioning of the mesh and a subsequent one-to-one mapping of the resulting subdomains to the available processors. This approach allows operations related to groups of elements (element stiffness matrix computations) to be performed independently within each subdomain. Upon entering the solution phase, each processor holds a subdomain stiffness matrix and "local" vectors (i.e. vectors related to a reduced set of nodal degrees of freedom). It is then apparent that all processors execute the same program on a different set of data (SPMD paradigm). In addition, one of the processors is responsible for serial, conventional I/O.

Apart from an initial data distribution and occasional gathering of results on the I/O processor, this model requires communication only during the solution phase. A geometric partitioning of a Finite Element mesh can be either element or node driven. A disjoint element partition assumes that an element can be owned by one processor only and consequently certain nodes in the mesh (interface nodes) will be shared. A disjoint node partition, one the other hand, uniquely maps every node to a single processor, and hence some elements will be split among processors. Typically, by adopting an element partitioning, information related to the common nodes needs to be communicated, while in the nodal approach information related to nodes on split elements needs to be exchanged. Both models can be implemented within the BSP framework; however, an element partitioning would require extra buffering and/or more synchronisation points during the solution phase. Consequently, the parallel versions of TOSCA, SCALA and ELEKTRA were built choosing to adopt a disjoint node partitioning of the Finite Element mesh. All the various iterative solvers used in such codes are variations of the basic conjugate gradient algorithm. The basic blocks appearing in the algorithm are:

1. vector **saxpy** operations (i.e. $y := y + \alpha x$);
2. preconditioning operations;
3. inner product / norm evaluations;
4. sparse matrix-vector multiplications;

The unique mapping of the d.o.f. to processors enables vector **saxpy** to be performed entirely in parallel. The same is true for preconditioning, if a block diagonal form of the preconditioner is adopted; in the current implementation of TOSCA and ELEKTRA, the existing preconditioners (DSLU, IC, ILU) were restricted to their block diagonal counterparts by ignoring terms related to non-owned degrees of freedom. Inner products / norm evaluation require a reduction operation, i.e. only local contribution for the owned d.o.f.s are computed in parallel, and the final result is computed at the end, by adding up all the partial contributions; the final result is then duplicated among all the local processors. A more complex phase is the sparse matrix-vector multiplication: we can explain what happens with a simple example (see Fig. 1): we assume to have only one d.o.f. per node (scalar potential case), the problem has 10 nodes and 4 elements. We adopt a node partition, assigning to PROC1 the nodes 1, 2, 3, 4, 5, 6 (and elements (1,2,4,3), (3,4,6,5), (5,6,8,7), and to PROC2 the nodes 7,8,9,10 (and elements (5,6,8,7), (7,8,10,9). Note that element (5,6,8,7) is shared between the processors. If we have to evaluate a "shared" row of the product, for example row 6, we must calculate something like the following equation:

$$u_6 = k_{63}p_3 + k_{64}p_4 + k_{65}p_5 + k_{66}p_6 + k_{67}p_7 + k_{68}p_8 \tag{10}$$

Although node 5 resides on PROC1, we cannot compute u_6, unless p_7 and p_8 (residing on PROC2) are available to PROC1. Nodes 7 and 8 are said to be "halo nodes" for the PROC1. Similarly, nodes 5 and 6 are said to be "halo nodes" for the PROC2.

The "halo nodes" must be copied into a local address space before executing an operation like the product matrix-vector. Clearly an efficient implementation must keep the number of "halo nodes" as low as possible and balanced over the

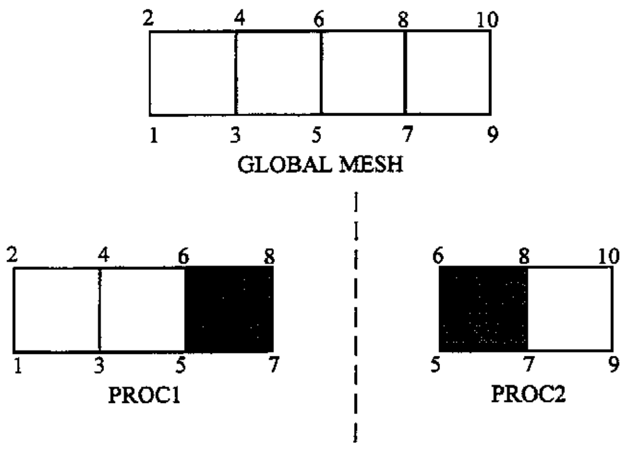

Fig. 1. Example of a nodal partitioning of the mesh; the "halo nodes" belong to the dark element

