Contents lists available at SciVerse ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

A novel technique for cohomology computations in engineering practice

Paweł Dłotko [a,1], Ruben Specogna [b,*]

[a] Jagiellonian University, Institute of Computer Science, Lojasiewicza 4, 30348 Kraków, Poland
[b] Università di Udine, Dipartimento di Ingegneria Elettrica, Gestionale e Meccanica, Via delle Scienze 208, 33100 Udine, Italy

ARTICLE INFO

Article history:
Received 14 February 2012
Received in revised form 13 June 2012
Accepted 9 August 2012
Available online 24 August 2012

Keywords:
(Co) homology computation
(Co) homology generators
Definition of potentials
Eddy-currents
Thick cuts

ABSTRACT

The problem of computing cohomology generators of a cell complex is gaining more and more interest in various branches of science ranging from computational physics to biology. Focusing on engineering applications, cohomology generators are currently used in computer aided design (CAD) and in potential definition for computational electromagnetics and fluid dynamics.

The aim of this paper is to introduce a novel technique to effectively compute cohomology generators focusing on the application involving the potential definition for *h*-oriented eddy-current formulations. This technique, which has been called *Thinned Current Technique* (TCT), is completely automatic, computationally efficient and general. The TCT runs in most cases in linear time and exhibits a speed up of orders of magnitude with respect to the best alternative documented implementation.

1. Introduction

The problem of computing cohomology generators [1–5] of a cell complex is gaining more and more interest in various branches of science ranging from computational physics [6–23] to biology [24] and quantum chemistry [25]. Focusing on engineering, cohomology generators are currently used in computer aided design (CAD) for feature detection [26,27], shape analysis [28,29], parametrization and mesh generation [30,31] and in definition of potentials for magnetostatic, eddy-current problems [6–23] and for Navier–Stokes equations [16]. In particular, in the last 25 years, cohomology computation draw sensible attention in the computational electromagnetics community. In fact, generators of the first cohomology group—usually called *thick cuts* [10–23] in this context—are expressly needed to solve eddy-current problems with the efficient magnetic scalar potential-based T-Ω formulation [11,17,18,20,23]. Even though a considerable effort has been invested by the computational electromagnetics community to develop fast and general algorithms to produce cohomology generators, the required computational complexity and memory consumption render them still far from being attractive.

In mathematics and computer science communities, computations of cohomology groups and generators has not been extensively explored so far. As far as we know, the only field of mathematics where cohomology group has been computed (with cup and cap products [32,5]) is group theory [33,34]. As for the case of computing cohomology of cell complexes the only approach known to the Authors has been by using the concept of graph pyramids in [29]. However, this approach is limited to two-dimensional cubical complexes only. The lack of results concerning cohomology computations—in contrast with the great number of publications concerning homology computations—could be due to several reasons. One may be that cohomology theory is less intuitive and render finding novel applications more difficult.

A general algorithm for the computation of a basis of the cohomology group over integers is well known since many decades ago and it is based on the *Smith Normal Form* [1–5] computation. The problem of this algorithm is that its computational complexity is hyper-cubical with the best implementation available [35] and consequently it cannot be used in practice even on extremely coarse meshes. The same problem in case of homology computations has been solved by using a sparse matrix data structure [36] and *reductions techniques* [37]. Sparse matrices are used to effectively store boundary matrices and reduction techniques are used to reduce the complex before the Smith Normal Form computation is run. A survey of these reduction techniques in the context of computational electromagnetics is addressed in the review paper [38]. One can in fact use Smith Normal Form and some of the reductions designed for homology computations in order to obtain cohomology groups and their generators. 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

* Corresponding author. Tel.: +39 0432 558285; fax: +39 0432 558251.
E-mail addresses: dlotko@ii.uj.edu.pl (P. Dłotko), ruben.specogna@uniud.it (R. Specogna).
[1] P.D. was partially supported by grant IP 2010 046370.

so-called *pull-back* operation [37]. Pulling back the generators represents, however, a considerable extra computational cost. Therefore, 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 concept of shaving for cohomology has been introduced in [20]. In the same paper, the authors introduced an automatic, general, and efficient algorithm to compute cohomology generators in computational electromagnetics. The [20] algorithm uses as shavings the reduction procedures presented in [39,40].

