Copyright © 2010 Tech Science Press CMES, vol.60, no.3, pp.247-277, 2010

Efficient Cohomology Computation for Electromagnetic Modeling

Paweł Dłotko[1] and Ruben Specogna[2]

Abstract: The systematic potential design is of high importance in computational electromagnetics. For example, it is well known that when the efficient eddy-current formulations based on a magnetic scalar potential are employed in problems which involve conductive regions with holes, the so-called *thick cuts* are needed to make the boundary value problem well defined. Therefore, a considerable effort has been invested over the past twenty-five years to develop fast and general algorithms to compute thick cuts automatically. Nevertheless, none of the approaches proposed in literature meet all the requirements of being automatic, computationally efficient and general. In this paper, an automatic, computationally efficient and provably general algorithm is presented. It is based on a rigorous algorithm to compute a cohomology basis of the insulating region with state-of-art reductions techniques—the acyclic sub-complex technique, among others—expressly designed for cohomology computations over simplicial complexes. Its effectiveness is demonstrated by presenting a number of practical benchmarks. The automatic nature of the proposed approach together with its low computational time enable the routinely use of cohomology computations in computational electromagnetics.

Keywords: Cell Method (CM), Finite Integration Technique (FIT), Finite Element Method (FEM), eddy-currents, potential design, thick cuts, computational cohomology, reduction methods, acyclic sub-complex technique.

1 Introduction

This paper considers the numerical solution of magneto-quasi-static Boundary Value Problems (BVP)—also called eddy-current problems—which are obtained by neglecting the displacement current in Ampère–Maxwell's equation [Maxwell (1891)].

[1] Jagiellonian University, Institute of Computer Science, Łojasiewicza 6, 30348 Kraków, Poland, dlotko@ii.uj.edu.pl.
[2] Università di Udine, Dipartimento di Ingegegneria Elettrica, Gestionale e Meccanica (DIEGM), via delle Scienze 208, 33100 Udine, Italy, ruben.specogna@uniud.it.

It is well known that two families of formulations for magneto-quasi-static problems exist, depending on the set of potentials chosen, see for example [Bossavit (1984)]. In this paper, the attention is focused on the so-called *h*-formulations, which are based on a magnetic scalar potential. The reason for this choice is that *h*-formulations are more efficient with respect to the complementary family of *b*-formulations, since they usually require about an order of magnitude less unknowns.

Nonetheless, when *h*-oriented formulations are employed in problems which involve conductive regions with holes, the design of potentials is not straightforward since the so-called *thick cuts* need to be introduced to make the BVP well defined, see for example [Dłotko, Specogna, and Trevisan (2009); Specogna, Suuriniemi, and Trevisan (2008); Ren (2002); Henrotte and Hameyer (2003)][1].

In this paper, the so-called Discrete Geometric Approach (DGA) is used as working framework. In the last years, the DGA gained popularity, becoming an attractive method to solve BVP arising in various physical theories, see for example [Weiland (1977); Bossavit (1998); Bossavit and Kettunen (2000); Tarhasaari, Kettunen, and Bossavit (1999); Specogna and Trevisan (2008); Specogna, Suuriniemi, and Trevisan (2008); Dłotko, Specogna, and Trevisan (2009); Codecasa, Specogna, and Trevisan (2009, 2010)]. The DGA presents some pedagogical and computational advantages with respect to the widely used Finite Element Method (FEM). First of all, the exploitation of the topological nature of Maxwell's equations and the geometric structure behind them, allows to reformulate the mathematical description of physical laws of electromagnetism directly in algebraic form. Such a reformulation can be elegantly formalized by using algebraic topology [Branin (1966); Tonti (1975, 1998)]. Taking advantage of this formalism, physical variables are modeled as cochains and Maxwell's laws are enforced by means of the coboundary operator. The information about the metric and the material properties are encoded in the constitutive relations, that are modeled as discrete counterparts of the Hodge star operator [Tarhasaari, Kettunen, and Bossavit (1999)] usually called *constitutive matrices*. Then, by combining Maxwell's laws formulated by means of the coboundary operator together with constitutive matrices, an algebraic system of equations is directly obtained, yielding to a simple, accurate and efficient numerical technique. Nonetheless, considering the DGA as working framework does not

[1] Other definitions of cuts have been introduced in the literature, due to the use of the old FEM nodal basis functions. In particular, the so-called `thin cuts` have been introduced both rigorously by means of homology [Kotiuga (1987, 1988, 1989); Suuriniemi (2004); Gross and Kotiuga (2004)] or by heuristic homotopy-based approaches, see for example [Harold and Simkin (1985); Leonard, Lai, Hill-Cottingham, and Rodger (1993); Simkin, Taylor, and Xu (2004); Dular (2005)]. The algorithms that generate thin cuts cannot be used for the thick cut extraction in general, as described in [Dłotko, Specogna, and Trevisan (2009)].

limit the generality of the results contained in this paper, since the Finite Element Method (FEM) and the Finite Differences (FD) can be easily reinterpreted in the DGA framework as well, see for example [Bossavit (1998); Tarhasaari, Kettunen, and Bossavit (1999)]. Consequently, the algorithms introduced in this paper can be employed, without any modification, for the automatic potential design in the corresponding widely used FEM formulation.

The originality of the approach induced by the DGA lies in the fact that the design of potentials is tackled directly within a discrete topological setting. In fact, thanks to the reformulation of Maxwell's laws by using the coboundary operator, homology and cohomology with integer coefficients are employed for the potential design in place of the continuous theory known as *de Rham cohomology*, routinely used especially in the FEM context, see for example [Ren (2002); Gross and Kotiuga (2004); Kotiuga (1987, 1988, 1989); Henrotte and Hameyer (2003); Specogna, Suuriniemi, and Trevisan (2008)].

It has been already shown in [Dłotko, Specogna, and Trevisan (2009)] that the thick cuts are generators of the 1-st cohomology group with integer coefficients of the insulating region. Even though a considerable effort has been made in the computational electromagnetic community to develop fast and general algorithms to produce thick cuts, all the proposed algorithms, reviewed in Section 4, are not satisfactory in practice, whether because they are not automatic, require an unacceptable amount of computational time or because of provable theoretical limitations in their generality.

A general algorithm for the computation of a basis of the 1-st cohomology group over integers is well known since many decades ago and it is based on the celebrated Smith Normal Form [Munkres (1984)] computation. The problem of this algorithm is that its computational complexity is hyper-cubical with the best implementation available [Storjohann (1996)] and consequently it cannot be used in practice even on extremely coarse meshes. Hence, some *reductions techniques* [Mrozek and Batko (2009); Mrozek, Pilarczyk, and Żelazna (2008); Kaczynski, Mrozek, and Ślusarek (1998)] are used to reduce the complex before the Smith Normal Form computation is run. Once the cohomology generators are found on the reduced complex via the standard Smith Normal Form algorithm, they are restored into the original complex by the so-called *pull-back* operation [Mrozek and Wanner (2010)]. These reduction techniques have proved to be efficient for homology computations [Dłotko, Specogna, and Trevisan (2009); Kaczynski, Mischaikow, and Mrozek (2004)], but we are not aware of any attempt to make a cohomology computation by using reduction techniques. In particular, a so-called *shaving* for cohomology is desired, which enables a reduction of the complex without the need of pulling-back the generators. This is due to the property that the cohomology generators of the shaved complex are generators also of the original complex.

The aim of this paper is to present an original, automatic, general, and efficient algorithm to compute cohomology generators and use it as a tool for the thick cuts computation in computational electromagnetics. The presented algorithm uses as shavings the reduction procedures presented in [Mrozek, Pilarczyk, and Żelazna (2008); Mrozek and Batko (2009)]. The presented method is tested over a number of practical benchmarks. An intuitive introduction to the computational aspects of homology and cohomology theory is provided in addition.

The 1-st cohomology group generators have been shown to be useful also to couple the geometric A-χ formulation with electric circuits, see [Dłotko, Specogna, and Trevisan (2010)]. Hence, the techniques presented in this paper can be also used for this purpose.

Recently, cohomology generators have been shown to be very useful also in computer science, being employed, for example, in global mesh parametrization, texture mapping, shape matching and shape morphing [Gu and Yau (2002); Gu, Wang, and Yau (2003); Guo, Li, Bao, Gu, and Qin (2006); Desbrun, Kanso, and Tong (2008)]. In all of these applications, fast algorithms to obtain the cohomology generators are needed. The methods presented in this paper can be used to obtain them.

The paper is structured as follows. In Section 2, a survey of the relevant topics of algebraic topology together with a link to electromagnetic modeling is provided. In Section 3, the need of a 1-st cohomology group basis for electromagnetic potential design is recalled. In Section 4, previous approaches to solve the problem of the thick cut computations are reviewed. In the Section 5, a detailed and intuitive presentation of the algorithms used in cohomology computations is addressed. In the Section 6, real-sized numerical examples are provided to show the efficiency and robustness of the presented method. Finally, in the Section 7, the conclusions are drawn.

2 Computational topology and computational electromagnetism

2.1 Geometrical mesh and simplicial complex

We assume that the domain of interest is meshed with a tetrahedral Finite Element mesh. The mesh is generated, from the considered geometry of the problem, by one of the standard mesh generators, for example [Schöberl (1997)]. Since we are concerned about computer algorithms, the mesh is assumed to be finite.

Although eddy-current formulations need the mesh of the whole computational domain (conductive plus insulating regions), cohomology generators have to be computed in the insulating region only [Dłotko, Specogna, and Trevisan (2009)].

Therefore, in the rest of the paper, we consider the restriction of the mesh in the insulating region only, which we denote as $\mathscr{M}$.

Since in this paper we are going to work on a discrete level only, it is important to distinguish two different concepts. The first one is the geometrical mesh $\mathscr{M}$. The second one is the simplicial complex, denoted in this paper by $\mathscr{K}$, which is going to be used in the presented algorithms.

We assume that the geometrical mesh $\mathscr{M}$, obtained by the mesh generator, consists of a set of tetrahedra. For each tetrahedron, the coordinates of its vertices are known. Each vertex, for simplicity, is uniquely determined by its unique integer label. Moreover, we assume that $\mathscr{M}$ is conformal, i.e. the intersection of any two tetrahedra is either empty, or its common vertex, edge or face.

