Volume Integral Equations in Nonlinear 3D Magnetostatics

Lauri Kettunen and **Kimmo Forsman**

Tampere University of Technology
Laboratory of Electricity and Magnetism, P.O. Box 692
FIN-33101 Tampere, Finland

David Levine and **William Gropp**

Argonne National Laboratory
Mathematics and Computer Science Division
9700 South Cass Avenue
Argonne, Illinois 60439, U.S.A.

SUMMARY

In this paper a discussion of volume integral formulations in three-dimensional nonlinear magnetostatics is presented. Integral formulations are examined in connection with Whitney's elements in order to find new approaches. A numerical algorithm based on an h-formulation is introduced. Results of demanding application problems are shown demonstrating the characteristics of this kind of volume integral approach. In addition, a discussion of the parallelized version of the numerical code based on the h-type approach is presented, appended with numerical results illustrating the advantages of combining integral formulations with concurrent computing.

INTRODUCTION

A numerical approximation for static or low-frequency electromagnetic problems can be computed with either partial differential equations (PDEs) or integral equations. Partial differential equations are often favorable because they offer cost-effective solutions for three-dimensional problems. Integral equations have their own advantages; for example, air regions can be excluded, and the exterior boundary condition (i.e., that the magnetic field vanishes in infinity) is satisfied automatically (see, e.g., [1], [2]). Moreover, in linear problems, boundary integral equations give often a reasonably accurate solution with a relative small number of unknowns.

During the past twenty years PDEs have dominated research in the field of numerical computation of low-frequency electromagnetic fields, and they have almost overwhelmed the developments in integral formulations. Probably one of the main reasons that integral equations have not attracted researchers as much as PDEs is the dense integral equation matrix. It is well known that iterative solvers such as ICCG (incomplete Choleski factorization, conjugate gradient) are effective in solving a sparse systems of equations [3]. On the other hand, it is not clear how a full system of equations should be effectively solved to avoid a total execution count of $O(n^3)$ with direct methods like LU factorization. PDEs do not lead to the same problem.

Some groups have developed so-called hybrid formulations in order to combine the desirable properties of PDEs and boundary integral equations. This kind of system produces a matrix that has a dense block whose size is related to the number of nodes on the exterior boundary; the rest of the matrix is sparse. Hybrid formulations retain the advantages of having to discretize only nonair (i.e., magnetic and conducting) regions. A difficulty in hybrid formulations, however, is the size of the dense submatrix: if it becomes large enough, the sparsity property is lost. Also, in a parallel computing environment, the equation matrix should be carefully split into submatrices in order to balance the load between processors. Nevertheless, hybrid formulations are a competitive alternative to volume integral equations in solving nonlinear problems, having many of the properties of integral equations but having a less dense, or even a sparse, matrix from the numerical point of view.

The purpose of this paper is to study the use of integral equations on a workstation and a high-performance computing environment in order to solve accurately three-dimensional (3D) nonlinear magnetostatic problems. Our final goal is time-dependent problems; hence,

we have developed and implemented an h-type formulation, which can be generalized to low-frequency eddy current problems with magnetic materials.

BACKGROUND

One of the first volume integral equation approaches for nonlinear magnetostatics was the GFUN code, developed in early 1970s [4, 5, 6, 7, 8, 9]. GFUN is based on a piecewise constant vector approximation of magnetic field strength H inside elements. Components of H are solved independently within each element; this approach implies that H does not have any kind of continuity between neighboring elements. As a result, the generation of the system of equations is simple, and there is also some flexibility in the mesh generation. For instance, the tesselation by elements does not need to be "finite elements": two distinct elements do not have to share a face, an edge, or a vertex.

The GFUN code was shown to be successful in solving practical problems, but it also led to some difficulties. The number of unknowns in the system of equations is $n = n_{elements} \times 2$ or $n = n_{elements} \times 3$ for the 2D and 3D case, respectively. This means that in refining the mesh, the solution time of the integral equation system increases rapidly. In addition, if susceptibility is large enough, the system matrix becomes illconditioned and causes a "looping pattern" [8] in the H-field. Results in air are still reasonable, but the H-field within magnetic parts is useless.

A dense system matrix is inherent to integral equations and thus unavoidable, but the size of the matrix can be dramatically decreased by approximating field variables in standard finite element spaces. The looping problem is also avoided by choosing a type of element that imposes proper continuity properties. The benefits of a proper choice of basis functions were noticed by Iselin in 1976 [10] and Pasciak in 1983 [11]. They both introduced a volume integral formulation based on scalar potentials and "nodal elements." Recently, Lin Han et al. also published a scalar potential formulation [12].

A good understanding of different formulations and possibilities can be achieved by examining Whitney elements and integral equations together. Whitney elements are a class of finite elements (named after English mathematician H. Whitney [13].) Whitney elements were introduced by Bossavit in connection with computational electromagnetics [14, 15, 16, 17]. During the past five years, Whitney elements have become popular and widely used by scientists and engineers in the numerical analysis of electromagnetic fields. Whitney elements offer a natural basis for imposing correct physical continuity properties on electromagnetic fields. They differ from traditional elements in the sense that the degrees of freedom are related to all kinds of simplices in a simplicial mesh, that is, to edges and to facets as well to nodes. Depending on which simplex the degrees of freedom are related to, these elements are often called nodal, edge, facet, or volume elements.

A integral formulation based on edge elements was introduced by Albanese and Rubinacci, who developed a formulation for eddy current problems with nonmagnetic materials [18], [19]. The formulation presented in this paper complements Albanese and Rubinacci's approach in the sense that a combination of these two formulations is capable of solving eddy current problems with magnetic materials.

FINITE ELEMENT SPACES

In this section we summarize the background of Whitney elements, based on the articles published by Bossavit [14, 15, 16, 17, 20].

Let us take a bounded region of space V. The surface of V is S. Region V is split into a finite number of tetrahedra such that the tesselation satisfies the standard properties of finite element meshes: two distinct tetrahedra share nothing, a node, a proper edge, or a proper facet. It is also assumed that curved surfaces are approximated with straight-sided tetrahedra. The sets of nodes, edges, facets, and tetrahedra are denoted N, E, F, and T, respectively. All nodes are numbered, and a node connectivity list is formed for the edges, facets, and tetrahedra: node $n = \{i\}$, edge $e = \{i, j\}$, facet $f = \{i, j, k\}$, and tetrahedron $t = \{i, j, k, l\}$. Coordinates of node $n = \{i\}$ are denoted with r_i.

