Eur. Phys. J. Appl. Phys. **53**, 20801 (2011)
DOI: 10.1051/epjap/2010100270

THE EUROPEAN PHYSICAL JOURNAL APPLIED PHYSICS

Regular Article

Efficient generalized source field computation for h-oriented magnetostatic formulations

P. Dłotko[1] and R. Specogna[2,a]

[1] Jagiellonian University, Institute of Computer Science, Łojasiewicza 6, 30348 Kraków, Poland
[2] Università di Udine, Dipartimento of Ingegneria Elettrica, Gestionale e Meccanica, Via delle Scienze 208, 33100 Udine, Italy

Received: 1st July 2010 / Received in final form: 11 October 2010 / Accepted: 2 December 2010
Published online: 28 January 2011 – © EDP Sciences

Abstract. A technique based on a tree-cotree decomposition, called *Spanning Tree Technique* (STT) in this paper, has been shown to be simple and efficient to compute the *generalized source magnetic fields* for *h*-oriented magnetostatic formulations when solenoidal source electric currents over the faces of the mesh are given as input. Yet, it has been recently shown that STT may frequently fail in practice. Other techniques, which circumvent STT problems, have been proposed in literature. However, all of them greatly worsen the computational complexity and memory requirements regarding the source field computation. The aim of this paper is to present a generalization of STT called *Extended Spanning Tree Technique* (ESTT), which is provably general and it retains the STT computational efficiency.

1 Introduction

In *h*-oriented finite element method (FEM) magnetostatic formulations, or in the corresponding ones obtained by the Discrete Geometric Approach (DGA), cell method or Finite Integration Technique (FIT), algorithms based on the tree-cotree decomposition are commonly employed to compute the so-called *generalized source magnetic fields* [1] once the solenoidal source electric current over the faces of the mesh is given as input. Beside all the proposed algorithms, the one introduced in [2] – called *Spanning Tree Technique* (STT) in this paper – is particularly attractive, since it is simple to implement and exhibits an optimal complexity, being the running time linear with the number of mesh elements.

Let us consider an array of Degrees of Freedom (DoFs) $\mathbf{h}^s$, one for each edge of the mesh, needed to represent the rotational part of the magnetic field. The entry of the array $\mathbf{h}^s$ relative to each edge e, denoted as $\langle \mathbf{h}^s, e \rangle$, is defined as the line integral of the generalized source magnetic field h^s on the edge e. Hence, the array $\mathbf{h}^s$ contains the generalized source magneto-motive forces (mmfs) produced by the known source currents. The source currents are described by means of an array of DoFs $\mathbf{i}^s$ which contains as entries the currents on each face of the mesh. The STT is an algorithm which computes $\mathbf{h}^s$ when $\mathbf{i}^s$ is provided as input.

In electromagnetic modeling, it is customary to assume that the computational domain D is topologically trivial (i.e. it has the same homology as a ball). Let us assume that D is covered by a mesh whose incidence is encoded in the simplicial complex $\mathcal{K}$ and let us fix a spanning tree $\mathcal{T}$ using as graph the nodes and edges in $\mathcal{K}$. After setting the values of $\mathbf{h}^s$ relative to the spanning tree edges to zero, the STT enforces the discrete Ampère's law iteratively on each face of the mesh [2]. In fact, if the boundary edges of a face are all set but one, the $\mathbf{h}^s$ on the missing edge can be uniquely determined by using the discrete Ampère's law. This technique, recalled with more details in Section 2 of this paper, has been presented in many papers as [2–5]. STT is frequently claimed to be general in the literature, even though any attempt to rigorously analyze and prove the STT termination has been reported. The algorithm has been recently analyzed by the authors in [6] and it has been rigorously demonstrated – by concrete counter-examples – that various practical situations in which the algorithm fails exist. In particular, there exist situations involving a topologically trivial complex (for example, an arbitrary mesh of a cube or a ball) and a spanning tree on which the STT algorithm "hangs" in an infinite loop. Many different algorithms that do not present this problem – reviewed in Section 2.2 – have been proposed in the literature. Nonetheless, none of them exhibit a linear complexity. Better still, their running time is frequently bigger than the time needed to solve the original magnetostatic problem itself.