Starting from the geometrical mesh $\mathscr{M}$, a structure called simplicial complex is constructed. A finite collection $\mathscr{K}$ of finite, non-empty sets is called a *simplicial complex*[2], if for every set $S \in \mathscr{K}$ and for every $T \subset S$ one has that $T \in \mathscr{K}$. In other words, a simplicial complex is a set of sets closed to the operation of taking a subset. The elements of $\mathscr{K}$ are referred to as *simplices*. A simplex $S \in \mathscr{K}$ such that the cardinality of S is equal to $k+1$ is referred to as k-simplex. The set of all k-simplices in $\mathscr{K}$ is denoted by $\mathscr{K}_k$. Moreover, for the sake of simplicity, we assume that the elements of the simplices in $\mathscr{K}$ are integer numbers being the labels of the vertices in $\mathscr{M}$. For a simplex $S \in \mathscr{K}_k$ by a *face* of S we mean any simplex $K \in \mathscr{K}_{k-1}$ such that $K \subset S$. When K is a face of S we say that S is a *coface* of K. We would like to point out that the presented definition of simplicial complex can be automatically used as a data structure to be stored in a computer. This, in fact, is described in the Section 5. The geometrical mesh $\mathscr{M}$ is a geometric object. We assume here that the set-theoretic sum of the tetrahedra in it forms the insulating region of the considered domain which is a subset of $\mathbb{R}^3$ having non-trivial 1-st homology group[3]. The simplicial complex $\mathscr{K}$ is a combinatorial structure used to effectively store the topological information about the mesh $\mathscr{M}$ in a computer, disregarding all the metric information in it. An algorithm that converts $\mathscr{M}$ into $\mathscr{K}$ is presented in Section 5.1.

2.2 Oriented simplices, chains and cochains

Let us assume that a simplicial complex $\mathscr{K}$ is given. In this Section, the basic concepts of algebraic topology are reviewed.

[2] Usually, in the mathematical literature, $\mathscr{K}$ is referred to as abstract simplicial complex. However, since this is the only complex considered in this paper, we decided to simplify the notation and call it simplicial complex.

[3] In this case there is a need for thick cuts. If the 1-st homology group of $\mathscr{K}$ is trivial, no thick cut is needed.

It turns out that simplices in a simplicial complex, being plain sets, are too general to introduce the definitions of the boundary and coboundary operators used in algebraic topology. There is the need of some kind of ordering of each simplex. Hence, the concept of *oriented simplex* is now introduced.

Let us consider all orderings of the $k+1$ elements of a given k-simplex S. The ordering of S is an arbitrary bijection $\sigma : \{0,\dots,k\} \to S$. Two orderings σ and σ' of S are said to be equivalent, if they differ by an even permutation. Therefore, the presented equivalence relation divides the set of all orderings of S into two equivalence classes. Such a class of ordering is called an *(inner) orientation* of S. By oriented simplex we mean a simplex with a chosen ordering. From now on, let us fix the orientation for each simplex and let us work on the oriented simplices only. For a k-simplex S, we fix the increasing ordering of the labels of the vertices in S, which (with a little abuse of notation) is denoted by $S = [x_0, x_1, \dots, x_n]$, where $x_i < x_{i+1}$ holds for $i \in \{0,\dots,n-1\}$[4].

In standard textbooks on algebraic topology, cohomology theory is always treated in a very abstract and hardly accessible way. Namely, it is introduced as a theory which is *dual* to the *homology theory*.

In this paper, we would like to present an algorithm-oriented approach which dramatically simplifies the exposition. For the standard approach to cohomology theory one can consult the book [Hatcher (2002)]. Moreover, in this paper we consider only homology and cohomology groups with integer coefficients. In fact, it is a standard result in algebraic topology that (co)homology with integer coefficients is the most general one (for a demonstration see for example Th. 3.2 and Th. 3.A.3 in [Hatcher (2002)]).

Let us introduce first the concept of *elementary cochain*. For a fixed oriented simplex $S \in \mathscr{K}$, an elementary cochain $\hat{S}$ is a map:

$$\hat{S}(K) = \begin{cases} 1 & \text{if } S = K\,, \\ 0 & \text{for all } K \in \mathscr{K} \setminus S\,. \end{cases}$$

A k-chain is a formal combination of oriented k-simplices with integer coefficients. The group of k-chains of $\mathscr{K}$ is denoted by $C_k(\mathscr{K})$. For $c \in C_k(\mathscr{K})$ we write $c = \sum_{S \in \mathscr{K}_k} \alpha^S S$. It is very natural and easy to store chains in a computer. In fact, the set of all k-simplices in $\mathscr{K}$ with the chosen orientation forms a basis of $C_k(\mathscr{K})$ and therefore the chain $c = \sum_{S \in \mathscr{K}_k} \alpha^S S$ can be stored as an array in which the value

[4] This, rather technical assumption, fruitfully simplifies the implementation. When computing the boundary and coboundary of each simplex there is the need to restrict the considered simplices. According to this convention, the considered ordering ensures that the orientation of the simplex $[x_0, x_1, \dots, x_{i-1}, x_i, x_{i-1}, \dots, x_n]$, restricted to the sub-simplex $[x_0, x_1, \dots, x_{i-1}, x_{i-1}, \dots, x_n]$, is the orientation of the considered sub-simplex.

α^S is stored in the place corresponding[5] to the simplex S. For a chain $c \in C_k(\mathscr{K})$ such that $c = \sum_{S \in \mathscr{K}_k} \alpha^S S$, the *support* of c is $|c| = \{S \in \mathscr{K}_k \text{ such that } \alpha^S \neq 0\}$.

A k-cochain c^* is a map $c^* : C_k(\mathscr{K}) \to \mathbb{Z}$. The group of all k-cochains is denoted by $C^k(\mathscr{K})$. Therefore, from the algorithmic point of view, the k-cochain c^* assigns to each k-chain $c \in C_k(\mathscr{K})$ an integer value. Of course, there are infinitely many possible k-chains. So, the basic question is: Is there a way to store a k-cochain (a map with an infinite domain!) in a computer? It is straightforward to see that the set of elementary k-cochains $\hat{S}$ for all $S \in \mathscr{K}_k$ forms a basis of $C^k(\mathscr{K})$. In other words, the value of the k-cochain on a k-chain is uniquely determined by the values on the oriented simplices and a cochain c^* can be written as $\sum_{S \in \mathscr{K}_k} \alpha^S \hat{S}$. Therefore, it is straightforward to see that k-cochains—as well as k-chains—can be stored into arrays. This property allows us to tailor without radical changes the existing code to compute homology groups and generators [*capd.ii.uj.edu.pl* (2010)], to compute cohomology groups and generators. For a cochain $c^* = \sum_{S \in \mathscr{K}_k} \alpha^S \hat{S}$, the *support* of c^* is defined as $|c^*| = \{S \in \mathscr{K}_k \text{ such that } \alpha^S \neq 0\}$.

In this Section we have talked about the groups of chains and cochains. The group is a formal mathematical structure with an operation (addition in our case). In a group there has to exist the neutral element of the operation (the chain c_z with all $\alpha^S = 0$, the cochain c_z^* being the zero map) and each element has to posses an inverse element with respect to the operation (for a chain $c = \sum_{S \in \mathscr{K}} \alpha^S S$ the inverse element is simply $-c = \sum_{S \in \mathscr{K}} -\alpha^S S$, for a cochain c^* the inverse is $-c^*$).

2.3 (Co)boundary operator and (co)homology

For $T \in \mathscr{K}_k$ such that $T = [x_0, \dots x_{i-1}, x_i, x_{i+1} \dots, x_k]$ and $S \in \mathscr{K}_{k-1}$ such that $S = [x_0, \dots, x_{i-1}, x_{i+1} \dots, x_k]$, we define $\kappa(T,S) := (-1)^i$. In the other case, we define $\kappa(T,S) := 0$.

For $k \geq 1$, the *boundary operator* $\partial_k : C_k(\mathscr{K}) \to C_{k-1}(\mathscr{K})$ is defined. For $S \in \mathscr{K}_k$ one has $\partial^k(S) = \sum_{T \in \mathscr{K}_{k-1}} \kappa(S,T)T$.

For $k \geq 1$, the *coboundary operator* $\delta^k : C^{k-1}(\mathscr{K}) \to C^k(\mathscr{K})$ is defined. For $S \in \mathscr{K}_{k-1}$ one has $\delta^k(S) = \sum_{T \in \mathscr{K}_k} \kappa(T,S)T$.

The idea behind the coboundary operator is presented in Fig. 1 by means of an example.

It is straightforward to verify that $\delta^{k+1} \circ \delta^k = 0$, as well as $\partial_k \circ \partial_{k+1} = 0$. The proof of this fact is very easy and can be found in every standard textbooks dealing with cohomology theory like [Hatcher (2002)].

The k-coboundary operator gives rise to a classification of cochains. From $\delta^{k+1} \circ$

[5] It is easy to provide some enumeration of all simplices in $\mathscr{K}_k$.

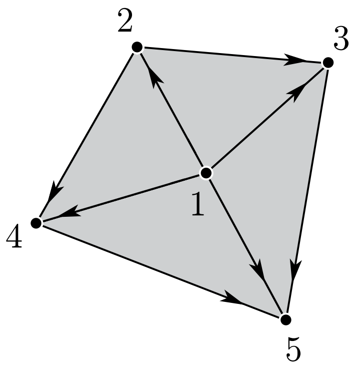

Figure 1: Let us consider the complex $\mathscr{K}$ illustrated in the picture above consisting of four 2-simplices, eight 1-simplices and five 0-simplices. In this case, $\delta^1[1] = -[1,2]-[1,3]-[1,4]-[1,5]$. Consequently, $\delta^2\delta^1[1] = -\delta^2[1,2]-\delta^2[1,3]-\delta^2[1,4]-\delta^2[1,5] = -[1,2,4]-[1,2,3]+[1,2,3]-[1,3,5]+[1,2,4]-[1,4,5]+[1,3,5]+[1,4,5]=0$. Analogously, $\partial_2[1,4,5]=[4,5]-[1,5]+[1,4]$ and $\partial_1\partial_2[1,4,5]=\partial_1[4,5]-\partial_1[1,5]+\partial_1[1,4]=[5]-[4]-[5]+[1]+[4]-[1]=0$.