processors; this consideration is intimately linked with the definition of an adequate parallel mesh partition. For this scope two techniques were tested: the Recursive Spatial Bisection (RSB) [6], and the Recursive Moment of Inertia (RMI) [7]; the latter was adopted, as the first was computationally too expensive. The RMI is based on the minimisation of the halo sizes obtained partitioning the mesh cutting it across the principal axes of inertia. The method therefore attempts to define thin elongated subdomains within the mesh. To reach a reasonable load balancing for the most critical phases of the computation (that in this case were identified to be the matrix-vector product and the preconditioning), the procedure has been modified so as to generate partitions with the same number of nonzero entries per subdomain. This is achieved assigning to each node as "weight" the number of nonzero entries it contributes to the stiffness matrix. Examining an ordered list of nodes together with the weights list, it is possible to define balanced groups of nodes. Note that in this way other phases, like the stiffness matrix assembly, and other vector operations may not be perfectly balanced, but this is not a major problem, as the computationally most intensive operations are optimised.

V. Coil Calculation Phase

If the source of the magnetic field to be modelled is a system of coils, the formulations used by TOSCA and ELEKTRA require the evaluation of the magnetic field due to them, according to Biot-Savart law (2), and the evaluation of additional right hand side terms. The cost of this calculation may sometimes largely dominate the total computation time. Even though most of this computation is perfectly data parallel, it is essential for efficient calculation that an adaptive quadrature method is used, because the accuracy, and therefore the computation time, depend strongly on how close the field point is to the coil. The cost per integral is thus clearly highly irregular with respect to the position if a purely geometric data decomposition is adopted, resulting in grossly unbalanced computational loads on the different processors. A different data decomposition technique had to be adopted, based on a cyclic distribution of these integrals across processors. In this way the local cost continuity property can be easily exploited. This strategy can be implemented in a straightforward way within the BSP/SPMD approach adopted within the project. An advantage of this static approach to a dynamically varying workload is that the data can be distributed in large units, maximising communication efficiency. The R.H.S. computational costs are dramatically reduced following the technique just explained in problems with a large number of conductors.

Moreover, on the interface between total and reduced scalar potential regions, certain line integrals of the source magnetic fields have to be evaluated. They take the following form:

$$\mathbf{I} = \int_{e} \mathbf{H}_s \cdot d\mathbf{e} \qquad (11)$$

where **e** is the path along an element edge. This element edges are found by searching the element database (on the local hard disk unit) for nodes on the interface between the different scalar potential, buffering them on the processor 0, and distributing them across the processor network. The edge search procedure is based on the construction of a tree connecting the nodes belonging to the total/reduced potentials, and the assignment of "potential jumps" to the specific nodes using the information contained in the tree. The tree formation may need the database to be re-read several times; however this process can be optimised reordering the disk data organisation. The final contribution to the R.H.S. terms comes from the calculation of surface integrals over the interface total/reduced potential faces. They take the following form:

$$\mathbf{I} = \int_{S} N_i (\mathbf{H}_C \cdot \mathbf{n})\, ds \qquad (12)$$

where **n** is the normal to the element facet. As before, the computation of this term requires a search on the hard disk unit, the buffering of the faces on the processor 0, and their subsequent distribution across the processor network. Only one complete read of the database is required here, so this phase is much faster than the calculation over the edges.

VI. Beam Calculation Phase

The particle beam modelling handled by SCALA can be of interest for the design of cathode ray tubes and other devices requiring the calculation of electron beam trajectories, as well as the computation of the fields. Steady state solutions can be calculated by tracking the electron beam through the device; the modelling is performed distributing the total charge into a number of individual rays, each of which is then tracked integrating the equation of motion along the path followed by the particle by making use of the field computed with the Finite Element method.

This should be a perfectly data parallel operation, since rays do not interact within each iteration; however, also in this calculation, though element based, the cost is again distributed in a highly non-uniform way, because the beams are very localised, and the tracking makes use of an adaptive integration procedure. The approach was therefore to adopt a cyclic distribution method, applying it to the individual particle beams themselves. In this way an effective load balancing among processors can be obtained, and the parallelisation becomes very efficient, particularly with a high number of rays.

VII. Results of Some Test Cases

A number of test cases of different types have been selected within the project to test the parallelised versions of the codes. All tests have been chosen among real design cases, also trying to define "difficult" situations in order to provide a serious evaluation of parallelisation effectiveness in an industrial environment for the different codes, in various possible analysis situations.