Our purpose is to consider functions and vector fields constructed with Whitney elements. Having a set of degrees of freedom and basis functions, the functions and vector fields are linear combinations

$$g \;=\; \sum_i g_i \; w_i \;. \tag{1}$$

Finite element spaces W^p spanned by the basis functions are generated by elements of degree p associated with p-simplices, $p = 0, 1, 2, 3$ (for nodes, edges, facets, and tetrahedra, respectively). The barycentric function λ_i is a piecewise linear continuous function in V, which equals 1 in r_i, is positive in tetrahedra sharing $n = \{i\}$, and equals 0 inside tetrahedra that do not contain vertex $n = \{i\}$. The basis functions that span spaces W^0, W^1, and W^2 are

$$w_n \;=\; \lambda_i, \quad n = \{i\}, \tag{2}$$

$$w_e \;=\; \lambda_i \nabla\lambda_j - \lambda_j \nabla\lambda_i, \quad e = \{i, j\}, \tag{3}$$

$$w_f \;=\; 2(\lambda_i \nabla\lambda_j \times \nabla\lambda_k + \lambda_j \nabla\lambda_k \times \nabla\lambda_l + \lambda_k \nabla\lambda_i \times \nabla\lambda_j), \quad f = \{i, j, k\}. \tag{4}$$

As can be seen, the basis function w_n of node n is equivalent to that of a classical Lagrange element of order 1. It can be shown that a scalar field in W^0 is continuous, a vector field in W^1 is tangentially continuous, and a vector field in W^2 has normal continuity.

In constructing g's, the degrees of freedom are $g_i \;=\; g(r_i)$, $g_e = \int_e g$, and $g_f = \int_f g$ (i.e., g at r_i, circulation of g along edge e, flux of g across facet f), for $p = 0, 1, 2$, respectively.

Spaces W^0, W^1, W^2, and W^3 are related to each other such that

$$grad\; W^0 \;\subset\; W^1, \;\; curl\; W^1 \;\subset\; W^2, \;\; div\; W^2 \;\subset\; W^3 \;. \tag{5}$$

In other words, grad w_n and curl w_e are a linear combination of some w_e's and w_f's, respectively. Once the simplices are numbered, one can form incidence matrices $\mathbf{G}$, $\mathbf{C}$, and $\mathbf{D}$, which represent the incidence relations between the simplices. For example, the size of $\mathbf{G}$ is $n_{egdes} \times n_{nodes}$, and all the entries of $\mathbf{G}$ are -1, 0, or 1. If V and S are simply connected, we can write

$$grad\; \psi \;\;=\;\; \sum \{(\mathbf{G}\; \bar{\psi})_e w_e \;\;\mid\;\; e \in E \;\} \;. \tag{6}$$

A property between W^0 and W^1, which is useful in developing integral formulations, can now be easily verified: the kernel of *curl* in W^1 is exactly *grad* W^0. Similarly, the kernel of *div* in W^2 is precisely *curl* W^1. These injective properties are shown with a sequence

$$W^0 \xrightarrow{grad} W^1 \xrightarrow{curl} W^2 \xrightarrow{div} W^3 .$$

Similarly for the vector spaces $\mathbf{W}^p$, $p = 0, 1, 2, 3$, spanned by the vectors of the degrees of freedom, one can form a sequence (assuming simply connected V and S)

$$\mathbf{W}^0 \xrightarrow{\mathbf{G}} \mathbf{W}^1 \xrightarrow{\mathbf{C}} \mathbf{W}^2 \xrightarrow{\mathbf{D}} \mathbf{W}^3 .$$

TREE–CO-TREE SEPARATION

Heretofore we have considered how one Whitney space can be mapped into a kernel of another one (rightmost to it in the sequence). For integral equations, we wish also to know the inverse: we wish to represent a gradient, curl, or div field using edge, facet, or volume elements, respectively. For instance, since the kernel of *curl* in W^1 is exactly *grad* W^0, there exists a set of degrees of freedom that represents a gradient field in W^1 (assuming simply connected V). Similarly there is a set of degrees of freedom representing a curl field in W^2. From a practical point of view, there is a problem in selecting an independent set of degrees of freedom. This problem was indirectly solved by Albanese and Rubinacci [18, 19]. They implied a "two-component gauge" [21]

$$T \cdot u = 0 \tag{7}$$

for vector potential T by forming a co-tree from the graph of all edges in the mesh and disregarding the degrees of freedom associated with the tree edges. The purpose of a gauge is to remove the choice of an arbitrary gradient field related to a vector potential; that is, a gradient field can be added to the vector potential without altering the curl of it. In Albanese and Rubinacci's case, the number of unknowns was decreased by the number of tree edges, to remove the arbitrary choice of a gradient field.

Let us denote the number of edges with n_{edges} and the number of edges in a tree with n_{tree}. By definition of a tree, tree edges connect all the nodes of a mesh without forming any closed loops. Thus, by interpeting the edge-circulations along tree edges as differences in scalar potential, (in a simply connected region) a scalar potential field in W^0 can be formed up to a constant. The gradient of this scalar field is in the kernel of *curl* in W^1. Thus, if we wish to approximate a gradient field in W^1, the number of independent unknowns equals the number of tree edges in the mesh. All the edge-circulations along co-tree edges can be defined with the aid of tree edges, since the sum of the degrees of freedom around any closed path formed by a co-tree edge and the corresponding set of tree edges must be zero (Fig. 1). Thus, after a tree is formed and all the edges are numbered, a rectangular incidence matrix $\mathbf{R}$, entries of which are all -1, 0, or 1, can be

formed (the size of $\mathbf{R}$ is $n_{edges} \times n_{tree}$), and the degrees of freedom $\mathbf{h}$ can be calculated with the coefficients $\mathbf{h}^t$ of tree edges:

$$\mathbf{h} = \mathbf{R}\mathbf{h}^t. \tag{8}$$

In the same manner a curl field can be represented, with the degrees of freedom associated with a set of facets that does not possess any closed volumes. (The sum of the degrees of freedom around any closed surface must be zero.) An incidence matrix $\mathbf{S}$ similar to $\mathbf{R}$ can be formed, and all the degrees of freedom $\mathbf{b}$ are found from a reduced set of coefficients $\mathbf{b}^t$:

$$\mathbf{b} = \mathbf{S}\mathbf{b}^t. \tag{9}$$

A gradient field h in W^1 is thus

$$h = \sum_{e=1}^{n_{edges}} h_e w_e = \sum_{e=1}^{n_{edges}} (\mathbf{R}\mathbf{h}^t)_e w_e = \sum_{i=1}^{n_{tree}} h_i^t (\sum_{e=1}^{n_{edges}} w_e R_{e,i}) = \sum_{i=1}^{n_{tree}} h_i^t v_i \ . \tag{10}$$