The aim of this paper is to introduce an extension of STT, called *Extended Spanning Tree Technique* (ESTT), which does not suffer from the failures of STT presented in [6] and in all practical cases it retains STT computational efficiency. The presented approach is based on the idea of symbolic computations that can be easily

[a] e-mail: ruben.specogna@uniud.it

implemented in any object oriented programming language or in Matlab®. The fundamental idea is, when STT hangs, to declare the value of $\mathbf{h}^s$ over one edge whose value is not computed yet to a (unknown) symbolic value and continue iterating with STT. At the end, some system of equations needs to be solved to compute the unknown symbolic values. Each equation of the reduced system enforces the discrete Ampère's law locally on one face f and involve in general some of the unknown symbolic values and some of the $\mathbf{h}^s$ already set. As it will be shown, in all practical cases the system results void or consists of few unknowns.

The paper is structured as follows. In Section 2 a survey of the STT algorithm and other alternative algorithms used for the generalized source field computation is addressed. Section 3 deals with the presentation of the *Extended Spanning Tree Technique* (ESTT), which eliminates the STT failures retaining STT computational efficiency. Some statistics are presented in Section 4 which enable to find the best spanning tree generation strategy suitable with the STT algorithm. Once the presented strategies are applied, the probability of STT failure is low but still possible. When using ESTT, the optimal tree generation strategy allows to minimize the size of the small linear system to be solved in ESTT. The impact of the extension of the generalized source field support on the computational time and on the quality of the solution is investigated in addition. In Section 5, the conclusion are drawn.

2 The Spanning Tree Technique (STT)

The array $\mathbf{h}^s$ has to be constructed in such a way that the discrete Ampère's law holds on every faces of the mesh

$$\mathbf{C}\,\mathbf{h}^s = \mathbf{i}^s, \tag{1}$$

where $\mathbf{C}$ is the usual incidence matrix between the faces and the edges of the mesh. $\mathbf{i}^s$ is a given array having an entry for each face of the mesh being a real (or complex) number corresponding to the current associated to the considered face[1]. To have a consistent solution, let us assume that the given array $\mathbf{i}^s$ represents a solenoidal current. Namely, it verifies $\mathbf{D}\mathbf{i}^s = \mathbf{0}$, where $\mathbf{D}$ is the incidence matrix between cells and faces. If this would not be the case, since $\mathbf{DC} = \mathbf{0}$, the following contradiction would arise

$$\mathbf{0} \neq \mathbf{D}\mathbf{i}^s = \mathbf{DC}\,\mathbf{h}^s = \mathbf{0}.$$

Therefore, a necessary condition in order to have a consistent solution is that a solenoidal current has to be provided as input. In cohomology theory such an array $\mathbf{i}^s$ represents a 2-cocycle, see for example [7,8]. Since the cell complex $\mathcal{K}$ is assumed to be homologically (and therefore cohomologically) trivial, each cocycle is a coboundary. Hence, there has to exist a 1-cochain $\mathbf{h}^s$ such that $\mathbf{C}\,\mathbf{h}^s = \mathbf{i}^s$. Consequently, the solenoidality of the current is a necessary and sufficient condition for the existence of a solution.

The rank of the $\mathbf{C}$ matrix is obviously not maximal, thus equation (1) has an infinite number of solutions. In fact, if two different arrays, $\mathbf{h}_1^s$ and $\mathbf{h}_2^s$ which represent two cocycles in the same cohomology class (i.e. which differ by the coboundary $\mathbf{G}$ of a magnetic scalar potential $\boldsymbol{\Omega}$), then the following holds

$$\mathbf{C}\mathbf{h}_1^s = \mathbf{C}(\mathbf{h}_2^s + \mathbf{G}\boldsymbol{\Omega}) = \mathbf{i}^s, \tag{2}$$

where $\mathbf{G}$ is the incidence matrix between the edges and the nodes of the mesh. Equation (2) holds since $\mathbf{CG} = \mathbf{0}$.

Now a tree-cotree decomposition is introduced. Let us fix a spanning tree $\mathcal{T}$ using as graph the nodes and edges in $\mathcal{K}$, together with the corresponding cotree $\mathcal{C}$. Let us order the edge's labels in such a way that the edges belonging to the cotree come first, followed by the edges in the tree. Therefore, in what follows, the new basis obtained after the reordering is considered in place of the old one. The matrix $\mathbf{C}$ and the vector $\mathbf{h}^s$ can be consequently partitioned in

$$\mathbf{C}_{\mathcal{C}}\mathbf{h}_{\mathcal{C}}^s = \mathbf{i}^s - \mathbf{C}_{\mathcal{T}}\mathbf{h}_{\mathcal{T}}^s. \tag{3}$$

The coefficients $\mathbf{h}_{\mathcal{T}}^s$ relative to tree edges can be fixed arbitrarily. It is well known, in fact, that fixing the value over the tree edges corresponds to eliminate the kernel $\ker(\mathbf{C})$ of the incidence matrix ([9], p. 106). Let us fix the coefficients of $\mathbf{h}^s$ relative to tree edges to zero. Since the kernel of the system of equation (3) has been eliminated, its rank becomes full and a unique solution of

$$\mathbf{C}_{\mathcal{C}}\mathbf{h}_{\mathcal{C}}^s = \mathbf{i}^s \tag{4}$$

exists.