An attempt of producing thick cuts avoiding (co) homology computations has been reported in [11]. First, the first cohomology group generators are found on the surface of conductors. However, only half of them have to be selected, namely the ones that are non-bounding in the conducting region. The question of how to select them automatically and efficiently is left unaddressed in the paper. Second, it grows an acyclic sub-complex inside the insulating region and, at the end of this process, the complement of the acyclic sub-complex is considered as the support of thick cuts. Again, how to construct the acyclic sub-complex is not described at all in the paper. The lack of these and other necessary details does not allow a serious analysis of this algorithm.

In [12], a different algorithm—which the Authors call *Generalized Spanning Tree Technique* (GSTT)—has been introduced. This algorithm attempts to generate a basis for the first cohomology group once a basis for the first homology group is given as input. In [12], homology generators have been constructed "by hand" and no discussion about the termination of the algorithm has been addressed. In [18], an efficient algorithm to automatically produce the homology generators has been described and in [21] a detailed analysis of the GSTT has been presented by the Authors. The conclusion is that the GSTT algorithm may present many termination failures that are very difficult to solve in practice.

We would like also to point out that an algorithm similar to the GSTT has been introduced by Kotiuga in the pioneering works [6–8,13]. More precisely, a GSTT-like algorithm is applied on the dual complex to obtain the so-called *thin cuts*, that are generators of the first cohomology group of the insulating domain on the dual complex. This technique is not attractive from the computational complexity point of view because a non-physical Poisson problem have to be solved for each generator to avoid cut self-intersections (i.e. the level sets of the solution of the each non-physical Poisson problem are embedded sub-manifolds).

Even using the best general and documented algorithm for cohomology computation [20], the timings are far from being satisfactory for everyday industrial problems, since the remaining part of the electromagnetic simulation with state-of-the-art implementations is frequently at least one order of magnitude faster than cohomology computation. Just to present an example, the spiral conductor represented in Fig. 10 requires less than a minute for solving the linear system with the CDICE code [41], while the computation of cohomology generators with state-of-the-art reductions needs at least one order of magnitude more time, (612 or 4040 s, as shown in Table 2, depending on the reduction technique employed.) The gap tends even to increase when one considers large scale problems. For this reason, there is the need for an automatic and general algorithm for cohomology computations which exhibits a sensible speed up, at least for typical configurations arising in engineering practice.

The novel contributions of this paper are: (a) a new thinning algorithm, that works only on top dimensional cells, (b) an adaptation of the ESTT algorithm [22] to be run without the need of the cell complex data structure, (c) Theorem 2 that provides a theoretical background for a novel algorithm for cohomology computations introduced in this paper called *Thinned Current Technique* (TCT) which is easy to implement in a parallel or distributed computing environment.

The paper is structured as follows. In Section 2 the need of a first cohomology group basis for electromagnetic potentials definition is recalled. In Section 3, we present a novel algorithm for cohomology computation called *Thinned Current Technique* (TCT), which stems from an alternative approach with respect to the ones presented in the literature survey. Section 4 describes the modifications that need to be applied when one wants to take advantage of some symmetry in the eddy-current problem. In Section 5, real-life examples are provided to benchmark the efficiency and robustness of the presented method with respect to the ones documented in literature. Finally, in Section 6, the conclusions are drawn.

2. (Co) homology in electromagnetic modeling

For the sake of brevity, the relevant concepts from algebraic topology as (co) chains, (co) boundaries and (co) homology are not recalled in this paper. For an informal introduction to the subject the reader is invited to consult [13,18,21]. A more formal presentation can be found in any algebraic topology textbook as [5].