Tests have been performed on a number of parallel architectures located at different european sites and made accessible to project partners, via network, by the Europort 2 supervisor, Smith. Furthermore, additional machines have been made available by Oxford Parallel. The parallel machines that have been used, to various extents, within the project to run test cases were the Meiko CS2, the IBM SP2, the Silicon Graphics Power Challenge and a DEC Alpha network.

Because of space reasons, it is not possible to report here on the whole set of results obtained on the different parallel architectures tested. In this paper attention will be focused on some of the results obtained by the test cases defined and run at the Department of Electrical Engineering of the University of Genova, code named PERMRI and SUPERMAG.

A. PERMRI

PERMRI is a nonlinear magnetostatic test case, modelling a permanent magnet structure for Magnetic Resonance Imaging (MRI), shown in Fig. 2.

The number of nodes must be high, since the most important requirement here is field uniformity. In this case there are no coils: the field sources are a number of permanent magnets inserted into the magnetic structure. Four test runs, I to IV, with different numbers of unknowns were made (4,000, 12,000, 30,000 and 45,000 respectively). The results for the largest case on the Meiko CS2 and on the IBM SP2 are reported in Tab.3 and 4, showing both solver and total performance.

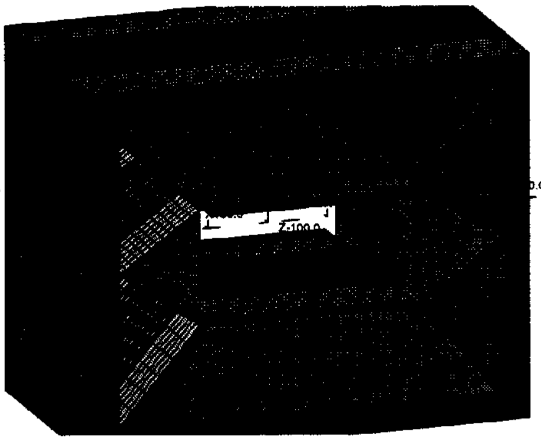

Fig. 2. Geometry of the PERMRI test case

It should be remarked here that all time figures are expressed as *elapsed*, and not CPU time. This is clearly due to the fact that the speedup interesting from the user's viewpoint is the elapsed time speedup.

TABLE 3
SUMMARY OF RESULTS FOR PERMRI TEST CASE ON THE MEIKO CS2

Test run IV (about 45,000 unknowns)					
Proc.	Iter. (min max)	Solver time	Solver speedup	Total time	Total speedup
scalar	122	1196 s	-	1915 s	-
	142				
1	122	1210 s	0.99	1841 s	0.96
	142				
2	133	655 s	1.83	1179 s	1.62
	152				
4	141	338 s	3.54	735 s	2.6
	162				
8	150	179 s	6.68	540 s	3.55
	174				

TABLE 4
SUMMARY OF RESULTS FOR PERMRI TEST CASE ON THE IBM SP2

Test run IV (about 45,000 unknowns)					
Proc.	Iter. (min max)	Solver time	Solver speedup	Total time	Total speedup
scalar	122	308 s	-	529 s	-
	142				
1	122	344 s	0.90	634 s	0.83
	142				
2	133	187 s	1.65	386 s	1.37
	152				
4	141	105 s	2.93	245 s	2.16
	162				
8	150	66 s	4.67	178 s	2.97
	174				
16	157	44 s	7.00	159 s	3.32
	180				

B. SUPERMAG

SUPERMAG is a quasi-static, linear, transient eddy current problem that models the development of eddy currents into an aluminium cryostat, designed to contain superconducting coils for M.H.D. generation, during a fast transient. The scope of the modelling is to evaluate the dissipation due to eddy currents generated by an exponential decay of the main field.

The computation is extremely heavy, because the transient is handled with time discretisation and a solution of a system of algebraic equations must be performed at each time step, describing with acceptable accuracy the thin aluminum layer, much smaller than the coils.

Four test runs, I to IV, were performed also for this problem with about 6,000, 12,000, 18,000 and 36,000 unknowns respectively. The geometry of the system is shown in Fig. 4.

Table 5 and 6 show the results for the largest run on the Meiko CS2 and on the IBM SP2 machines, for the different numbers of processors used, with the same structure of the previous test case.