The STT algorithm is introduced as an attempt to solve equation (4) by means of back-substitutions only [2]. In other words, if the algorithm succeed, there is no need to use a linear system of equation solver or even to explicitly construct the matrix $\mathbf{C}$ by using a sparse matrix data structure. In fact, $\mathbf{h}^s$ may be obtained by means of the STT algorithm presented in Table 1.

The STT termination has been taken for granted in many papers and no rigorous proofs regarding its termination or – on the contrary – counter-examples of non-termination have been provided. Recently, the authors have shown, by some concrete counter-examples, that various problems may arise [6].

2.1 A simple example of STT failure

To clearly see that the STT algorithm may fail, the following counter-example is presented [6]. A three-dimensional homologically trivial complex made by eight tetrahedra, twenty-two faces, twenty-one edges and eight nodes is considered. An exploded view of the tetrahedra is visible on the top of Figure 1. A spanning tree is formed by considering the thick edges represented on the bottom of Figure 1. If the STT algorithm – as implemented for example

[1] In case of magnetostatic problems or eddy-current problems solved in time domain, the current array is real-valued. In case of eddy-current problems solved in frequency domain, the current array is complex-valued.

Table 1. The STT algorithm.

```
checkBoundary ( simplex T )

1. int numOfEdg := 0, simplex F := 0, double sum := 0;
2. for every simplex E being an edge of T
   (a) if ⟨h^s, E⟩ is defined then
         i. numOfEdg + +;
        ii. sum := sum + C[F, E]⟨h^s, E⟩;
   (b) else F := E;
3. return (numOfEdg, sum, F).

STT ( simplicial complex K )

1. Generate T – a spanning tree of K. Let S be an empty
   list;
2. for every edge E ∈ T set ⟨h^s, E⟩ := 0;
3. for every triangle T ∈ K
   (a) (numOfEdg, sum, F) := checkBoundary( T );
   (b) if ( numOfEdg = 2 ) S := S ∪ T;
4. while( S is not empty )
   (a) take any T ∈ S; S := S \ T;
   (b) (numOfEdg, sum, F) := checkBoundary( T );
   (c) if (numOfEdg = 2) then
         i. ⟨h^s, F⟩ := −C[T, F] sum + ⟨i^s, T⟩;
        ii. for every triangle C whose boundary edge is F
            do
             A. (n, s, F') := checkBoundary( C );
             B. if (n = 2) S := S ∪ C;
5. return h^s.
```

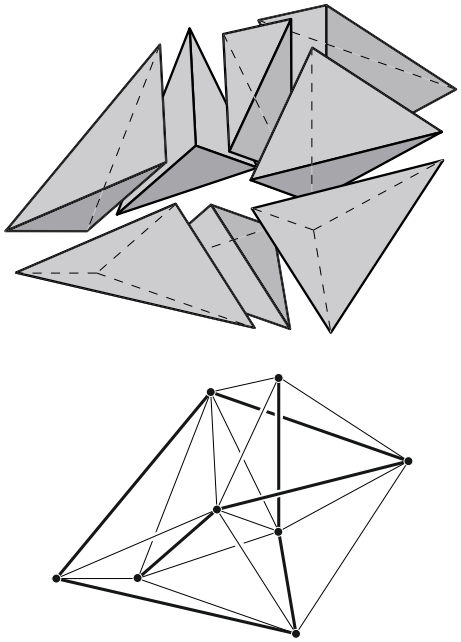

Fig. 1. A simple counter-example for the STT algorithm termination.

in Table 1 – is run, it hangs in an infinite loop. This is due to the fact that no edge can be set since each face has zero or one tree edges in its boundary. This pedagogical counter-example shows that the STT is not general, since its termination cannot be taken for granted.