Let us assume that the cell complex $\mathcal{K}$ provided by the input mesh of the computational domain is homeomorphic to the three-dimensional ball, which is a standard assumption in practical meaningful problems. Elements of $\mathcal{K}$ belonging to the conducting region are stored in the sub-complex $\mathcal{K}_c$, whereas the elements of $\mathcal{K}$ belonging to the insulating region form the sub-complex $\mathcal{K}_a$. Moreover, we assume, that $\mathcal{K}, \mathcal{K}_a$ and $\mathcal{K}_c$ are combinatorial manifolds (see [42]).

In this section, we intuitively explain why cohomology theory is expressly needed in the context of computational electromagnetics. A more comprehensive and detailed treatment on this topic is presented in [23]. To this aim, let us concentrate on the definition of potentials in the insulating region $\mathcal{K}_a$ for h-oriented eddy-current formulations.[2] The discrete Ampère's law in the insulating region can be written as

$$\delta \boldsymbol{F} = \boldsymbol{I} = \boldsymbol{0}, \tag{1}$$

where $\boldsymbol{I}$ is the complex-valued electric current 2-cochain—being zero by hypothesis since $\mathcal{K}_a$ models the insulating region—and $\boldsymbol{F}$ is the magneto-motive force (mmf) complex-valued 1-cochain. Thanks to the discrete Ampère's law, $\boldsymbol{F}$ is a 1-cocycle in $\mathcal{K}_a, (\boldsymbol{F} \in Z^1(\mathcal{K}_a, \mathbb{C}))$. Consequently, the 1-cocycle $\boldsymbol{F}$ can be expressed as a linear combination of a basis of the first cohomology group $H^1(\mathcal{K}_a, \mathbb{C})$ plus a 1-coboundary $B^1(\mathcal{K}_a, \mathbb{C})$. However, the basis $H^1(\mathcal{K}_a, \mathbb{C})$ in this case can be obtained form a basis of $H^1(\mathcal{K}_a, \mathbb{Z})$ where the elements of $\mathbb{Z}$ are treated as elements of $\mathbb{C}$.[3] Later in the paper the homology and cohomology groups are considered over integers if the coefficient group is not specified. The 1-coboundary

[2] 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 the whole paper in place of the group of complex numbers without any further changes.

[3] Instead of giving the formal argument, let us just present the very idea. Since $\mathcal{K}_a$ is embedded in $\mathbb{R}^3$, the homology groups are torsion free. Therefore, from the Universal Coefficient Theorem for homology (see Th. 3.A [43]), there is a bijective correspondence of generators of $H_1(\mathcal{K}_a, \mathbb{C})$ and generators of $H_1(\mathcal{K}_a, \mathbb{Z})$. Therefore, from the Universal Coefficient Theorem for cohomology (see Th. 3.1 [43]), there is a bijective correspondence between generators $H^1(\mathcal{K}_a, \mathbb{C})$ and $H_1(\mathcal{K}_a, \mathbb{Z})$. Again from the Universal Coefficient Theorem for cohomology, there is a bijective correspondence between generators of $H^1(\mathcal{K}_a, \mathbb{Z})$ and $H_1(\mathcal{K}_a, \mathbb{Z})$. Putting those two together we get the bijective correspondence between generators of $H^1(\mathcal{K}_a, \mathbb{Z})$ and $H^1(\mathcal{K}_a, \mathbb{C})$. We are using cohomology computed over integers since they—unlike complex and real cohomologies—can be rigorously computed with a computer.