The new basis functions v associated with the tree edges are hence linear combinations of w_e's. Similarly, we find basis functions for a curl field b in $\mathbf{W}^2$. If we form a set of facets that does not possess closed volumes, we can define new basis functions v, which are linear combinations of w_f's:

$$b = \sum_f b_f w_f = \sum_j b_j^t v_j \ . \tag{11}$$

Hence we have three alternatives for expressing a gradient field $h = grad\ \phi$ or a curl field $b = curl\ a$:

$$h = \sum_n \phi_n\ grad\ w_n = \sum_e h_e w_e = \sum_e h_e^t v_e \ , \tag{12}$$

$$b = \sum_e a_e\ curl\ w_e = \sum_f b_f w_f = \sum_f b_f^t v_f \ . \tag{13}$$

INTEGRAL FORMULATIONS AND VARIOUS ALTERNATIVES

Let us denote magnetic flux density with B, magnetic field strength with H, and magnetization with M. A physical description of magnetostatic fields can be developed using the idea of superposition of fields from current and magnetization sources. Let B^s and H^s be B and H from current sources in the absence of magnetic materials, respectively. Fields caused by magnetization are denoted with B^m and H^m. The total B and H fields are hence

$$B = B^m + B^s \tag{14}$$

and

$$H = H^m + H^s \ . \tag{15}$$

Source fields B^s and H^s generated by currents J are constructed as follows:

$$A^s(r) = \frac{\mu_0}{4\pi} \int_{V^s} \frac{J(r')}{|r - r'|} dv' \ , \tag{16}$$

$B^s = curl\ A^s$, and $H^s = \frac{1}{\mu_0} B^s$. V^s is the region where $J(r) \neq 0$. Fields B^m and H^m are given by

$$A^m(r) = \frac{\mu_0}{4\pi} \int\limits_{V^m} \frac{M(r') \times (r - r')}{|r - r'|^3} dv' \ , \tag{17}$$

($\times$ is a cross product) $B^m = curl\ A^m$, and

$$H^m(r) = \frac{1}{\mu_0} B^m - M = grad \left[\frac{-1}{4\pi} \int\limits_{V^m} \frac{M(r') \cdot (r - r')}{|r - r'|^3} dv' \right] = \ -grad\ \phi. \tag{18}$$

The subregion of V, where $\chi \neq 0$, is denoted by V^m. Assuming isotropic materials, H and B are related to magnetization M such that

$$M = \chi(|H|)H \tag{19}$$

and

$$M = (\frac{1}{\mu_0} - \frac{1}{\mu(|B|)})B. \tag{20}$$

Let us assume that V^m and S^m are simply connected. B and H are approximated in W^2 and W^1, respectively. Multiplying Maxwell's equations

$$\nabla \times H = J \tag{21}$$

and

$$\nabla \cdot B = 0, \tag{22}$$

with appropriate test functions a' and ψ', respectively, and applying integral relationships (i.e., theorems analogous to Green's first identity), we get

$$\int\limits_{S^m} (a' \times \frac{1}{\mu}\, curl\, A) \cdot n - \int\limits_{V^m} curl\, a' \cdot \frac{1}{\mu}\, curl\, A^m = \int\limits_{V^m} curl\, a' \cdot \frac{1}{\mu}\, B^s \tag{23}$$

and

$$\int\limits_{S^m} \mu\psi' \frac{\partial \psi}{dn} + \int\limits_{V^m} grad\, \psi' \cdot \mu\ grad\ \phi = \int\limits_{V^m} grad\, \psi' \cdot \mu H^s \ . \tag{24}$$

From (23) and (24) we get the following variational forms:

$$\int\limits_{V^m} \frac{1}{\mu}\, B \cdot b' - \int\limits_{V^m} \frac{1}{\mu}\, B^m \cdot b' = \int\limits_{V^m} \frac{1}{\mu}\, B^s \cdot b', \quad \forall b' \in W^2 \cap\ ker(div) \tag{25}$$

and

$$\int\limits_{V^m} \mu H \cdot h' - \int\limits_{V^m} \mu H^m \cdot h' = \int\limits_{V^m} \mu H^s \cdot h', \quad \forall h' \in W^1 \cap\ ker(curl). \tag{26}$$

In Equations (23)–(26), any (simply connected) region where $J = 0$ can be selected instead of V^m. We call (25) and (26) b- and h-type volume integral formulations. As

expected, the background of the integral equation approach is similar to that of PDEs and hybrid formulations. For instance, (26) can be interpreted as arising from the same ground as the PDE scalar potential formulations [22] or the h-type hybrid formulations [17], [23]. Similarly, the weighted residual form of GFUN by Simkin [24] can now be understood to be related to Equations (22), (24), and (26).

We have now several alternatives for generating a numerical algorithm. We can either use the b-formulation and solve $\mathbf{a}$ or $\mathbf{b^t}$ from (13) and (25), or use the h-formulation (26) and solve $\bar{\phi}$ or $\mathbf{h^t}$ from (12) and (26). If the b-formulation is used and $\mathbf{a}$ is selected, a gauge $A \cdot u$ similar to Albanese and Rubinacci's gauge has to be added in order to achieve a unique solution [25].

Theoretically, assuming exact arithmetic, the solution is the same whether $\mathbf{a}$ or $\mathbf{b}^t$ is solved. The same holds also for $\bar{\phi}$ and $\mathbf{h}^t$. From the numerical point of view, however, there are some differences. If $\mathbf{b}^t$ or $\mathbf{h}^t$ is solved, one has to form a set of facets or edges that does not possess closed volumes and loops, respectively. In addition, one has to form a connectivity list, including data on how the coefficients of the rest of the facets or edges can be expressed in terms of the coefficients of this independent set. This requires extra operations generating the integral equation matrix, compared with the case if $\mathbf{a}$ or $\bar{\phi}$ is chosen to be solved.

Another aspect of the comparison of the formulations is the possibility of generalization to a larger class of problems. If $J(r) \neq 0$ in V^m, H fails to be a gradient field, and then it is sensible to choose $\mathbf{h}$ to be solved. The most interesting class of problem where $J(r) \neq 0$ in V^m is the low-frequency (i.e., eddy current) problems. If $\mathbf{h}^t$ is chosen as a solution variable in the static case, the formulation can be combined with the eddy current formulation by Albanese and Rubinacci [18],[19] for time-dependent problems. In the static limit, the solution variable is $\mathbf{h}^t$. In the time-dependent case, if the system has no magnetic parts, then $\mathbf{t}$ is solved (i.e., circulations of electric vector potential T along the co-tree edges); if there are magnetic and conducting objects, the solution variable is $\mathbf{h}$ (along all the edges).