When a gauged magnetostatic formulation is used, it is required to construct a tree that is complete on the boundary as described in [2–5]. The reader should be aware that also if the tree is complete on the boundary STT failures may easily happen, see a concrete counter-example in ([6], Sect. 5.1.2). In this paper, we concentrate on ungauged formulations, which are known to be more efficient with respect to the gauged ones. Dealing with ungauged formulations, no boundary condition is needed for the generalized source magnetic field.

A modification of the STT is proposed in [10] dealing with a cubical structured mesh. This algorithm, if applied on a tetrahedral mesh, frequently fails in practice as we are going to show. It turns out that it has even problems with two-dimensional complexes, where the STT is provably general. To show this, let us consider the two-dimensional complex in Figure 2a. The currents specified for each face are all zero except for the face S, represented by a dark triangle in Figure 2a. Let the current associated to S be an arbitrary non-zero value, say one to fix the ideas. In the first iteration of [10] algorithm, the dark triangle in Figure 2b is considered. A tree, represented in the picture by thick edges, is found on S and a zero value is associated to these edges. The value relative to the third edge is set to a non-zero value, indicated in the picture by a dotted edge (depending on orientations, not shown in the picture for the sake of clarity, the non-zero value is 1 or -1), by using the discrete Ampère's law. Next, all the triangles sharing an edge with S are considered. For each of them, a local tree is constructed by using as tree edges as many edges already imposed as possible. As usual, a zero value is associated to the tree edges and the cotree edges are determined accordingly to discrete Ampère's law, see Figure 2c. In all the pictures, the edges associated with a zero value are represented by thick black edges. After considering all the triangles in the set, a new set is formed by the triangles which share an edge with the set considered in the previous iteration and not yet considered, see Figure 2d. This procedure should run until all the triangles are considered. Let us analyze the output of the algorithm after the fourth iteration, see Figure 2e. As one can easily see, the output is wrong. In fact, let us concentrate on the cycle made by edges with a zero associated value represented in Figure 2e. This cycle imposes the circulation of the m.m.f. to zero, while should be once since the cycle encircle the unit current through S.

One purpose of this paper is to show that STT performs much better when the so-called *BFS trees* are used. A BFS tree can be obtained by using BFS (Breadth-First Search) strategy [11] to the graph consisting of the nodes and the edges of the mesh. Previously, it was known that this kind of trees – called also *minimal diameter trees* – produce less iterations in the STT [4,12] algorithm, but it was not pointed out that this is expressly required to reduce the probability of failure to an acceptable value.

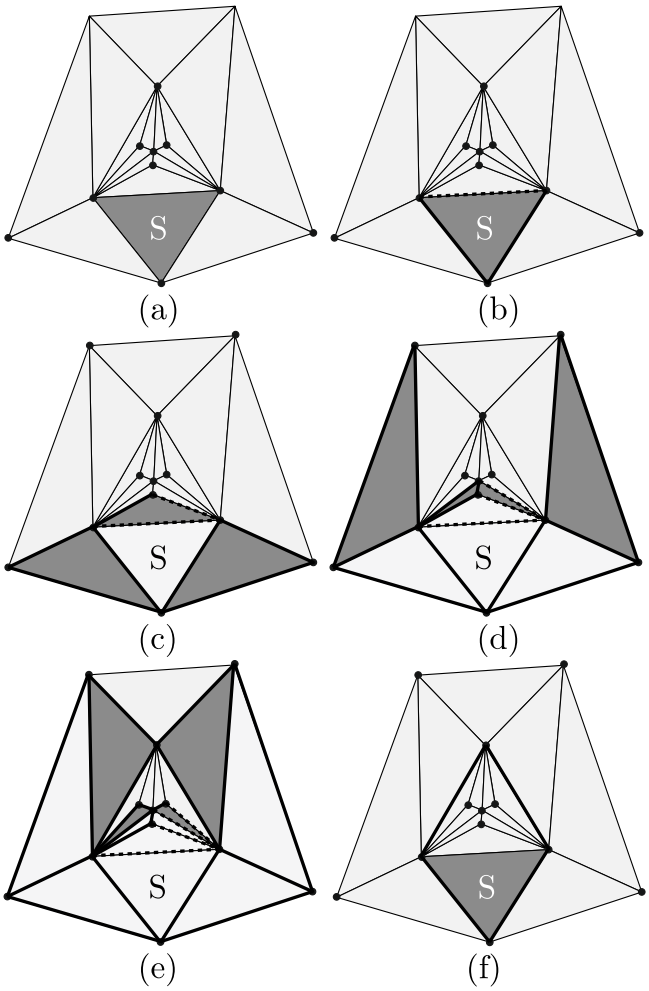

Fig. 2. A simple counter-example for the [10] algorithm. (a) The two-dimensional complex used in the counter-example. (b–e) The first four iterations. The result produced in the last (fourth) iteration is wrong, since discrete Ampère's law is violated. In fact, the m.m.f. circulation evaluated on the cycle represented in (f) by thick edges is zero, while should match the current though the triangle S.