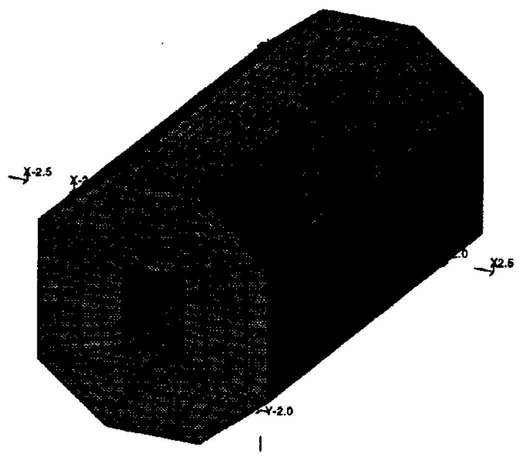

Fig. 3. Geometry of the SUPERMAG test case

TABLE 5
SUMMARY OF RESULTS FOR SUPERMAG TEST CASE ON THE MEIKO CS2

Test run IV (about 36,000 unknowns)					
Proc.	Iter. (min max)	Solver time	Solver speedup	Total time	Total speedup
scalar	103 133	34918 s	-	35812 s	-
1	103 133	36372 s	0.96	37180 s	0.96
2	155 204	29328 s	1.19	29968 s	1.20
4	170 205	18910 s	1.85	19488 s	1.84
8	174 232	13049 s	2.68	13615 s	2.63

TABLE 6
SUMMARY OF RESULTS FOR SUPERMAG TESTCASE ON THE IBM SP2

Test run IV (about 36,000 unknowns)					
Proc.	Iter. (min max)	Solver time	Solver speedup	Total time	Total speedup
scalar	102 133	9381 s	-	10492 s	-
1	102 133	11335 s	0.87	11967 s	0.88
2	153 201	9134 s	1.08	9512 s	1.10
4	161 209	5821 s	1.69	6070 s	1.73
8	162 222	4250 s	2.31	4437 s	2.36
16	172 235	3573 s	2.75	3737 s	2.81

VIII. CONCLUSIONS

The parallelisation of industrial electromagnetic CAD software performed in the PARTEL project of Europort 2 gave positive results in terms of ability to modify existing analysis codes to run on parallel machines.

Making use of the BSP model, the overall modifications to the codes were kept to a minimum, and the parallelisation was implemented in a rather straightforward way. More than one type of parallel decomposition had to be used, due to the very different requirements in the solution procedures of the codes. This was simple to implement with the approach selected, since the BSP approach allows to clearly distinguish among the parallel phases; results were good on every architecture examined. The block preconditioner behaviour was surprisingly good for all test cases, giving rise to a modest increase in the number of iterations in all cases tested. The use of the BSP model also allowed to analyse the computational cost on different platforms.

In terms of overall speedup for design test cases, always evaluated in terms of elapsed time and with some tests deliberately chosen among "difficult" ones to maximise the reliability of results from a practical standpoint, it was shown that results can be notably different as a function of the computational weight of the solver or other parallelisable phases in the specific problems being solved.

For the test cases reported here, all of the class of the "difficult" ones, speedups have proven to be fair but not too impressive, but with increasing significance as problem size increases. Better results have been obtained in other tests, particularly in the SCALA runs, since the highly parallel beam computation phase provides a nearly linear speedup with the number of processors in large problems.

In general, it can be concluded that parallelisation of existing electromagnetic Finite Element codes has proven viable with acceptable efforts using the BSP approach, and that it can provide computational advantages, that, though function of a number of different conditions, are certainly of increasing interest for the largest problem sizes and for the extremely demanding computational loads increasingly required by the current fast development of automatic design optimisation.

REFERENCES

[1] L. G. Valiant, "A bridging model for parallel computation", *Comm. ACM*, 33(8), pp. 103-111, August 1990.

[2] J. M. D. Hill, W. F. McColl, "An initial proposal for the BSP Worldwide standard library", *Technical Report,* Oxford University Computing Laboratory, January 1996.

[3] J. Simkin, C. W. Trowbridge, "Three dimensional non-linear electromagnetic field computations using scalar potentials", *IEE Proc.*, vol. 127, n.6, 1990.

[4] P. Girdinio, M. Repetto, J. Simkin, "Finite Element Modelling of Charged Beams", *IEEE Trans. on Mag.*, vol. MAG-30, n.5, p.2932-2935, September 1994.

[5] C. R. I. Emson, C. F. Bryant, C. W. Trowbridge, "A General Purpose 3-D Formulation for Eddy Currents Using the Lorentz Gauge", *IEEE Trans. on Mag.*, vol. MAG-26, p.2373, September 1990.

[6] H. D. Simon, "Partitioning of Unstructured Problems for Parallel Processing", *Comput. Systems Engnrg.*, 2(2/3), pp. 135-148, 1991.

[7] C. Farhat, M. Lesoinne, "Automatic Partitioning of Unstructured Meshes for the Parallel Solution of Problems in Computational Mechanics", *IJNME*, **36**, pp. 745-764, 1993.