The choice between a b- or an h-formulation depends on the type of problem, and it is difficult to predict which one gives more accurate results with the same amount of work. In fact, having both a b- and an h-type of solution would be optimal, in the sense that with both solutions one has an indicator of how the mesh should be refined.

At this point we have chosen the h-formulation of Equation (26) with $\mathbf{h}^t$ as a solution variable for implementation. We have some practical reasons for this choice. First, as already mentioned, this approach has a natural extension to eddy current problems. Second, the number of unknowns is n-1 for each distinct region, where n is the number of nodes in the region. A b-formulation typically has far more unknowns. (The ratio of the number of tree edges and co-tree edges is 1:6 for a regular infinite mesh.) Finally, the amount of temporary data needed to store during the iteration of nonlinear solution is smaller with the h-formulation. In the generation of the integral equation matrix one has to form data that depends only on the geometry. In order to avoid regeneration of this information, the data is stored on a temporary file. In the case of (26), we have to store

($n_{elements} \times n_{tree} \times 3$) real numbers (in either single or double precision). A b-formulation requires ($n_{elements} \times n_{cotree} \times 3$) numbers to be stored.

NUMERICAL IMPLEMENTATION

In V^m, H is a gradient field and we need to solve only the degrees of freedom $\mathbf{h}_e^t$ related to the tree edges. Thus H is given by

$$H = \sum_e \mathbf{h}_e^t v_e \ . \tag{27}$$

The integral equation matrix $\mathbf{L}$ is the sum of two matrices $\mathbf{H}$ and $\mathbf{H}^m$. Element $\{i,j\}$ of matrix $\mathbf{H}$ is

$$H_{i,j} = \int_{V^m} \mu v_i \cdot v_j . \tag{28}$$

The entries of $\mathbf{H}^m$ are

$$H^m_{i,j} = -\int_{V^m}\int_{V^m} \mu(r)\chi(r')\Big[\frac{v_i(r)\cdot v_j(r')}{|r-r'|^3} - \frac{3(v_i(r)\cdot(r-r'))(v_j(r')\cdot(r-r'))}{|r-r'|^5}\Big] dv dv' \ . \tag{29}$$

An alternative approach of forming matrix $\mathbf{H}^m$ is to approximate H^m in W^1, that is, in the same finite element space as H. In this case element $\{i,j\}$ of $\mathbf{H}^m$ is defined by

$$H^m_{i,j} = -\int_{V^m} \mu(r) v_i(r) \cdot \sum_e v_e \Big[\sum_{k=1}^{2} (-1)^k \int_{V^m} \frac{\chi(r')v_j(r')\cdot(r_k - r')}{|r_k - r'|^3} dv'\Big]_e dv \ . \tag{30}$$

The degrees of freedom associated with H^m are equivalent to the differences in scalar potential between the end nodes of edges. Therefore, Equation (30) includes an additional sum statement, which is denoted with index k. Here, $\{r_k \mid k = 1,2\}$ are the end nodes of edge e.

The system of equations to be solved is now

$$\mathbf{L}\mathbf{h}_e^t = (\mathbf{H} + \mathbf{H}^m)\mathbf{h}_e^t = \mathbf{f} \ , \tag{31}$$

where $\mathbf{f}$ contains the terms due to source currents:

$$f_i = \int_{V^m} \mu(\sum_e \mathbf{h}_e^s v_e) \cdot v_i \ . \tag{32}$$

The resulting integral equation matrix is asymmetric with both choices of matrix $\mathbf{H}^m$. In the case of (29) a symmetric matrix is, however, available by multiplying both sides of (26) with χ/μ.

Main Parts of the Software

We have implemented the h-formulation of (26) with both options of forming $\mathbf{H}^m$ (i.e., (29) and (30)). In addition, we have written a parallel version with $\mathbf{H}^m$ formed as in (30). The code we have developed is called GFUNET.

The main parts of the software are routines that find a tree, form the paths corresponding with co-tree edges, generate the integral equation and Jacobian matrix, compute the terms on the right-hand side, and solve the system of equations.

The data of a tree and paths corresponds with the nonzero entries of the incidence matrix $\mathbf{R}$ in (8). Earlier, we chose a tree rather arbitrarily, but recently we have changed the tree generation routines. A tree is spanned from a "root node," and we attempt to minimize the number of edges in paths (a path is a set of tree edges that connect the end nodes of a co-tree edge). The purpose of the minimization of the length of paths is related to iterative solvers. If the paths are short and the indices of edges properly set, the entries of the matrix with the largest absolute value come closer to the diagonal. Otherwise we have not encountered the choice of a tree to be significant. The CPU-time required to form a tree and the corresponding paths is usually fractions of seconds and in any case meaningless compared with the total execution time.

The terms on the right-hand side are integrated analytically or semi-analytically using Biot-Savart's law. To compute the circulation of H^s along edges, one may select Gaussian integration with a fixed number of integration points or an adaptive scheme.

The routines requiring the most CPU-time are the integral equation matrix generation and the solver. At first glance it may seem that the matrix generation becomes very complicated because the data of the tree and the corresponding paths has to be involved in the system of equations. In practice, however, the integral equation matrix can be built efficiently without allocating any extra memory.

To reduce the number of coefficients locally from six to three, we first define a local tree and a local basis for each tetrahedron [25], [26]. A tetrahedron has four nodes, and thus a local tree has three edges. Let us denote a tetrahedron with V^t. The local basis functions are constant vectors in V^t. Thus H is a constant vector and χ a scalar in V^t. Let us now consider (31) with $\mathbf{H}^m$ formed as in (30). We may rearrange terms of Eq. (30) such that

$$\int_{V^t} \frac{\chi(r')v(r')\cdot(r-r')}{|r-r'|^3}dv' = \chi(r')v(r')\cdot\int_{V^t}\frac{r-r'}{|r-r'|^3}dv'. \tag{33}$$

Excluding susceptibility, the rest of the terms on the right-hand side of (33) depend solely on the geometry of the problem. Computation of these terms requires a major part of the time needed to generate the integral equation matrix. Therefore these terms are computed only once and then stored in a file to be read during solving a nonlinear problem by iteration. This is, however, a critical part of the integral formulation. The size of these files is $(n_{elements} \times n_{tree} \times 3)$ entries, and they easily become very large. Unfortunately, this property is inherent in integral equations, since all the entities have a contribution to each other. (However, the GFUN code does not suffer from a similar problem; susceptibility can be removed by division from the entries of the matrix, because

of the lack of tangential continuity in H. On the other hand, the integral equation matrix of GFUN requires even more space than our temporary data storage files.)