2.2 A survey on different algorithms

Some alternatives with respect to the STT algorithm, that do not suffer from a lack of generality, have also been proposed in the literature.

A naïve solution to the problem would be to solve the linear system of equation (4) by a linear system solver. To this aim, a sparse matrix data structure has to be first created and the linear system has to be rigorously solved, which of course substantially increase both the computational time and the coding effort (in fact, an integer-based solver is usually not needed in FEM codes). For example, in [13] this is done with a reordering and a Gaussian elimination which is claimed to avoid the usual fill-in increasing during the Gaussian elimination. It is rather difficult to have some guarantees that the fill-in increasing will be small enough to enable to reach a solution in practice. After the factorization is available, the system is solved by back-substitution over reals. Nonetheless, it is well known that, during integer elimination, arbitrary big integers may be present in the computation, requiring a costly package to manage arbitrary large integers.

An iterative real-valued (or complex-valued) solver has been used for example in [14,15], where the source field is obtained by means of a FEM projection technique. Nonetheless, the cost of solving such a big linear system is not negligible concerning both the computational complexity and memory requirements. Actually, as already pointed out in [14], it requires even more computational effort than solving the original magnetostatic or eddy-current problem.

Techniques based on the so-called *fundamental cycles*[2] are also possible, see for example [1,16]. This approach requires a huge computational effort to retrieve all the fundamental cycles. Moreover, the currents over surfaces whose boundaries are the fundamental cycles have to be efficiently determined. Especially how to provide quickly these currents is not addressed at all in the cited papers. Solve this problem in general would require to find a surface whose boundary is the fundamental cycle, which again requires to solve rigorously a non-maximal rank system on integers.

To conclude, even though if these approaches are provably general, all of them increase the coding and computational complexity and the memory requirements with respect to STT. An effective extension of the STT, called *Extended Spanning Tree Technique* (ESTT), is introduced in the next Section. ESTT results provably general and exhibits a linear complexity on average.

3 The Extended Spanning Tree Technique (ESTT)

As already discussed in Section 2.1, the STT algorithm hangs in an infinite loop when there are still edges without a $\mathbf{h}^s$ value set but does not exist a triangle T having exactly one boundary edge still to be set. However, due to the uniqueness of the solution, we know that each edge value has to be set to a precise real (or complex) number.

The ESTT algorithm works exactly as the STT algorithm until it hangs in an infinite loop. When it hangs, the list of the remaining triangles is searched for a triangle T having a unique edge set by the algorithm.

- If such T is found, an edge E' of T for which the value $\langle \mathbf{h}^s, E' \rangle$ is not set yet, is set to the unknown fixed value x_1.
- If such a T is not found, then any triangle T is picked. Two edges E_1, E_2 of T are set to the unknown fixed values x_1 and x_2.

Since after this simple manipulation there exists a triangle T with exactly one boundary edge still to be set, the

[2] Let us consider a spanning tree over a graph. Each cotree edge form, together with some tree edges, one and only one cycle which is called fundamental cycle.

Table 2. C++ style class of extended number.

```
class extNumber
{
public:
    //here suitable arithmetic operators and
    //conversions should be implemented.
    extNumber( double val )
    {
        this->value = val;
        this->wasValueSet = true;
    }
    extNumber()
    {
        for(int i=0;i!=this->numberOfSymVar;++i)
             this->symbolicVar.push_back(0);
        this->symbolicVar.push_back(1);
        ++this->numberOfSymVar;
        this->wasValueSet = false;
    }
private:
     double value;
     bool wasValueSet;
     std::vector< double > symbolicVar;
     static int numberOfSymVar;
};
int extNumber::numberOfSymVar = 0.
```