$\delta^k = 0$, it is straightforward to verify that image of δ^k is a sub-group of kernel of δ^{k+1}. $im\delta^k$ is denoted as k-coboundary ($B^k(\mathscr{K})$) and $ker\delta^{k+1}$ is denoted as k-cocycle ($Z^k(\mathscr{K})$).

The k-cohomology group is defined as the quotient group $H^k(\mathscr{K}) = Z^k(\mathscr{K})/B^k(\mathscr{K})$.

An analogous situation holds for the chains. Therefore, the k-cycles $Z_k(\mathscr{K})$, k-boundaries $B_k(\mathscr{K})$ and the k-homology group $H_k(\mathscr{K}) = Z_k(\mathscr{K})/B_k(\mathscr{K})$ can be defined.

The dimension of the k-cohomology group is referred to as k-th Betti number ($\beta_k(\mathscr{K}) = dimH^k(\mathscr{K})$) [6].

We would like to point out that the cohomology group is a quotient group. Therefore, its elements are equivalence classes of cocycles. Two k-cocycles c^1, c^2 are in the same cohomology class if there exists a $(k-1)$-cocycle w such that $c^1 = c^2 + \delta^k w$. Each element of the cohomology class is referred to as a representant of the cohomology class.

In the algorithms we cannot deal with the whole equivalence class, but it suffice to have one representant for each cohomology class. Our algorithms, designed for

[6] Usually in homology theory by Betti number the dimension of homology group is denoted. Due to the homology-cohomology duality shown in Dłotko, Specogna, and Trevisan (2009), the presented definition is equivalent to the standard one in case of simplicial complexes embedded in $\mathbb{R}^3$.

cohomology computations, return both the Betti numbers and the representatives of the generators (one for each element of the cohomology basis).

The obvious question now is if the cohomology of all finite simplicial complex can be computed. In other words, can happen that Betti numbers are infinite or that there are infinitely many Betti numbers? In such a case, for obvious reasons, the cohomology computations would be futile. Fortunately, this is not the case. It is shown in [Kaczynski, Mischaikow, and Mrozek (2004)] that the cohomology group of a finite simplicial complex is a finitely generated abelian group[7]. Therefore, there is just a finite number of Betti numbers and all of them are finite. But we have even more. In our setting, as it is shown in [Dłotko and Specogna (2010)], the homology and cohomology groups are free groups[8]. Moreover, it is standard in algebraic topology that, for a simplicial complex of dimension 3, the cohomology group may be non-trivial only in dimensions 0, 1 and 2. Therefore, the output of the (co)homology computations is always finite. In Section 5, the original algorithms to compute cohomology groups introduced in this paper are presented.

2.3.1 Example of cohomology generators

Let us now present an example of simplices with non-zero coefficients in 1-cocycles which represent a cohomology basis of an annulus. The grey triangles in Fig. 2a represent a triangulated annulus, whose corresponding simplicial complex $\mathscr{K}$ consists of twelve 2-simplices, twenty-four 1-simplices and twelve 0-simplices. Two 1-cochains which have as supports the blue and red thick edges in Fig. 2a, respectively, are two different representatives of the generator for $H^1(\mathscr{K})$. The coefficients are also represented in the same Figure.

We would like to point out that every 1-cycle which is homologically non-trivial (i.e. which is not a boundary) in $\mathscr{K}$ has to cross the presented cohomology generator. Considering the number of times it crosses the cohomology generator (considering also the orientations of the edges) gives a detailed information about the homological nature of the 1-cycle.

In Fig. 2b, one of the two generators of the complex $\mathscr{K}$ obtained by triangulating the complement of a double torus (represented in grey in the picture) with respect to a larger box (only outlined in the picture for the sake of clarity) is represented. The depicted green edges lie in the support of one of the two 1-st cohomology group generators (the triangulation of the insulating region is not represented in the

[7] Being more precise, this result is shown in the cited reference for the homology group. However, the conclusion follows easily form the homology-cohomology duality presented in [Dłotko, Specogna, and Trevisan (2009)].

[8] Namely, the so-called torsion coefficients are not present. The torsion coefficients are present for instance in the (co)homology groups of the Klein bottle.

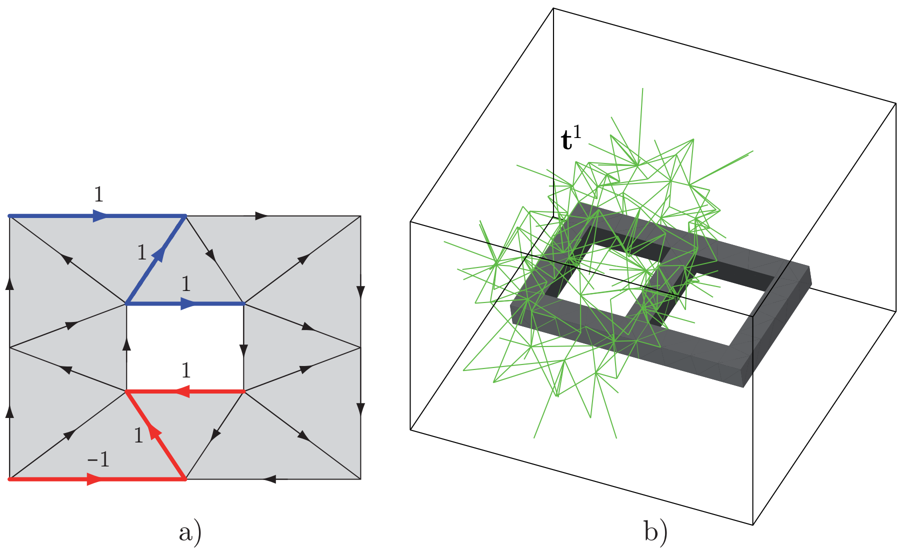

Figure 2: Examples of 1-st cohomology group generators.

picture for the sake of clarity).

3 Cohomology in electromagnetic modeling

In this Section, we intuitively explain why cohomology theory is useful in computational electromagnetics. For a more detailed explanation please refer to [Dłotko, Specogna, and Trevisan (2009); Dłotko and Specogna (2010)].

Let us concentrate on the design of potentials in the insulating region $\mathscr{K}$ for h-oriented eddy-current formulations[9]. The discrete Ampére's law in the insulating region can be written as

$$\delta \mathbf{F} = \mathbf{I} = \mathbf{0},$$

where $\mathbf{I}$ is the complex-valued electric current 2-cochain[10]—being zero by the hypothesis since $\mathscr{K}$ models the insulating region—and $\mathbf{F}$ is the magneto-motive force

[9] We assume to solve the eddy-current problem in frequency domain. If the eddy-current problem is solved in time domain, the group of reals has to be used in place of the group of complex numbers without any further changes.

[10] It is straightforward to define the complex-valued chains and cochains by using the group of complex numbers $\mathbb{C}$ in place of $\mathbb{Z}$ which has been used in the integer-valued chains and cochains definition. In this way, the complex-valued homology $H_k(\mathscr{K},\mathbb{C})$ and cohomology $H^k(\mathscr{K},\mathbb{C})$ group can be introduced.

(m.m.f.) complex-valued 1-cochain. Thanks to the discrete Ampére's law, **F** is a 1-cocycle in $\mathscr{K}$, hence $\mathbf{F} \in Z^1(\mathscr{K}, \mathbb{C})$.

Consequently, the 1-cocycle **F** can be expressed by the sum of a 1-coboundary $B^1(\mathscr{K}, \mathbb{C})$ and a basis of the 1-st cohomology group $H^1(\mathscr{K}, \mathbb{C})$. The 1-coboundary $B^1(\mathscr{K}, \mathbb{C})$ can be obtained by taking the 0-coboundary of a complex-valued 0-cochain magnetic scalar potential Ω. Hence we have

$$\mathbf{F} = \delta\Omega + \sum_{j=1}^{\beta_1(\mathscr{K})} \mathbf{T}^j_{cut},$$

where the $\{\mathbf{T}^j_{cut}\}_{j=1}^{\beta_1(\mathscr{K})}$ are the representatives of the 1-st cohomology group $H^1(\mathscr{K}, \mathbb{C})$ generators.

It is demonstrated in [Dłotko and Specogna (2010)] how fixing the basis for the cohomology group $\{\mathbf{T}^j_{cut}\}_{j=1}^{\beta_1(\mathscr{K})}$, fixes also the dot product (see [Hatcher (2002)]) of each cohomology generator over the generators $\{c_j\}_{j=1}^{\beta_1(\mathscr{K})}$ of the dual 1-st complex homology group $H_1(\mathscr{K}, \mathbb{C})$ basis

$$\langle \mathbf{T}^j_{cut}, c_i \rangle = i_j\, \delta_{ij},$$

where, thanks to Ampére's law, the dot product of **F** over the homology generator c_j have to match the electric current i_j *linked* by c_j

$$\langle \mathbf{F}, c_j \rangle = \langle \sum_{i=1}^{\beta_1(\mathscr{K})} \mathbf{T}^i_{cut}, c_j \rangle = \langle \mathbf{T}^j_{cut}, c_j \rangle = i_j,$$

see [Dłotko, Specogna, and Trevisan (2009)].

Now, from the Universal Coefficient Theorem for cohomology, as explained in [Dłotko and Specogna (2010)], each $\{\mathbf{T}^j_{cut}\}_{j=1}^{\beta_1(\mathscr{K})}$ can be written as

$$\mathbf{T}^j_{cut} = i_j \mathbf{t}^j,$$

where $\mathbf{t}^j$ is a representative of the 1-st cohomology group basis over integers.

Therefore it is straightforward to see that, in order to be able to design the potential in the insulating region, one needs the representatives of the $H^1(\mathscr{K})$ generators over integers. Efficient algorithms to obtain them are provided in Section 5.

4 Previous approaches for thick cuts generation

Few algorithms to generate thick cuts have been presented in the literature.