In Equation (33), terms

$$v(r') \cdot \int_{V^t} \frac{r - r'}{|r - r'|^3} dv' \tag{34}$$

can be integrated analytically. In the software we have a flag that can be set to employ either fully analytic integration or numerical integration combined with analytic integration for the self-field terms. In our experience, it seems that from the practical point of view all the other terms except those for which $r \in V^t$ can be integrated numerically without losing accuracy.

If matrix $\mathbf{H}^m$ is chosen as in (30), it can be shown (Appendix A) that is not necessary to form the second volume integral at all. Assuming exact arithmetic, the solution is precisely the same whether or not the second volume integral is taken into account.

If $\mathbf{H}^m$ is formed according to (29), the first integral can be carried out analytically and the second numerically. The drawback of this alternative is the amount of space needed for temporary data storage. There is even more data to be stored than in the case of (30). (The amount of data depends on the number of integration points.) In addition, in our experience, results of this option are in all cases inaccurate compared with the results based on (30).

The system of equations resulting from (30) and (31) is nonsymmetric. We have used LU factorization with back substitution and the generalized minimal residual (GMRES) iterative solver with restart option to solve the system of equations. The efficiency of the GMRES solver depends on the preconditioner, on the number of cycles before restart, and on the initial guess. To achieve the full performance of a GMRES solver requires considerable testing, and we are still examining different options.

The nonlinear problem due to $\chi = \chi(|H|)$ is solved by iteration. An initial guess for χ is inserted, and typically we first iterate five to eight cycles with simple update of susceptibility before switching to Newton-Raphson iteration. In the generation of the Jacobian matrix, the precomputed geometry-dependent data is used. The time needed to generate the Jacobian matrix is typically only 1.25–2.0 times greater than the time needed to update the integral equation matrix.

The parallel version is based on Chameleon parallel programming tools [27], which provide a low-overhead interface to many vendors' message-passing libraries. Chameleon also supplies a uniform interface for program startup. In combination with message-passing packages such as p4 [28] or PVM [29], [30], the software needs no changes to run on a collection of workstations connected via Ethernet as well as on parallel supercomputers.

In the parallel version, each processor generates a rowwise decomposed block of the integral equation matrix (and the right-hand side) without any data broadcast. After the solution of the system of equations, the leading processor collects the solution vector and tests the convergence of the nonlinear iteration. If more cycles are needed, each processor

updates susceptibility data and its block of the matrix.

TEST RESULTS AND APPLICATIONS

In this section we give results demonstrating characteristics of the integral formulations. The first two examples are the international TEAM benchmark problems number 13 and 20 [31], [32]. The third example is a positron accumulator ring dipole magnet of the Advanced Photon Source (APS) of Argonne National Laboratory. For the first two examples, measured data is available. The results of the integral formulation of the last problem are compared with the solution computed by TOSCA [33], which is a commercial FEM software for nonlinear 3D magnetostatics.

The applications were chosen such that they also demonstrate the difficulties we have experienced. In addition, in the results shown in this paper no particular attempt has been made to minimize the computing time. The options of the code were those we have found reasonable in general. The results are reported as much as possible as if the software were used as an engineering tool. All the sequential results are computed on a DEC Alpha 3000-600 AXP and the parallel results on an IBM SP1 with RS/6000 model 370 processors.

TEAM Problem 13

TEAM problem 13 consists of thin steel plates, which are excited below the saturation level (Fig.) [31]. One of the main difficulties in the problem is a narrow air gap between the steel plates. The groups who have solved the problem agree that a high number of elements is required in order to achieve accurate results below the saturation level [34].

We have conducted many experiments with problem 13 and found that, even with a small number of elements, results are not unreasonable. This seems to be a characteristic feature of the h-type integral formulation. Thus we exploit the advantage of being able to compute reasonable results quickly with a small amount of elements by interpolating an initial guess from an existing solution. The number of cycles and the total computing time of a large problem are in many cases reduced by inserting a better initial guess than just a constant susceptibility in each material.

In our case, one-fourth of TEAM problem 13 has to be discretized. Results of the benchmark problem computed with three different discretizations are shown in Figs. 3 and 4. The number of nodes, elements, and equations with the charged CPU-times of the main parts of the sequential code are shown in Table 1.

We believe that TEAM problem 13 fits well with integral equations. Because only magnetic regions have to be discretized and the steel plates are thin, a relatively large number of elements can be concentrated close to the air gap and to the bend of the plates (Fig. 5). With PDEs such a refinement of tetrahedra would contain also numerous elements in the air close to the gap and the bend.

In this kind of problem consisting of thin plates, one may argue that a hybrid formulation is not, in principle, any better than a volume integral approach. The ratio between the interior nodes and the exterior nodes is very small; thus, from the numerical point of view,

Table 1: Timing of the main parts of GFUNET: TEAM problem 13 with three different meshes

Case	1	2	3
Number of nodes	184	803	3694
Number of elements	487	2503	14625
Number of equations	182	801	3692
Tree generation (s)	0.01	0.16	3.0
Path generation (s)	0.002	0.003	0.03
RHS generation (s)	7.6	30.9	141
Matrix generation (s)	5.2	100	2692
Matrix update (s)	0.08	2.7	157
Jacobian generation (s)	0.15	4.4	237
LU-solver (s)	0.09	3.9	348
Number of iterations	13	10	11
Total CPU-time (s)	16.0	207	8809

the sparsity property of the equation matrix of the hybrid formulation is lost. Therefore, in both cases, one has to solve a dense system of equations.

TEAM Problem 20

TEAM problem 20 (Fig. 6) offers a good test example for studying the accuracy of force calculations, because measured data is available [32]. As in problem 13, problem 20 includes narrow air gaps between two separate pieces of steel. The air gaps cause some difficulties in computing the force between the distinct parts, if integral equations are used.

First of all, if the force is computed with Maxwell's stress tensor, the solution is affected by large numerical errors in the magnetic field just outside the steel. The errors are due to the fact that the element or elements close to point in air where the field has to be computed dominate the solution. In the case of problem 20, one should have a very large number of elements close to the air gaps to avoid this problem. Compared with integral equations, PDEs have the benefit of offering easy and quick computation of the energy stored in the magnetic field. This allows the use of virtual work to estimate forces. With integral equations such an approach is not as practical, because the energy is not easily available. (Neither virtual work or Maxwell's stress tensor is optimal, however, because they are both known to be somewhat unstable in numerical computation.)