STT algorithm can continue iterating. The only difference is that the algorithm operates on an extended implementation of real (or complex) numbers. On one hand, the number can be explicitly known, on the other, it represents some unknown fixed value. Such an implementation of a number may be obtained easily by using any object oriented programming language or Matlab®. Let us present in Table 2 an example of extended class of a real number implemented in C++. In the class presented in Table 2 two constructors are indicated. The first one creates a standard double variable packed in the `extNumber` class object. The second one, used when the STT algorithm hangs, creates a new unknown symbolic variable. To store the symbolic variables, the standard template library [17] vector `symbolicVar` is used. The symbolic variables are enumerated with integers starting from 0. To enumerate the variables, a static variable `numberOfSymVar` being the member of the `extNumber` class is used. Therefore, the value of the ith symbolic variable is stored at the ith position of `symbolicVar` vector. Such an approach enables fast and easy arithmetic operations using symbolic variables (which are in fact the arithmetic operations on the $symbolicVar$ vectors). The obvious details are left to the reader.

Now, the ESTT algorithm is presented. Instead of providing a full-length implementation, only the differences with respect to the STT algorithm are discussed. Namely, the point (4) of the STT algorithm is reorganized as already described in this Section to obtain the ESTT algorithm presented in Table 3.

Table 3. The changes that need to be applied to STT in order to obtain the ESTT algorithm.

($4a$) `while` (`true`)

1. `while`(S is not empty)
   (a) take any $T \in S$; $S := S \setminus T$;
   (b) $(\text{numOfEdg}, \text{sum}, F) =$ `checkBoundary`(T);
   (c) `if` $(\text{numOfEdg} = 2)$ `then`
      i. $\langle \mathbf{h}^s, F\rangle := -\mathbf{C}[T, F]\,\text{sum} + \langle \mathbf{i}^s, T\rangle$
      ii. `for` every triangle C whose boundary edge is F `do`
         A. $(n, s, F') =$ `checkBoundary`(C);
         B. `if` $(n = 2)$ $S := S \cup C$;
2. `if for` every edge E the value $\langle \mathbf{h}^s, E\rangle$ is set, `then break`;
3. search for a triangle T having the value $\langle \mathbf{h}^s, E\rangle$ set for exactly one edge E;
4. `if` such a triangle T exists, for one of its edges E' not already set impose the value by using `extNumber`() constructor;
5. `if` such a triangle T does not exists pick any triangle T'
   (a) Let E and E' be two edges not already set in the boundary of T';
   (b) Set the values for E and E' by using `extNumber`() constructor;
6. `for` every triangle T which have the value of exactly two boundary edges set `do` $S := S \cup T$;

($4b$) For every triangle $T \in \mathcal{K}$ if $\langle \mathbf{h}^s, C^T T\rangle$ is a nonzero unknown value add the equation $\langle \mathbf{h}^s, C^T T\rangle = 0$ to the linear system of equations;
($4c$) Solve the linear system of equations (one can use for example [20]) and set the values of the suitable edges.

3.1 Proof of ESTT generality

We already pointed out that the solution of the system (4) is unique. When the STT algorithm is propagating, setting a value to an edge E is equivalent to obtaining a unique solution of one equation in equation (4). This equation, together with the corresponding unknown, can be canceled out and both the rank of the resulting system and the number of unknowns is decreased by one. Therefore, since the rank of the system was maximal, during each iteration of the STT algorithm it remains maximal although the system itself gets smaller.

When STT hangs, the unknown symbolic variable is created and propagated by the ESTT algorithm. When adding a new symbolic variable, no value is computed and no equation is canceled out in the considered linear system. Therefore, the system and its rank remain invariant. Later, once that the propagation is resumed, the rank goes down again. It is therefore clear, that the rank of the resulting system is equal to the number of symbolic variables used in course of ESTT algorithm run.

Once the `while` loop in the ESTT algorithm terminates, the linear system of equations have to be created and solved in order to determinate the values of all

symbolic variables. When constructing the system, trivial equations involving some symbolic variables (for example the ones like $(x_1+1)-(x_1)+(1)=2$) are eliminated. Also the repetitions of the same equation are removed from the final system. In all the considered cases the number of equations obtained in this way matches the number of unknowns which is a very small or void in practice.

3.2 ESTT rigorous complexity analysis

When STT does not hang in an infinite loop, the performance of ESTT and STT are very similar considering the computational complexity[3]. In this case, ESTT uses a little more memory to store the numbers[4]. In practice, the extra cost of ESTT with respect to STT is not visible to the user.