The chronologically first one has been introduced in [Ren (2002)]. It is a homotopy-based algorithm which is supposed to produce thick cuts. It is based on the intuitive idea of "growing a simply-connected bubble inside $\mathscr{K}$". Then, at the end of this process, the "complement of the bubble" is claimed to be the union of the supports of all the thick cuts. However, the lack of necessary details does not allow a serious analysis of the algorithm. In particular, it is not straightforward a) how the bubble is grown, b) how to find the coefficients of the edges for each thick cut separately from the complement of the bubble and c) how to discriminate cycles in the surface whose support form loops around holes from the ones that form loops around branches of the conductive region. Theoretical evidences can prove that this algorithm is unreliable in practice.

In [Henrotte and Hameyer (2003)], an algorithm which the Authors call *Generalized Spanning Tree Technique* (GSTT) has been introduced. This algorithm attempts to generate a basis for the 1-st cohomology group, once a basis for the 1-st homology group is given as input. In [Henrotte and Hameyer (2003)], homology generators have been constructed "by hand" and no discussion about the termination of the algorithm has been addressed. In [Dłotko, Specogna, and Trevisan (2009)], the Authors describe an efficient algorithm to automatically produce the homology generators and in [Dłotko and Specogna (2010)] a detailed analysis of the GSTT has been presented. In particular, in [Dłotko and Specogna (2010)], many problems are highlighted which are very difficult to solve in practice.

We would like also to point out that, as already shown in [Dłotko, Specogna, and Trevisan (2009)], all algorithms which compute the so-called *thin cut*, like the ones described in [Kotiuga (1987, 1988, 1989); Suuriniemi (2004); Gross and Kotiuga (2004); Harold and Simkin (1985); Leonard, Lai, Hill-Cottingham, and Rodger (1993); Simkin, Taylor, and Xu (2004); Dular (2005)], are not useful for thick cut computation in general.

5 Algorithms

In this Section, the algorithms to obtain 1-st cohomology group generators are presented. Our implementation heavily bases on the CAPD code [*capd.ii.uj.edu.pl* (2010)] for homology computations. The detailed explanation of homology computations for cubical complexes can be found in [Kaczynski, Mischaikow, and Mrozek (2004)].

Homology and cohomology theories have a long stand history. They have been fruitfully used to compute the so-called Conley index in the theory of dynamical systems [Mischaikow and Mrozek (1995)]. The main problem with the standard approach to homology and cohomology computations is the complexity of the standard algorithm, which is hyper-cubical [Storjohann (1996)] with respect to the number of simplices in $\mathscr{K}$. Therefore, it cannot be used in practical applications. Some faster approaches need to be found to make the cohomology computations viable in practice. The idea of the reduction algorithms (see [Kaczynski, Mischaikow, and Mrozek (2004); Kaczynski, Mrozek, and S̀lusarek (1998); Mrozek and Batko (2009); Mrozek, Pilarczyk, and Żelazna (2008)]) turned out to be a breakthrough. In fact, how to make a hyper-cubical algorithm feasible in practice? The answer is: Make the input of the algorithm as small as possible in a way that interesting information are preserved. Therefore, reduction algorithms are design as a pre-processing stage for the standard (co)homology algorithm[11]. During this pre-processing stage some simplices from the initial complex $\mathscr{K}$ are removed in a way that the cohomology group of the complex does not change.

Most of the reduction algorithms are design in order to obtain just the Betti numbers, because this was needed for the Conley index computations. Therefore, once a random reduction method is applied to the complex $\mathscr{K}$, usually cohomology generators in the reduced complex cannot be used as a cohomology generators in the initial complex. Since our need is to find exactly the representatives of cohomology generators, we would like to use in the computations a special class of reduction techniques referred to as *shaving*. When a shaving is applied to the complex $\mathscr{K}$, cohomology generators in the reduced complex are also cohomology generators in the initial complex. Such an approach makes the computation of cohomology generators faster. The speed-up is so sensible that—with these techniques—cohomology computations can be routinely used in practice. In this Section, apart form presenting standard algorithms for cohomology computations, we describe in details such a shaving procedures for cohomology computations.

5.1 Simplicial complex

In this Section, the data structure `Simplex` used to store simplices is presented. In order to be able to effectively apply shaving procedures, each `Simplex` is equipped with a pointer to its boundary and coboundary elements[12]. Moreover, during the shaving some elements are removed from the complex $\mathscr{K}$. In order not to change the whole structure of the `Simplicial Complex` when some elements are removed, the so-called *lazy delete* approach is used. In this case, every `Simplex` is also equipped with a boolean flag called `isDeleted`. This flag indicates if the `Simplex` is still in the complex (when it is set to `false`) or if it was already removed (when it is set to `true`). When presenting the algorithms, we use $C++$-like

[11] By a standard (co)homology algorithm we mean the Smith Normal Form algorithm which is essentially the same for homology and cohomology computations.

[12] By a coboundary element of a simplex A we mean a simplex B which has A in its boundary.

object-oriented programming style.

```
class Simplex
  1. set <Simplex* > boudanry;
  2. set <Simplex* > coboundary;
  3. boolean isDeleted;
  4. set < integer > vertices;
  5. Simplex(set < integer > v)
     (a) this- >vertices = v;
     (b) this- >isDeleted = false;
```

Table 1: The `Simplex` data structure.

It is assumed that the default value of the flag `isDeleted` is `false`. The data structure `Simplicial Complex` used to keep the simplicial complex in a computer is a simple aggregation of pointers to the `Simplex` data structure. We would like to point out that both in the class `Simplex` and in the class `Simplicial Complex` in the lists and sets we keep only the pointers to the `Simplex` data structure.

To be able to distinguish two data structures S_1, S_2 of a type `Simplex` one can implement a subroutine which compare the sets S_1`.vertices` and S_2`.vertices`. In further algorithms, for the sake of simplicity, this subroutine is not used explicitly being hidden into a set data structure[13].

Now, the algorithm to convert the geometrical mesh $\mathscr{M}$ to the `Simplicial Complex` $\mathscr{K}$ is presented in Table 3. It is presented as the constructor of the class `Simplicial Complex`. We assume that the vertices of each tetrahedron are marked with integer numbers as it is done in [Schöberl (1997)].

The operations on the sets can be effectively implemented. One can, for instance, use one of the standard template library implementations like [Josuttis (1999)]. After that the algorithm presented in Table 3 runs, the `Simplicial Complex` $\mathscr{K}$ is returned[14].

[13] Adding to the set S an element that is already in S does not have any effect. Therefore the set data structure have to be able to compare elements, and this is where the subroutine is used.

[14] Since the data structure `Simplicial Complex` returned by the algorithm is directly inspired by

```
class Simplicial Complex
    1. set <Simplex* > ZeroDimSimpl;
    2. set <Simplex* > OneDimSimpl;
    3. set <Simplex* > TwoDimSimpl;
    4. set <Simplex* > ThreeDimSimpl;
    5. Simplicial Complex( Geometrical mesh $\mathscr{M}$ )
```

Table 2: The Simplicial Complex data structure.

5.2 Shaving procedures

Let us assume that the Simplicial Complex $\mathscr{K}$ has been created as described in Section 5.1. In this Section, the details of the shaving procedures for cohomology computations are presented.

The so-called *acyclic sub-complex method* [Mrozek, Pilarczyk, and Żelazna (2008)] is a shaving for cohomology computation (the detailed mathematical proof of this fact will be published elsewhere). By an acyclic sub-complex $\mathscr{A}$ of a complex $\mathscr{K}$ we mean a set $\mathscr{A}$ having trivial homology in all dimensions except from dimension zero[15]. The dimension of 0-th homology group of $\mathscr{A}$ is 1. The idea of the acyclic sub-complex method is to remove from the simplicial complex $\mathscr{K}$ the largest possible acyclic sub-complex $\mathscr{A}$. We do not want to go into theoretical details here, but before we proceed to the algorithms themselves, let us present in Figure 3 an example which shows that the acyclic sub-complex algorithm is indeed a shaving for cohomology computations.

In the following, two essentially different algorithms to find the acyclic sub-complex are presented. The first one bases on the idea of building the acyclic space as presented in [Mrozek, Pilarczyk, and Żelazna (2008)]. The second one uses the so-called coreduction algorithm [Mrozek and Batko (2009)] to remove the acyclic sub-complex form the initial complex $\mathscr{K}$.

the abstract definition of simplicial complex $\mathscr{K}$ presented in Section 5.1, further on in this paper we use the notation $\mathscr{K}$ to indicate both the mathematical simplicial complex and the data structure Simplicial Complex returned by the Algorithm 3.

[15] The dimension of 0-th homology group measures the number of connected components of the complex.

```
Simplicial Complex( Geometrical mesh $\mathscr{M}$ )
  1. Let $\mathscr{K}$ be an empty Simplicial Complex;
  2. for every tetrahedron $T \in \mathscr{M}$
      (a) sort the integers representing $T$ in increasing ordering $[x_0,x_1,x_2,x_3]$;
      (b) for every $i \in \{0,1,2,3\}$ $\mathscr{K}$.$ZeroDimSimpl \cup$ $new$ Simplex($x_i$);
      (c) for every $i,j \in \{0,1,2,3\}$ such that $i < j$ $\mathscr{K}$.$OneDimSimpl \cup$ $new$Simplex($x_i,x_j$);
      (d) for every $i,j,k \in \{0,1,2,3\}$ such that $i < j < k$ $\mathscr{K}$.$TwoDimSimpl \cup$ $new$Simplex($x_i,x_j,x_k$);
      (e) $\mathscr{K}$.$TreeDimSimpl \cup$ $new$Simplex($x_0,x_1,x_2,x_3$);
  3. for every pair of Simplex $S,P \in \mathscr{K}$
      (a) if $S$ is a face of $P$, then
          i. $S.coface \cup P$;
          ii. $P.face \cup S$;
  4. return $\mathscr{K}$;
```

Table 3: The constructor of the `Simplicial Complex` class being the conversion of the geometrical mesh to the `Simplicial Complex` data structure.

5.3 Direct acyclic sub-complex computation

Let us give an idea on how the sub-complex $\mathscr{A}$ is created. The algorithm, which is inspired by the Algorithm 2 in [Mrozek, Pilarczyk, and Żelazna (2008)], can be found in Table 4. For each 3-simplex T, let $n(T)$ denotes a simple subroutine that returns all the neighbors of T in $\mathscr{K}$[16].