For these reasons we have looked at the option of interpreting magnetization with equivalent currents: $J^m = curl\ M$ and $K^m = M \times n$. Once the current distribution is found, the forces between magnetic parts can be integrated:

$$F = \int J^m \times B + \int K^m \times B. \tag{35}$$

Table 2: Forces of TEAM problem 20 with three different meshes

Case	1	2	3	Measured
Number of nodes	155	1013	2869	
Number of elements	366	3815	12562	
Number of equations	153	1011	2867	
Total CPU-time (s), 1000 AT	13.7	490.6	5590	
Total CPU-time (s), 3000 AT	10.3	278.5	3137	
Total CPU-time (s), 4500 AT	12.4	250.9	3185	
Total CPU-time (s), 5000 AT	12.3	262.1	3076	
Force F_z (N), 1000 AT	9.3	8.6	8.6	$8.1 \pm 4\%$
Force F_z (N), 3000 AT	62.3	57.3	57.7	$54.4 \pm 4\%$
Force F_z (N), 4500 AT	82.1	77.7	78.1	$75.0 \pm 4\%$
Force F_z (N), 5000 AT	87.2	82.9	83.4	$80.1 \pm 4\%$

In our case, J^m vanishes within the tetrahedra, and we have to deal only with surface current density K^m. This approach seems to be robust with our h-formulation. In addition, the use of equivalent currents is straightforward and does not require expertise in choosing integration surfaces as Maxwell's stress tensor does. In practice, we compute B analytically and integrate the cross product over facets numerically with Gaussian integration. The drawback of this approach is the time-consuming analytic integration of B. (This efficiency of force computations could be further improved by combining numerical and analytic integration.)

The forces computed with the sequential version are compared with the measured data in Table 2. (The reproducibility of the measurements is 4 % [35].) The accuracy of the computed forces of case 1 is noteworthy. Even when the charged CPU-time is less than 15 seconds, the accuracy is about 10%. (The charged CPU-time of the cases $I = 1000$ AT is bigger than the others because the geometric-dependent data has to be formed only once for each mesh.) On the other hand, with an increasing number of elements, convergence toward the measured values is slow. At this stage TEAM problem 20 has been available only for a short time, and it is too early to say how accurately the measured values will match results computed with various methods. (There is always the difficulty of defining accurately the BH-curve.) However, we have observed that the accuracy of the h-type integral formulation is sensitive to the distribution of elements. It is often quite difficult to predict how the mesh should be refined to increase overall accuracy. In our experience, each problem has some critical regions that must be properly discretized. Refining the mesh somewhere else seems to have only small, if any, effect on the accuracy of the solution.

APS PAR-Dipole Magnet

The last example is a dipole magnet from the Advanced Photon Source at Argonne National Laboratory. The poles are curved, and the magnet has shielding plates in front

of the coils. In addition, the ends of the poles are beveled (Figs. 7 and 8). Because the geometry is nontrivial, this problem is a challenging test for integral formulations. One difficulty with integral methods is the amount of memory needed for the matrix, which increases as $O(n^2)$, where n is the number of equations. Therefore, the maximum number of equations is fairly easily met (in a sequential computer), if geometrically complicated problems are solved.

The main difficulty in solving particle accelerator magnets with the h-type integral formulation is to identify a suitable discretization. We have found that the magnetic field between the poles of a dipole, a quadrupole, or a sextupole magnet is sensitive to the finite element mesh. Unless the mesh is properly refined, the B-field in the air gap tends to oscillate and to be slightly excessive.

Because of symmetry, one-fourth of the APS dipole magnet has to be discretized. We experimented with two different meshes. In the smaller case the number of nodes and tetrahedra were 3,212 and 13,257, respectively. The mesh is shown in Figure 9. The solution time was 7,896 seconds on a DEC Alpha 3000-600 AXP. The larger mesh contained 7,537 nodes and 34,645 elements; the solution time was 7,028 seconds on an IBM SP1 with 64 processors.

Results of the integral formulation are compared with results computed with TOSCA [33] using a very large number of elements and nodes. The end fields of the dipole along an arc in the center of the beam chamber computed with GFUNET and TOSCA are plotted in Figure 10. Here, the solid line is the result computed with TOSCA. The results computed with GFUNET are the squares (13,257 elements) and solid circles (34,645 elements).

The results show that the magnitude of the B-field is quite accurate with the less dense mesh, but the solution slightly oscillates in the region between the poles ($l < 372$ mm). Increasing the number of equations with a factor of two reduces the amplitude of the oscillation, but does not remove the problem completely. The oscillation problem is partly related to the tetrahedral mesh, and another problem is to define which way the mesh should be refined in order to increase accuracy. (In addition, the order in which the H^m-field is integrated from M's inside the tetrahedra may cause some cancellation errors in large problems.) In any case, this example clearly indicates that it is possible to solve geometrically complicated problems with integral equations. Further studies are needed, however, in order to recognize the source of the slight oscillation effect.

Parallel Version

Probably the main disadvantages of integral equation formulations are the dense matrix and the amount of memory and disk space required for data storage. In a parallel computing environment the amount of data to be kept in the main memory or stored into a temporary file is not such a restrictive problem as in a sequential computer. With a larger number of processors in use, the amount of memory and often also the amount of disk space available becomes larger. Therefore, a parallel computer offers not only more processing power, but also a platform for solving bigger problems.

The parallel version we have implemented is based on the formulation following from (30) and (31). The most important routines to run in parallel are the integral equation and Jacobian matrix generation, right-hand side generation, and the solver. In the rest of the routines, the CPU-time and the amount of communication needed among the processors are unimportant compared with the total CPU-time and the whole computing process. Therefore, details of the rest of the routines are not discussed in depth.

We split the matrices rowwise such that each processor generates and updates certain rows of the integral equation matrix and the Jacobian matrix. During generation of the matrices no data broadcasting between the processors is needed. In addition, only a very small amount of overlapping data has to be computed on two or more processors. As a result, the CPU-time needed to generate and update the matrices decreases linearly with an increasing number of processors. (The matrices are generated by computing the difference in scalar potential between the end nodes of each edge. As the matrices are decomposed by equations that are related to edges, and since nodes often belong to two or more edges, several processors may have to compute the same contributions to these common nodes.)

A challenging task in developing the parallel version is the solver. Since the matrix resulting from (30) is asymmetric (and nothing else is known), the system of equations have been solved with LU factorization. We have also developed a parallel version of a GMRES iterative solver. In general, the solution time of iterative solvers involving a dense matrix scale as $O(n^2)$ per iteration. The number of iterations is heavily dependent on the initial guess and on the choice of matrix preconditioner. Typically the GMRES iterative solver has been significantly faster than LU factorization in the tests we have carried out. Because we are still examining the performance of various preconditioners in a parallel computer, no timing results are presented in this paper.