Let us therefore analyze the time complexity of the STT algorithm presented in the Table 1. Since the tree can be constructed in linear time, the steps (1)–(3) need a linear time with respect to the number of elements in the complex. It is also clear that the procedure `checkBoundary` (`simplex` T) needs a constant time[5]. Therefore, also the body of the `while` loop in the point (4) of the algorithm needs a constant time. It remains to compute the number of iterations of the `while` loop in the point (4) of the algorithm. But it is clear that each triangle T will be considered only once. Hence, the complexity of the `while` loop is linear and consequently the complexity of the whole STT algorithm is linear.

When STT hangs in an infinite loop, an extra variable has to be added and, at the end, a linear system of equations has to be solved. Adding an extra variable in the algorithm does not affect the performance of the algorithm. The only important quantity is the number of the extra equations of the linear system that needs to be solved at the end. At the moment, there are no rigorous results to determine this size. Nonetheless, it seems very unlikely in practice to consider a mesh for which the linear system would consists in more than few, say three, equations. In fact, it is rare even to have a system with just two equations. Therefore, it is reasonable to assume that the number of extra equations on average is bounded by a constant. With this assumption it is straightforward to see that also the ESTT algorithm exhibits a linear complexity. In the worst possible case, the size of the system would be proportional to the size of the mesh. Anyway, the cost of solving it would be less than the algorithm presented in [13].

[3] The only difference is that STT uses processor directives to perform arithmetic operations on real (or complex) numbers, while ESTT uses objects of the class `extNumber` where the arithmetic operation can be effectively implemented as inline arithmetic operators. Hence, with a good implementation, the difference is not too big.

[4] This is due to the fact that, in the considered case, the class `extNumber` contains one real (or complex) number and one boolean value while STT needs only one real (or complex) number to be stored.

[5] Such a complexity may be achieved once the value $\langle \mathbf{h}^s, E\rangle$ is kept together with E as an extra field of the simplex data structure and can be accessed in constant time.

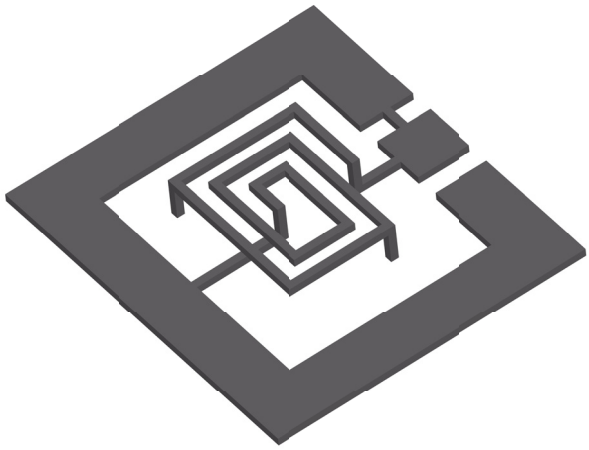

Fig. 3. The micro inductor used as a benchmark problem.

Table 4. STT performance with random trees.

#tetrahedra	#trees tested	#failures	Ratio
471	30 000	346	1.2%
2499	30 000	4562	15.2%
4603	30 000	8722	29.1%
12 271	11 400	8310	72.9%

4 Numerical results

The proposed algorithm has been applied to the source field computation on real-sized industrial magnetostatic problems without experimenting any difficulty. As an example, the micro inductor in Figure 3 surrounded by an insulating region is considered.

First of all, the statistics of STT failures on various meshes of the considered problem are produced. To this aim, the STT has been executed on thousands of random trees for each mesh producing the results in Table 4. The results show that, for big enough meshes, the probability of STT failure is very high. So we can conclude that STT does not perform satisfactorily by using random trees. By using ESTT, the correct result is obtained in each tested case.

Using BFS trees, no failures has been reported employing the STT on the benchmark problem. Nonetheless, the STT convergence using BFS trees cannot be proved, since concrete counter-examples exists. Namely, there exists three-dimensional meshes without holes and cavities in which STT fails even if using BFS trees. There is even some example, like the *Furch's knotted ball* [18], in which STT does not converge for *any* choices of BFS spanning tree. Hence, to have a provably good method which is reliable in practice, ESTT is expressly needed.