The details of the procedure *checkAcyclicity*$(\mathscr{A},T)$ are provided further on. For the moment, let us only assume that it exhibits a constant complexity and once it returns `true`, then the set $\mathscr{A} \cup T$ remains acyclic. The analysis of the complexity of the prototype of the algorithm presented in Table 4 can be found in [Mrozek,

[16] By a neighbor of a 3-simplex T we mean any 3-simplex W having non-empty intersection with T.

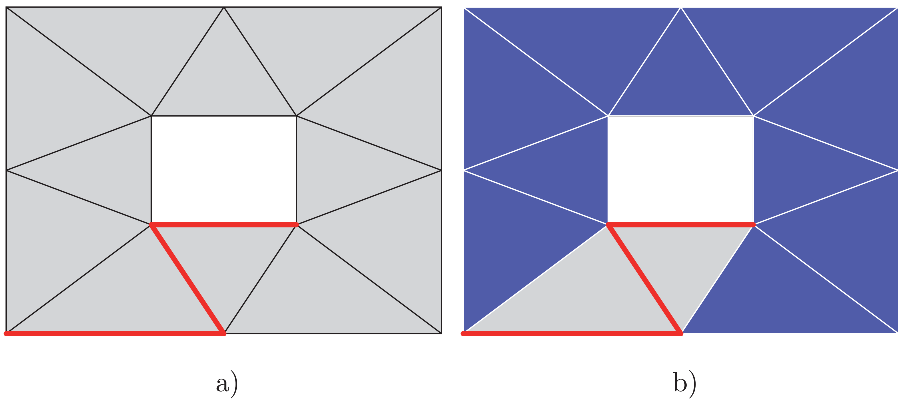

Figure 3: On the left, the simplicial complex representing an annulus is shown. The support of the cohomology generators is presented with the red thick edges. On the right, the blue 2-simplices are the simplices in the acyclic sub-complex $\mathscr{A}$. We would like to indicate that $\mathscr{A}$ is a closed complex (i.e. if an element is in $\mathscr{A}$, then all its faces are in $\mathscr{A}$). Therefore, the red edges which indicate the support of the cohomology generator in the reduced complex are the only edges remained in the reduced complex. This example illustrates that the acyclic sub-complex algorithm is indeed a shaving procedure for cohomology computations.

Pilarczyk, and Żelazna (2008)]. In fact, in case of simplices, it remains exactly the same. Therefore, for $n_0 = max_{T \in \mathscr{K}_3}$ $card\ n(T)$, the complexity of the algorithm is bounded by n_0 $card \mathscr{K}_3$ c, where c is the complexity of $checkAcyclicity(\mathscr{A}, T)$ procedure. Since in all meshes which are considered to be acceptable in electromagnetic modeling the number n_0 is a constant (i.e. does not grow for a fixed geometry when a sequence of refined meshes of this geometry are considered), then the complexity of the above algorithm may be considered to be linear.

Let us now discuss the details of the $checkAcyclicity(\mathscr{A}, T)$ procedure. In the acyclicity tests, as it is indicated in [Mrozek, Pilarczyk, and Żelazna (2008)], it is possible to use the corollary from the Mayer-Vietoris [Hatcher (2002)] Theorem. It roughly says that, if two sets A, B of simplices are acyclic and their intersection $A \cap B$ is acyclic, then the sum $A \cup B$ is also acyclic. But we know that the set $\mathscr{A}$ is acyclic. It is also straightforward that every tetrahedron T is acyclic. Therefore, if one confirms that the intersection $\mathscr{A} \cap T$ is also acyclic, then from the Mayer-Vietoris Theorem one has that $\mathscr{A} \cup T$ is acyclic. As it it indicated in [Mrozek, Pilarczyk, and Żelazna (2008)], the simplest and straightforward way of checking the acyclicity of the intersection $\mathscr{A} \cap T$ is simply by computing homology of

```
1. $\mathscr{A} := \emptyset$;
2. Let Q be empty set of pointers to Simplex;
3. Pick any $T \in \mathscr{K}$.ThreeDimSimpl; $\mathscr{A} := \mathscr{A} \cup T$;
4. Q := Q $\cup n(T)$;
5. while (Q $\neq \emptyset$)
    (a) $T := dequeue(\mathtt{Q})$;
    (b) if $checkAcyclicity(\mathscr{A}, T)$ then
        i. $\mathscr{A} := \mathscr{A} \cup T$;
        ii. for each $P \in n(T) \cap (\mathscr{K} \setminus \mathscr{A})$
            A. if $P \notin$Q then Q :=Q $\cup P$;
6. for every $T \in \mathscr{A}$
    (a) $T$.isDeleted = true;
    (b) for every $W$ being a boundary element of $T$ set $W$.isDeleted =
        true;
7. return $\mathscr{A}$;
```

Table 4: Acyclic sub-complex algorithm.

$\mathscr{A} \cap T$. Since this intersection would consists only of some boundary elements of T—that is, at most 14 elements—this test can be done in a constant time from the point of view of complexity analysis (although the constant would be reasonably large).

Therefore, we would like to present here two alternative approaches.

The first one, presented in Section 5.3.1, is based on the so-called coreduction algorithm [Mrozek and Batko (2009)]. In many practical cases, it enables to reduce the complex up to its homology generators. To do coreductions, similarly to the case of the acyclic sub-complex method, a more general structure with respect to the mathematical simplicial complex is needed. This is because the complex obtained during and after removing the acyclic sub-complex or by applying coreduction is not a simplicial complex anymore (the non-delted simplices in this complex do not have to have all its faces being non-deleted[17]). The detailed theoretical explanation of the coreduction algorithm can be found in [Mrozek and Batko (2009)]. Here, in the Figure 4, we would like to present the intuitive idea of the algorithm.

The second one, described in Section 5.3.2, is based on the idea of *lookup tables* for simplices, presented in [Mrozek, Pilarczyk, and Żelazna (2008)] for cubical complexes.

5.3.1 `checkAcyclicity` via local coreduction algorithm

Since removing coreduction pairs does not change the first and higher (co)homology groups (see [Mrozek and Batko (2009)]) of the simplicial complex, it is straightforward that if all the elements in $\mathscr{A} \cap T$ are removed with the coreductions, then $\mathscr{A} \cap T$ is an acyclic set. Consequently, in such case, T can be added to the set $\mathscr{A}$.

After introducing the method itself, let us present the algorithm in Table 5.

Intuitively, the algorithm presented in Table 5 puts to the set A all the elements in $\mathscr{A} \cap T$, applies the coreductions on those elements, and then, after detecting if all the elements in $\mathscr{A} \cap T$ are reduced, change back the fields `isDeleted` in all `Simplex` data structure elements in A.

It is straightforward that the presented algorithm works in linear time (the detailed proof can be found in [Mrozek and Batko (2009)]). Since the cardinality of the considered set A is bounded by 14 (the number of all boundary elements of a tetrahedron), the whole algorithm *checkAcyclicity*$(\mathscr{A}, T)$ works in constant time. Therefore, the total complexity of finding the maximal acyclic sub-complex $\mathscr{A}$ is linear with respect to the number of simplices in the complex $\mathscr{K}$. The experiments shown that this technique is much faster with respect to the pure Smith Normal Form computations.

5.3.2 `checkAcyclicity` via lookup tables

Despite the efficiency of the method presented in Section 5.3.1, no proof has been given that the algorithm presented in Table 5 returns `true` if and only if the intersection $\mathscr{A} \cap T$ is acyclic. One can only be sure that if it returns `true`, then the considered intersection is acyclic. Therefore another algorithm, which gives *if and only if* criterion, and which is faster than algorithm presented in Table 5, is now provided. It is based on a simple idea: Since the intersection $\mathscr{A} \cap T$ contains at most 14 elements, it is possible to check all the possible configurations of boundary elements of the 3-simplex T and list them all in a table as it is done in [Dłotko (2010)]. Then, for a given configuration, one is able to verify in a constant time if the considered configuration is acyclic or not by checking the suitable entry in the

[17] However, due to the `isDeleted` flag, this situation can easily be handled by our implementation.

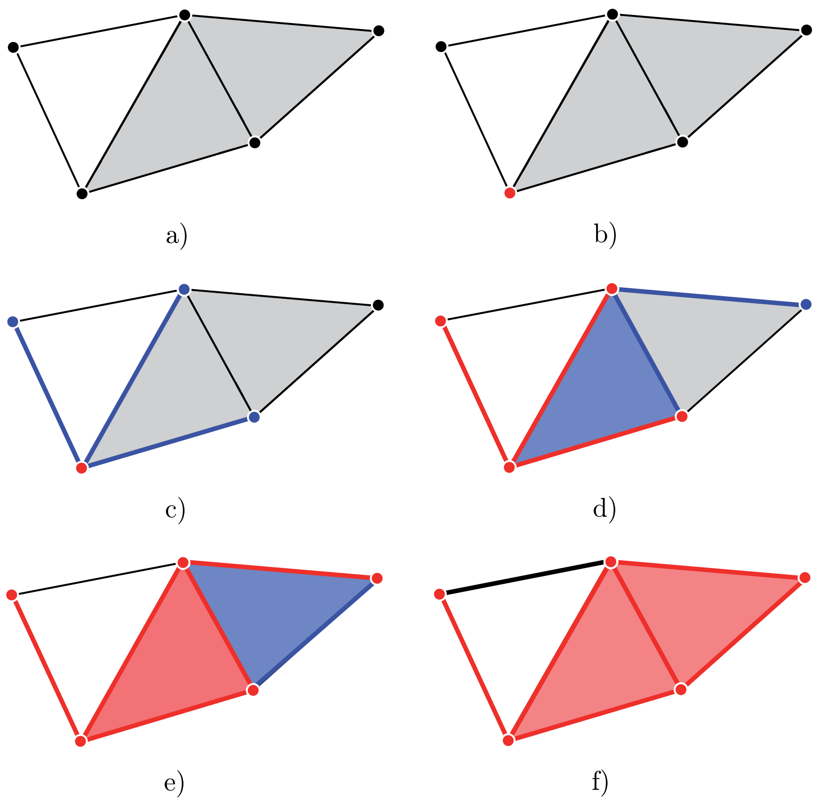

Figure 4: a) The complex having two 2-simplices, seven 1-simplices and five 0-simplices is presented. The idea of coreduction algorithm can be understood as computing the so-called *reduced homology*. b) In the first step, a single vertex is removed from the complex (the removed vertex is marked with red). Then, in a loop, as long as possible, the so-called *coreduction pairs* are removed. A simplex S is said to be a free coface, if it has only one non deleted simplex T in its boundary. In this case, the pair (S,T) is said to be a coreduction pair. In the pictures c), d) and e), the following steps of the coreduction algorithm are indicated. The coreduction pairs are indicated in blue. The deleted simplices are indicated in red. After all the pairs are removed, there is only one non deleted 1-simplex in the complex (indicated in bold black in the picture f)). In this case, we say that this simplex represents a 1-st homology generator in the reduced complex.