In Table 3 timing results for the PAR-dipole magnet and for TEAM problem 13 with dense meshes are shown. The CPU-times of parallelized routines are measured on the leading processor, whereas the total CPU-time is measured in three different manners. The minimum, the maximum, and the average CPU-time taken by the processors are shown in order to give an insight into the balance of the load between the processors. (The H^s-field due to currents is computed with relative accuracy. Thus the load in the RHS generation is not distributed uniformly, and the CPU-time may decrease more quickly than linearly.) Results of the computed fields are shown in Figures 3, 4, and 10.

These results demonstrate the advantages of parallel computing when integral equations are employed. The most important routines lend themselves to concurrent computing, and a remarkable speedup can be achieved. In addition, because of large amounts of disk space and main memory, problems leading to very large dense systems of equations can be solved without difficulty and within reasonable time.

CONCLUSIONS

In this paper the theoretical background of a b and h-type volume integral formulation is presented, demonstrating that volume integral approaches can be developed from the same basis as partial differential equation and hybrid formulations. One can develop several

Table 3: Timing of various parallelized routines of GFUNET

	PAR-dipole			TEAM 13
Numer of nodes	7537			8997
Number of elements	34645			39014
Number of equations	7536			8995
Processors	16	32	64	64
LU-solver (s)	948.2	533.0	308.1	504
RHS generation (s)	396.3	204.6	71.7	7.2
Matrix generation (s)	824.3	412.6	207.6	288
Total CPU-time/min (s)	17895	10769	6383	7741
Total CPU-time/max (s)	18413	11530	7028	8467
Total CPU-time/aver. (s)	18306	11208	6878	8332

kinds of integral approaches connected with Whitney elements showing the advantage of imposing interface conditions as the physics suggests.

Numerical results with a sequential and a parallel version of the h-formulation demonstrate that integral equations are useful in applications problems. On the other hand, the dense equation matrix and the large amounts of data one has to store in the memory are inherent problems related to integral equations. In practice, one must have a relatively powerful workstation, with enough main memory and disk space, or a parallel computer. However, reasonably accurate results can be obtained even with a very small number of equations, but then the distribution of elements has to be carefully set. This restriction holds especially for problems in which the magnetic parts are excited below the saturation level. If the problem is magnetized above the saturation level, the h integral formulation is efficient even with a small number of elements.

Based on the tests we have carried out, we feel that in the static case integral equations are competitive in many cases. There are some important advantages of using integral equations in shape optimization, but the main realm is probably in time-dependent problems with moving objects. The fact that one need not discretize air and that exterior boundary conditions are automatically incorporated offers significant advantages in this kind of problem.

ACKNOWLEDGMENTS

We thank Armo Pohjavirta, John Simkin, C. W. Trowbridge, and Larry Turner for helpful discussions. We thank Sean Pratt, Jennifer Rovegno, Diana Tabor, Hania Yassin, and Vector Fields, Inc., for their assistance in developing the parallel version of GFUNET. The authors gratefully acknowledge use of the Argonne High-Performance Computing Research Facility. The HPCRF is funded principally by the U.S. Department of Energy Office of Scientific Computing. The work of the third and fourth authors was supported by the Office of Scientific Computing, U.S. Department of Energy, under Contract W-31-109-Eng-38.

APPENDIX A

Reduction of the Double Integration

Let us assume that matrix $\mathbf{H^m}$ in Equation (31) is generated using Equation (30). In this case $H^m = \sum_i h_i^m v_i$ and $H^s = \sum_i h_i^s v_i$ (the h_i^m's depend on h_j's). Thus the system of equations can be written in the form

$$\sum_i h_i \int_{V^m} \mu v_i \cdot v_j - \sum_i h_i^m \int_{V^m} \mu v_i \cdot v_j = \sum_i h_i^s \int_{V^m} \mu v_i \cdot v_j, \ \forall v_j \in W^1 \cap ker(curl). \quad (36)$$

Since the integral statements on the left- and right-hand side are equal, an equivalent solution for (36) can be found by solving

$$h_i - h_i^m = h_i^s, \ \forall i = 1, ..., n_{tree} \quad (37)$$

(i.e., the equations in (36) are linear combinations of Eq. (37)).

References

[1] K. J. Binns, P. J. Lawrensen, and C. W. Trowbridge, *The Analytic and Numerical Solution of Electric and Magnetic Fields*. Chichester: John Wiley & Sons, 1992.

[2] C. W. Trowbridge, "Electromagnetic computing: The way ahead?", *IEEE Trans. Magn.*, vol. 24, no. 1, pp. 13–18, 1988.

[3] J. Simkin, "A comparison of integral and differential equation solutions for field problems," *IEEE Trans. Magn.*, vol. 18, no. 2, pp. 401–405, 1982.

[4] M. J. Newman, C. W. Trowbridge, and L. R. Turner, "GFUN: An interactive program as an aid to magnet design," in *Proc. 4th Int. Conf. Magn. Tech.* (Brookhaven, New York), September 1972.

[5] C. J. Collie, N. J. Diserens, M. J. Newman, and C. W. Trowbridge, "Progress in the development of an interactive computer program for magnetic field design and analysis in two and three dimensions," Report RL-73-077, Rutherford Laboratory, July 1973.

[6] L. R. Turner, "Direct calculation of magnetic fields in the presence of iron, as applied to the computer program GFUN," Report RL-73-102, Rutherford High Energy Laboratory, August 1973.

[7] C. W. Trowbridge, "Computer aided design at the Rutherford laboratory," Report RL-74-133, Rutherford Laboratory, June 1974.

[8] A. G. A. M. Armstrong, C. J. Collie, N. J. Diserens, M. J. Newman, J. Simkin, and C. W. Trowbridge, "New developments in the magnet design computer program GFUN," Report RL-75-066, Rutherford Laboratory, March 1975.

[9] C. W. Trowbridge, "Applications of integral equation methods for the numerical solution of magnetostatic and eddy current problems," Report RL-76-071, Rutherford Laboratory, June 1976.

[10] C. H. Iselin, "A scalar integral equation for magnetostatic fields," in *Proc. COMPUMAG Conference on the Computation of Electromagnetic Fields* (Oxford), pp. 15–18, April 1976.

[11] J. E. Pasciak, "The H-gradient method for magnetostatic field computations," *IEEE Trans. Magn.*, vol. 19, no. 6, pp. 2344–2347, 1983.