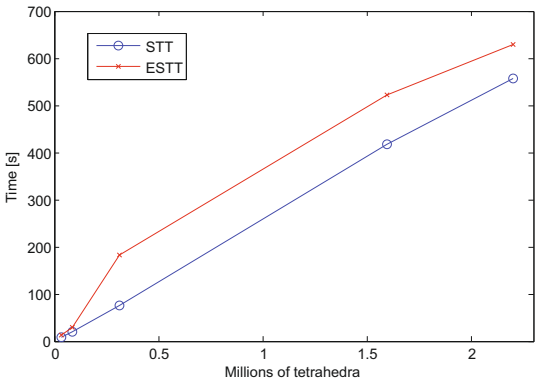

Fig. 4. (Color online) STT and ESTT timings.

The execution times of both STT and ESTT are represented in Figure 4. As expected, both show a linear complexity behavior.

Since ESTT is more general but slower, hence we propose the following algorithm employing a cascade of STT and ESTT:

1. The STT using a BFS tree is applied first. If STT converges, exit.
2. If STT fails, the slower but provably general ESTT is employed starting from the situation in which the STT stopped the propagation.

With this combination of STT and ESTT, one is able to get a fast and provably good algorithm for generalized source field computation.

In the literature, there has been some effort to reduce the support (i.e. reduce the number of edges with a non-zero $\mathbf{h}^s$ value) of the generated source fields $\mathbf{h}^s$. However, as already shown in [19], we experimented that this process does not worth the effort when STT/ESTT is used. This is because the time required to reduce the support is frequently greater than the time gained dealing with a compact support.

5 Conclusions

The *Spanning Tree Technique* (STT), widely used to impose sources in h-oriented formulations for magnetostatic problems, hangs in an infinite loop for some choices of the spanning tree. It has been shown, by using concrete examples, that these failures do happen frequently in practice and an extension is thereafter sought. Beside of the already proposed attempts to produce a different general and efficient algorithm, the aim of this paper has been to present the *Extended Spanning Tree Technique* (ESTT), which is provably general and yields to an optimal computational complexity. Some examples using real-sized three-dimensional finite element meshes are presented, showing the utility of ESTT for practical applications.

P.D. is partially supported by MNiSW grant N N206 625439.

References

1. P. Dular, F. Henrotte, F. Robert, A. Genon, W. Legros, IEEE Trans. Magn. **33**, 1398 (1997)
2. J.P. Webb, B. Forghani, IEEE Trans. Magn. **25**, 4126 (1989)
3. Y. Le Ménach, S. Clénet, F. Piriou, IEEE Trans. Magn. **34**, 2509 (1998)
4. F. Henrotte, K. Hameyer, IEEE Trans. Magn. **39**, 1167 (2003)
5. T. Henneron, S. Clenet, P. Dular, F. Piriou, J. Comput. Appl. Math. **215**, 438 (2008)
6. P. Dłoko, R. Specogna, SIAM J. Numer. Anal. **48**, 1601 (2010)
7. J.R. Munkres, *Elements of algebraic topology* (Perseus Books, Cambridge, MA, 1984)
8. P. Dłotko, R. Specogna, F. Trevisan, Comput. Meth. Appl. Mech. Eng. **198**, 3765 (2009)
9. G. Strang, *Linear algebra and its applications*, 3rd edn. (Thomson Business Information, Stanford, USA, 2003)
10. O. Biro, K. Preis, G. Vrisk, K.R. Richter, IEEE Trans. Magn. **29**, 1329 (1993)
11. T. Cormen, C. Leiserson, R. Rivest, C. Stein, *Introduction to Algorithms*, 2nd edn. (McGraw-Hill, 2002)
12. A. Murphy, Masters thesis, Boston University, 1991
13. Z. Cendes, Z. Badics, IEEE Trans. Magn. **43**, 1241 (2007)
14. O.-M. Midtgård, R. Nilssen, IEEE Trans. Magn. **34**, 2652 (1998)
15. C. Geuzaine, B. Meys, F. Henrotte, P. Dular, W. Legros, IEEE Trans. Magn. **35**, 1438 (1999)
16. K. Preis, I. Bardi, O. Biro, C. Magele, G. Vrisk, K.R. Richter, IEEE Trans. Magn. **28**, 1056 (1992)
17. N.M. Josuttis, *The C++ Standard Library: A Tutorial and Reference* (Addison-Wesley, USA, 1999)
18. G.M. Ziegler, Discrete Comput. Geom. **19**, 159 (1998)
19. T. Henneron, F. Piriou, A. Tounzi, S. Clénet, J.P.A. Bastos, N. Sadowski, J. Microw. Optoelectron. Electromagn. Appl. **8**, 135 (2009)
20. The Eigen Library, http://eigen.tuxfamily.org