lookup table (in the suitable entry of the lookup table a boolean value indicates the acyclicity of the considered configuration).

```
boolean checkAcyclicity($\mathscr{A}, T$)

 1. Let a list $L = \emptyset$, set A $:= \mathscr{A} \cap T$ and an boolean value isAcyclic be
    given;
 2. Let $I \in A$ represents a 0-simplex;
 3. $I$.isDeleted = true;
 4. for every $S \in I$.coboundary$\cap A$ do $L := L \cup S$;
 5. while $(L \neq \emptyset)$
    (a) $S :=$pop_front$(L)$;
    (b) if $S$ is a free coface in $A$, and $T \in A$ is its unique face having
        $T$.isDeleted$=$false
          i. $T$.isDeleted $=$ $S$.isDeleted $=$ true;
         ii. for every $W \in T$.coboundary$\cap A$ such that
             $W$.isDeleted$=$false if $W$ is a free coface do $L := L \cup W$;
    (c) else for every $W \in S$.coboundary$\cap A$ such that
        $W$.isDeleted$=$false and $W$ is a free coface do $L := L \cup W$;
 6. if for every $S \in A$ $S$.isDeleted$=$true then isAcyclic $:=$ true
 7. else isAcyclic $:=$ false;
 8. for every $S \in A$ do $S$.isDeleted$=$false;
 9. return isAcyclic;
```

Table 5: CheckAcyclicity algorithm.

The presented method has turned out to be the fastest one. However, we decided to present also the other methods because to effectively use the lookup tables one needs to be able, for a given 3-simplex T, to return all its boundary element in the order presented in Table 6. This is an extra constraint which in case of some simplicial complex implementations may be problematic to achieve.

```
boolean $checkAcyclicityByLookupTable(\mathscr{A}, T)$
    1. Let $L$ be the lookup table obtained from Dłotko (2010);
    2. Let $T = [v_1, v_2, v_3, v_4]$ such that $v_1 < v_2 < v_3 < v_4$;
    3. Let $N$ be the vector of boundary elements of $T$ ordered as follows :
       $[v_1], [v_2], [v_3], [v_4], [v_1, v_2], [v_1, v_3], [v_1, v_4], [v_2, v_3], [v_2, v_4], [v_3, v_4]$,
       $[v_1, v_2, v_3], [v_1, v_2, v_4], [v_1, v_3, v_4], [v_2, v_3, v_4]$;
    4. Let $W$ be the vector, such that $W[i] \in \{0, 1\}$. Let $W[i] = 1$ if and only if
       $N[i] \in \mathscr{A} \cap T$;
    5. integer index $:= \sum_{i=0}^{13} 2^i W[i]$;
    6. return $L[index]$;
```

Table 6: `CheckAcyclicity` algorithm by using lookup table.

5.4 *Acyclic sub-complex via coreduction algorithm*

Let us now present the second way to obtain the acyclic sub-complex. It can be demonstrated that the set $\mathscr{B}$ removed from the initial complex $\mathscr{K}$ by the coreduction algorithm as in Figure 4 is acyclic. In fact, the coreduction algorithm is a different version of the acyclic sub-complex algorithm presented above. Therefore, one can alternatively use the coreduction algorithm as a shaving for cohomology computations. The shaving algorithm using coreduction is presented in Table 7.

As it has been already shown in Section 6, the complexity of the algorithm in Table 7 is linear with respect to the cardinality of $\mathscr{K}$. The presented method is slower with respect to the ones presented in Section 5.3.1 or in Section 5.3.1. However, it is very easy to implement and the general and efficient template implementations of the coreduction algorithm are available.

5.5 *Cohomology computations*

The reduction algorithms presented in Section 5.2 turn out to be very efficient in practice. After one of the two shaving procedure is applied, usually there is only the need for minor algebraic computations with respect to the algebraic computations necessary for the initial complex.

Let $\mathscr{K}$ be the initial complex and $\mathscr{A}$ be the acyclic sub-complex found by one of the

```
Acyclic_Sub-complex_via_coreduction( Simplicial Complex $\mathscr{K}$ )
    1. Let $\mathscr{A} := \emptyset$, L be an empty list;
    2. Let $I \in \mathscr{K}$ represents 0-simplex;
    3. $I$.isDeleted = true;
    4. $\mathscr{A} := \mathscr{A} \cup I$;
    5. for every $S \in I$.coboundary $L := L \cup S$;
    6. while $L \neq \emptyset$
        (a) $S$ :=pop_front($L$);
        (b) if $S$ is a free coface, and $T$ is its unique face having $T$.isDeleted=false
            i. $T$.isDeleted = $S$.isDeleted = true;
            ii. $\mathscr{A} := \mathscr{A} \cup \{T, S\}$
            iii. for every $W \in T$.coboundary such that $W$.isDeleted=false if $W$ is a free coface do $L := L \cup W$;
        (c) else for every $W \in S$.coboundary such that $W$.isDeleted=false and $W$ is a free coface do $L := L \cup W$;
    7. return $\mathscr{A}$;
```

Table 7: Acyclic sub-complex via coreduction algorithm.

shaving procedures presented in Section 5.2. Now it is time to turn all the elements in $\mathscr{K} \setminus \mathscr{A}$ into the coboundary matrix data structure[18]. We would like to remind that the idea of shaving is to exclude some elements from the complex $\mathscr{K}$, which are not important from the point of view of information about cohomology. Therefore, the coboundary operator restricted to the set $\mathscr{K} \setminus \mathscr{A}$ is considered further on during algebraic computations.

Before turning into standard Smith Normal Form algorithm, we use the KMS algo-

[18] In homology computations, boundary matrices are created at this point, whereas, during cohomology computations, coboundary matrices—which are the transposed boundary matrices—are needed.

rithm [Kaczynski, Mrozek, and Šlusarek (1998)] to minimize the amount of algebraic computations. The procedure of cohomology computation and pulling-back the generators[19] is exactly the same as in the case of homology computations. Since these techniques are clearly and elegantly presented in [Kaczynski, Mischaikow, and Mrozek (2004)], we decided not to repeat them here.

For the implementation of the considered algebraic procedures, the code [*capd.ii.uj.edu.pl* (2010)] has been used. The only difference with respect to the homology computations is that the coboundary operator is provided to the code instead of the boundary operator. This minor change allows to use the standard software designed to compute homology for cohomology computations.

6 Numerical examples

The proposed algorithms have been applied to the thick cut computations for real-sized industrial eddy-current problems without experimenting any difficulty. For example, a micro inductor and a micro transformer are considered, see Figs. 5 and 6, respectively. To visualize the thick cuts produced by the algorithms proposed in this paper for the considered examples, one may represent the set of edges in the support of each thick cut, see for example Fig. 2b. To gain more insight, it is useful to plot the dual faces dual with respect to the set of edges in the support of each thick cut, see [Dłotko, Specogna, and Trevisan (2009)]. Such a sets of dual faces for the two considered examples are shown in Figs. 7 and 8. The convergence of the inductance value of the micro inductor with mesh refinement is shown in Fig. 9. The eddy-current problems have been solved by the two complementary formulations on the same meshes. The same algorithm used to produce thick cuts for the T-Ω formulation in the insulating region can be applied to produce thick cuts in the conducting region needed to couple A-χ formulation with electric circuits, see [Dłotko, Specogna, and Trevisan (2010)]. From an engineering viewpoint, being able to us both complementary formulations is very important, since the mean value obtained by the two formulations is quite accurate even for coarse meshes. In Figs. 10 and 11 the execution times obtained by using various algorithms are compared for the micro inductor and micro transformer example, respectively.

The algorithms presented in this paper do not take any assumptions on the topology of the input mesh, being the algorithms provably general. The algorithms have also been applied with success in eddy-current problems consisting of knotted conductors. For example Fig. 12 shows a trefoil knot conductor and the thick cut for this

[19] Since a shaving was used, there is no need to pull-back the reductions back to $\mathscr{K}$. One needs only to find the cohomology generators in $\mathscr{K} \setminus \mathscr{A}$. In this case, after all algebraic computations, one needs to pull-back only the KMS reductions [Kaczynski, Mrozek, and Šlusarek (1998)]. This subroutine is already a part of the [*capd.ii.uj.edu.pl* (2010)] software.

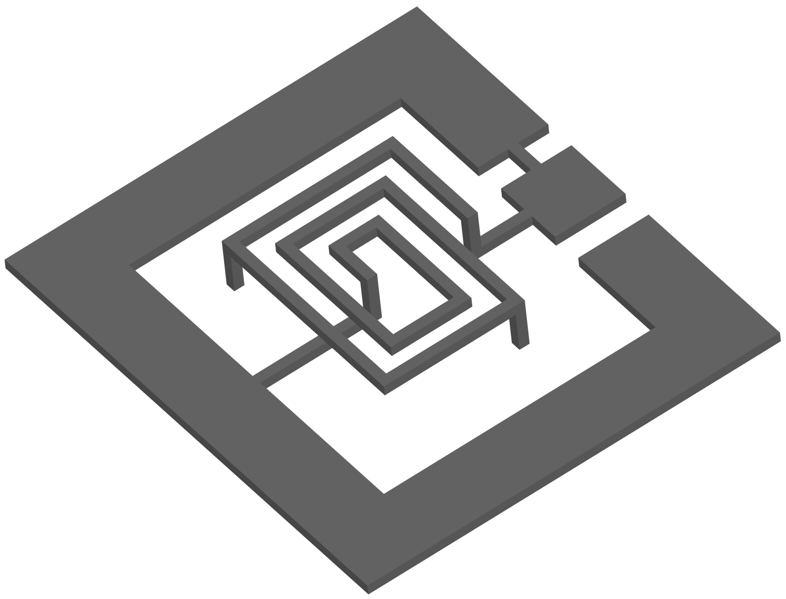

Figure 5: A micro inductor.