[12] L. Han, L.S. Tong, J. Yang, and L. Zhao, "Integral equation method using total scalar potential for the simulation of linear or nonlinear 3d magnetostatic field with open boundary," in *Record of the 9th COMPUMAG Conference on the Computation of Electromagnetic Fields* (Miami), pp. 402–403, October 1993.

[13] H. Whitney, *Geometric Integration Theory*. Princeton University Press, 1957.

[14] A. Bossavit, "Whitney forms: A class of finite elements for three-dimensional computations in electromagnetism," *IEE Proc.*, vol. 135, Pt. A, no. 8, pp. 493–500, 1988.

[15] A. Bossavit, "Simplicial finite elements for scattering problems in electromagnetism," *Comp. Meth. in Appl. Mech. and Eng.*, vol. 76, pp. 299–316, 1989.

[16] A. Bossavit, "A numerical approach to transient 3d non-linear eddy-current problems," *Appl. Elec. in Mat.*, vol. 1, pp. 65–75, 1990.

[17] A. Bossavit, "Mixed methods and the marriage between "mixed" finite elements and boundary elements," *Num. Meth. Part. Diff. Eq.*, vol. 7, pp. 347–362, 1991.

[18] R. Albanese, R. Martone, G. Miano, and G. Rubinacci, "A T formulation for 3D finite element eddy current computation," *IEEE Trans. Magn.*, vol. 21, no. 6, pp. 2299–2302, 1985.

[19] R. Albanese and G. Rubinacci, "Integral formulation for 3D eddy-current computation using edge elements," *IEE Proc.*, vol. 135, Pt. A, no. 7, pp. 457–462, 1989.

[20] A. Bossavit, "Magnetostatic problems in multiply connected regions: Some properties of the curl operator," *IEE Proc.*, vol. 135, Pt. A, no. 3, pp. 179–187, 1988.

[21] C. J. Carpenter, "Comparison of alternative formulations of 3-dimensional magnetic-field and eddy-current problems at power frequencies," *IEE Proc.*, vol. 124, pp. 1026–1034, November 1988.

[22] J. Simkin and C. Trowbridge, "On the use of the total scalar potential in the numerical solution of field problems in electromagnetics," *Int J. Num. Meth. Eng.*, vol. 14, pp. 423–440, 1979.

[23] Z. Ren, F. Bouillault, A. Razek, and J. Verité, "Comparison of different boundary integral formulations when coupled with finite elements in 3d electromagnetic modelling," *IEE Proc.*, vol. 135, Pt. A, no. 8, pp. 501–507, 1988.

[24] J. Simkin, "Electromagnetic field computation using finite elements," in *Proc. Sem. Num. Meth. Elec. Eng.*, (Jyväskylä, Finland), Univ. Jyväskylä, Dept. Math., 1988.

[25] L. Kettunen, *Volume Integral Formulations for Three Dimensional Electromagntic Field Computation.* Publications 96, Tampere University of Technology, 1992.

[26] L. Kettunen and L. Turner, "A volume integral formulation for nonlinear magnetostatics and eddy currents using edge elements," *IEEE Trans. Magn.*, vol. 28, pp. 1639–1642, 1992.

[27] W. Gropp and B. Smith, *User's manual for the Chameleon parallel programming tools.* Math. and Comp. Science Div., Argonne Nat. Lab., ANL-93/23, June 1993.

[28] R. Butler and E. Lusk, *User's guide to the p4 Parallel Programming System.* Math. and Comp. Science Div., Argonne Nat. Lab., ANL-92/17, October 1992.

[29] J. Dongarra, A. Geist, R. Manchek, and V. Sunderam, "Integrated PVM framework supports heterogeneous network computing," *Computers in Physics*, vol. 7, pp. 166–75, April 1993.

[30] A. Beguelin, J. J. Dongarra, G. A. Geist, R. Manchek, and V. S. Sunderam, "A users' guide to PVM parallel virtual machine," Report TM-11826, Oak Ridge National Laboratory, July 1991.

[31] T. Nakata, N. Takahashi, K. Fujiwara, K. Muramatsu, T.Imai, and Y. Shiraki, "Numerical analysis and experiments of 3-D non-linear magnetostatic model," in *Proc. Int. Symp. and TEAM Workshop, Okayama, Japan*, vol. 9, Supplement A, pp. 308–310, 1990.

[32] T. Nakata, N. Takahashi, and H. Morishige, "Analysis of 3-d static force problem," in *Proc. of TEAM Workshop on Computation of Applied Electromagnetics in Materials*, (Sapporo, Japan) 1993.

[33] Vector Fields Ltd., 24 Bankside, Kidlington, Oxford, U.K.

[34] T. Nakata and K. Fujiwara, "Summary of results for benchmark problem 13 (3-d non-linear magnetostatic model)," in *Proc. Third Int. TEAM Workshop* (R. Albanese, E. Coccorese, Y. Crutzen, and P. Molfino, eds.) (Sorrento, Italy), pp. 223–249, Commission of the European Communities, Joint Research Center, EUR 14173 EN, 1991.

[35] T. Nakata, N. Takahashi, M. Nakano, H. Morishige, K. Matsubara, J. Coulomb, and J. Sabonnadiere, "Improvement of measurement of 3-d static force problem (problem 20)," in *Proc. of Miami Int. ACES/TEAM Workshop* (Miami, Florida), Florida Int. Univ., 1993.

Figure 1: A loop formed by a co-tree edge (boldface) and the corresponding path of tree edges

Figure 2: Geometry of TEAM problem 13 (steel parts shaded, the coil in white)

Figure 3: Average flux density in the steel plates of TEAM problem 13. Case 1: dotted line; Case 2: dashed line; Case 3: long-dashed line; Case 4 (parallel version results): solid circles; measurements [34]: solid line

Figure 4: Magnetic flux density in air of TEAM problem 13. Case 1: dotted line; Case 2: dashed line; Case 3: long-dashed line; Case 4 (parallel version results): solid circles; measurements [34]: solid line

Figure 5: TEAM problem 13: surface discretization of Case 3 (3,694 nodes; 14,625 tetrahedra)

Figure 6: Left: steel parts of TEAM problem 20. Right: steel parts and the coil of TEAM problem 20

Figure 7: One-fourth of the APS PAR-dipole magnet without coils

Figure 8: Top view of one-fourth of the APS PAR-dipole magnet with the coils

Figure 9: Discretization on the surface of the APS PAR-dipole magnet without the front plate

Figure 10: End field of the APS PAR-dipole along the center of the beam chamber. Smaller mesh: squares; dense mesh: solid circles; TOSCA [33]: solid line