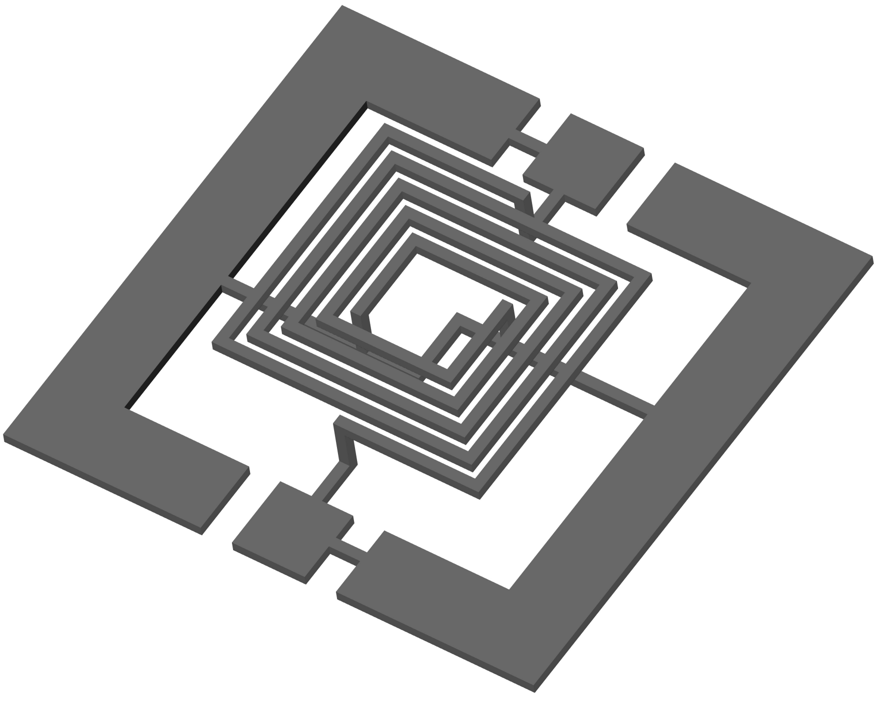

Figure 6: A micro transformer.

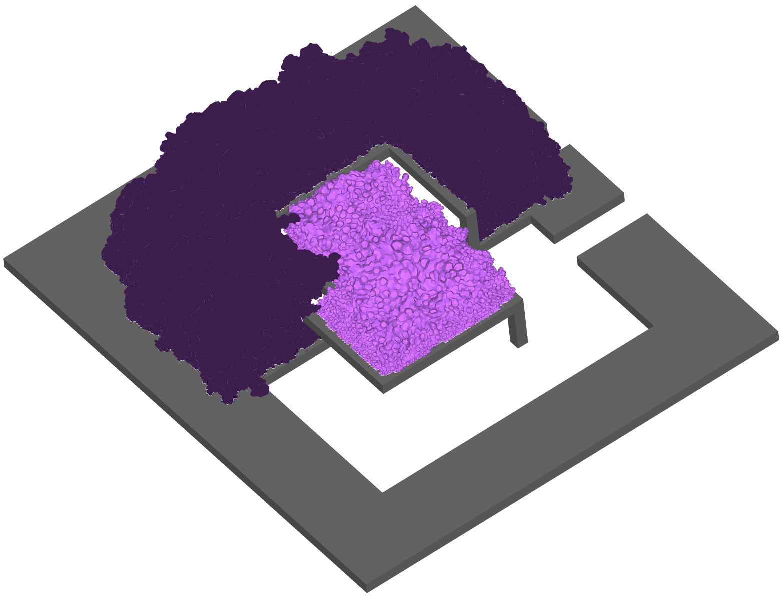

Figure 7: The thick cut for the micro inductor.

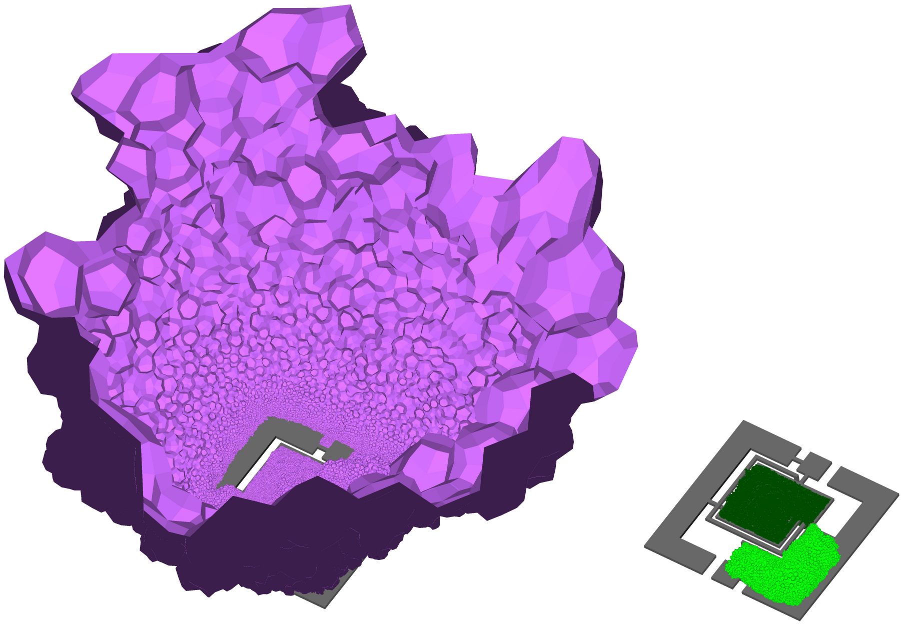

Figure 8: The two thick cuts for the micro transformer.

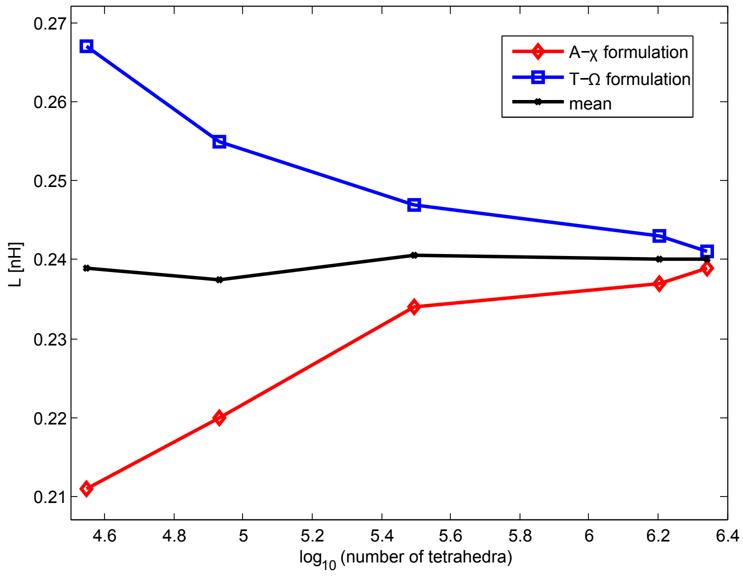

Figure 9: Convergence of the inductance value of the micro inductor with mesh refinement.

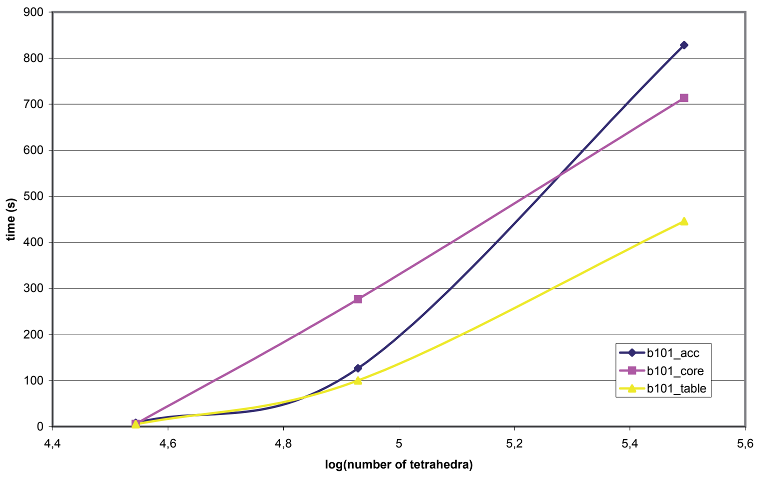

Figure 10: Execution times obtained by using various reduction techniques for the micro inductor example.

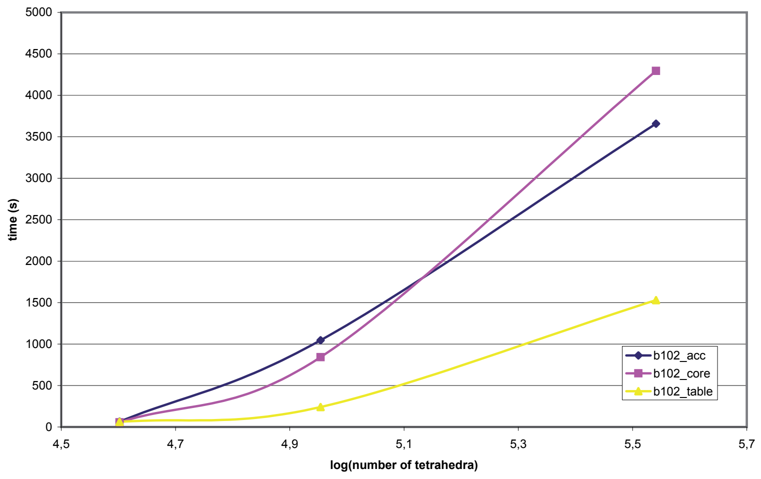

Figure 11: Execution times obtained by using various reduction techniques for the micro transformer example.

example is shown in Fig. 13.

7 Conclusion

In this paper, a new automatic, general and efficient technique for the 1-st cohomology group computation over integers has been introduced. All of the algorithms have been described in detail. This technique turned out to be fundamental, among the other applications of cohomology computation, for the generation of the thick cuts, needed for the potential design in eddy-current h-formulations. The feasibility of the presented technique with real-sized industrial problems has been demonstrated by concrete examples. We believe that the presented approach should be routinely included in the next-generation electromagnetic solvers.

Acknowledgement: The Authors would like to thank Marian Mrozek for many discussions and help without which this paper could not be written.

References

Bossavit, A. (1984): Two dual formulations of the 3d eddy currents problem. *COMPEL*, vol. 4, pp. 103–116.

Bossavit, A. (1998): How weak is the weak solution in finite elements methods? *IEEE Trans. Mag.*, vol. 34, pp. 2429–2432.

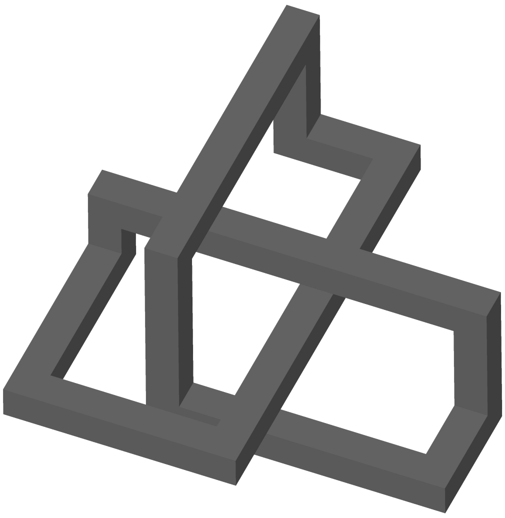

Figure 12: A trefoil knot massive conductor.

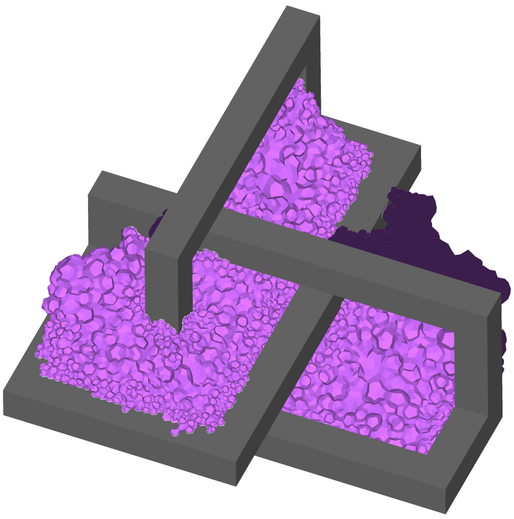

Figure 13: The thick cut for the trefoil knot massive conductor.

Bossavit, A.; Kettunen, L. (2000): Yee-like schemes on staggered cellular grids: A synthesis between fit and fem approaches. *IEEE Trans. Mag.*, vol. 36, pp. 861–867.

Branin, F. (1966): The algebraic-topological basis for network analogies and the vector calculus. *Proceedings of the Symposium on Generalized Networks, Polytechnic Press, Brooklin, NY*, pp. 453–491.

capd.ii.uj.edu.pl (2010): The capd library, 2010.

Codecasa, L.; Specogna, R.; Trevisan, F. (2009): Base functions and discrete constitutive relations for staggered polyhedral grids. *Comput. Meth. Appl. Mech. Eng.*, vol. 198, no. 9–12, pp. 1117–1123.

Codecasa, L.; Specogna, R.; Trevisan, F. (2010): A new set of basis functions for the discrete geometric approach. in press, 2010.

Desbrun, M.; Kanso, E.; Tong, Y. (2008): Discrete differential forms for computational modeling. *Oberwolfach Seminars, Discrete Differential Geometry, Birkhäuser, Basel, Switzerland*, vol. 28, pp. 287–324.

Dłotko, P. (2010): Acyclic configurations for boundary of 3 and 4 dimensional simplices, *www.ii.uj.edu.pl/ dlotko/accconf.html*, 2010.

Dłotko, P.; Specogna, R. (2010): Critical analysis of the spanning tree techniques. submitted to SIAM Journal on Numerical Analysis, 2010.

Dłotko, P.; Specogna, R.; Trevisan, F. (2009): Automatic generation of cuts suitable for the t-ω geometric eddy-current formulation. *Comput. Meth. Appl. Mech. Eng.*, vol. 198, pp. 3765–3781.

Dłotko, P.; Specogna, R.; Trevisan, F. (2010): Voltage and current sources for massive conductors suitable with the $a-\chi$ geometric formulation. *IEEE Transactions on Magnetics*, vol. 46.

Dular, P. (2005): Curl-conform source fields in finite element formulations: Automatic construction of a reduced form. *COMPEL*, vol. 24, pp. 364–373.

Gross, P.; Kotiuga, P. (2004): *Electromagnetic Theory and Computation: A Topological Approach.* Cambridge University Press.

Gu, X.; Wang, Y.; Yau, S.-T. (2003): Multiresolution computation of conformal structures of surfaces. *Journal of Systemics, Cybernetics and Informatics*, vol. 1, pp. 45–50.

Gu, X.; Yau, S. (2002): Computing conformal structures of surfaces. *Communications in Information and Systems*, vol. 2, pp. 121–146.

Guo, X.; Li, X.; Bao, Y.; Gu, X.; Qin, H. (2006): Meshless thin-shell simulation based on global conformal parameterization. *IEEE Trans. Vis. Comput. Gr.*, vol. 12, pp. 375–385.

Harold, C.; Simkin, J. (1985): Cutting multiply connected domains. *IEEE Trans. Magn.*, vol. 21, pp. 2495–2498.

Hatcher, A. (2002): *Algebraic topology*. Cambridge University Press, Cambridge, UK.

Henrotte, F.; Hameyer, K. (2003): An algorithm to construct the discrete cohomology basis functions required for magnetic scalar potential formulations without cuts. *IEEE Trans. Magn.*, vol. 39, pp. 1167–1170.

Josuttis, N. M. (1999): *The C++ Standard Library: A Tutorial and Reference*. Addison-Wesley U.S.A.

Kaczynski, T.; Mischaikow, K.; Mrozek, M. (2004): *Computational Homology*. Springer-Verlag, New York, USA.

Kaczynski, T.; Mrozek, M.; Ŝlusarek, M. (1998): Homology computation by reduction of chain complexes. *Computers and Mathematics*, vol. 35, pp. 59–70.

Kotiuga, P. (1987): On making cuts for magnetic scalar potentials in multiply connected regions. *J. Appl. Phys.*, vol. 61, pp. 3916–3918.

Kotiuga, P. (1988): Toward an algorithm to make cuts for magnetic scalar potentials in finite element meshes. *J. Appl. Phys.*, vol. 63, pp. 3357–3359.

Kotiuga, P. (1989): An algorithm to make cuts for magnetic scalar potentials in tetrahedral meshes based on the finite element method. *IEEE Trans. Magn*, vol. 25, pp. 4129–4131.

Leonard, P.; Lai, H.; Hill-Cottingham, R.; Rodger, D. (1993): Automatic implementation of cuts in multiply connected magnetic scalar region for 3-D eddy current models. *IEEE Trans. Magn.*, vol. 29, pp. 1368–1371.

Maxwell, J. (1891): A Treatise on Electricity and Magnetism. *Oxford : Clarendon Press*.

Mischaikow, K.; Mrozek, M. (1995): Chaos in lorenz equations: a computer assisted proof. *Bull. Amer. Math. Soc. (N.S.)*, vol. 33, pp. 66–72.

Mrozek, M.; Batko, B. (2009): Coreduction homology algorithm. *Discrete and Computational Geometry*, vol. 41, pp. 96–118.

Mrozek, M.; Pilarczyk, P.; Żelazna, N. (2008): Homology algorithm based on acyclic subspace. *Computers and Mathematics*, vol. 55, pp. 2395–2412.

Mrozek, M.; Wanner, T. (2010): Coreduction homology algorithm for inclusions and persistent homology. Preprint available online, 2010.

Munkres, J. R. (1984): *Elements of Algebraic Topology*. Addison-Wesley.

Ren, Z. (2002): t-ω formulation for eddy-current problems in multiply connected regions. *IEEE Trans. Magn.*, vol. 38, pp. 557–560.

Schöberl, J. (1997): Netgen - an advancing front 2d/3d-mesh generator based on abstract rules. *Computing and Visualization in Science*, vol. 1, pp. 41–52.

Simkin, J.; Taylor, S.; Xu, E. (2004): An efficient algorithm for cutting multiply connected regions. *IEEE Trans. Magn.*, vol. 40, pp. 707–709.

Specogna, R.; Suuriniemi, S.; Trevisan, F. (2008): Geometric t-ω approach to solve eddy-currents coupled to electric circuits. *Int. J. Numer. Meth. Eng.*, vol. 74, pp. 101–115.

Specogna, R.; Trevisan, F. (2008): Eddy-currents computation with T-Ω discrete geometric formulation for a NDE problem. *IEEE Trans. Mag.*, vol. 44, pp. 698–701.

Storjohann, A. (1996): Near optimal algorithms for computing smith normal form of integer matrices. *Proceedings of the 1996 international symposium on symbolic and algebraic computation, ISAAC 1996*, pp. 267–274.

Suuriniemi, S. (2004): *Homological computations in electromagnetic modeling*. PhD Thesis, Tampere University of Technology.

Tarhasaari, T.; Kettunen, L.; Bossavit, A. (1999): Some realizations of a discrete Hodge operator: a reinterpretation of finite element techniques. *IEEE Trans. on Mag.*, vol. 35, pp. 1494–1497.

Tonti, E. (1975): *On the formal structure of physical theories*. Quaderni dei Gruppi di Ricerca Matematica del CNR.

Tonti, E. (1998): Algebraic topology and computational electromagnetism. *Fourth International Workshop on the Electric and Magnetic Fields (EMF): from Numerical Models to Industrial Applications, Marseille, France*, pp. 284–294.

Weiland, T. (1977): A numerical method for the solution of the eigenvalue problem of longitudinally homogeneous waveguides. *Electronics and Communication (AE)*, vol. 31, pp. 308–311.