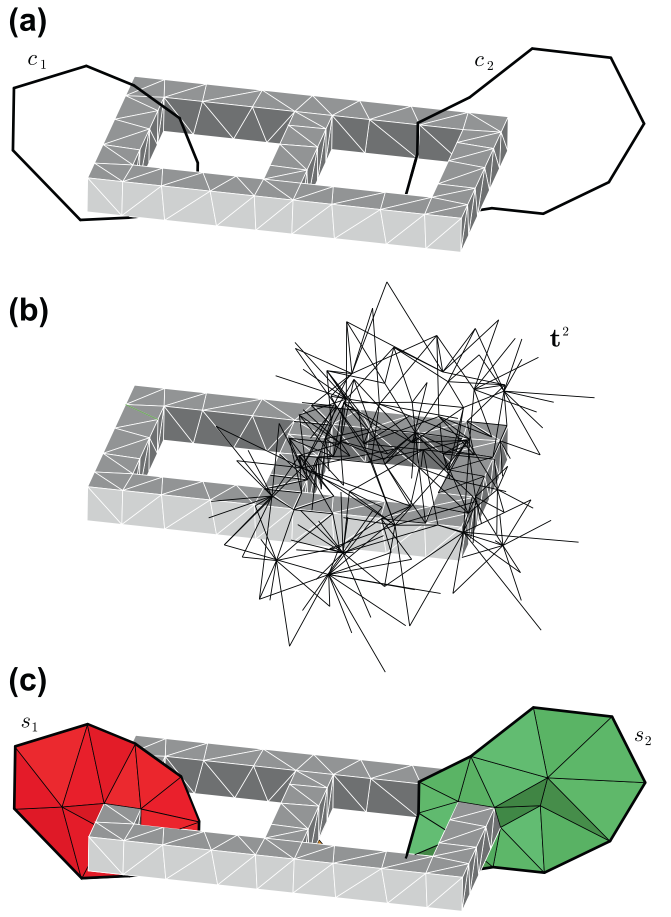

Fig. 1. (a) Two representatives c_1 and c_2 of $H_1(\mathcal{K}_a)$ group generators. (b) The support of the $H^1(\mathcal{K}_a)$ generator $\boldsymbol{t}^2$ dual to the homology generator c_2. (c) Two possible 2-chains $s_1, s_2 \in C_2(\mathcal{K})$ whose boundaries are c_1 and c_2, respectively.

$B^1(\mathcal{K}_a, \mathbb{C})$ is provided by taking the 0-coboundary of a complex-valued 0-cochain $\boldsymbol{\Omega}$ usually called *magnetic scalar potential*. We want to point out that $\boldsymbol{\Omega}$ is not unique and depends on the chosen first cohomology group generators and on *gauging* (i.e. fixing the value of $\boldsymbol{\Omega}$ in one arbitrary node). Hence, one has

$$\boldsymbol{F} = \delta\boldsymbol{\Omega} + \sum_{j=1}^{\beta_1(\mathcal{K}_a)} i_j \boldsymbol{t}^j, \tag{2}$$

where the $\{\boldsymbol{t}^j\}_{j=1}^{\beta_1(\mathcal{K}_a)}$ are the representatives of the first cohomology group $H^1(\mathcal{K}_a)$ generators over integers and $\beta_1(\mathcal{K}_a)$ is the first Betti number of the $\mathcal{K}_a$ sub-complex. By a little abuse of notation, when it is not confusing, by cohomology generators we refer to both the cohomology classes and their representatives. The coefficients $\{i_j\}_{j=1}^{\beta_1(\mathcal{K}_a)}$ of the linear combination, called *independent currents*, are defined in the following.

Example 1. In the present and following sections, to help the exposition, we are going to use as an example a conductor which is formed by a connected sum of two tori. This conductor, that form the conducting region $\mathcal{K}_c$, is surrounded by the insulating region $\mathcal{K}_a$. The conductor is represented in gray in Fig. 1a, while $\mathcal{K}_a$ is not depicted for the sake of clarity. In the same picture, two possible representatives c_1 and c_2 of the $H_1(\mathcal{K}_a)$ homology generators for the complement of the conductor are represented. In Fig. 1b, the support of the $\boldsymbol{t}^2$ cohomology generator dual to the homology generator c_2 is shown.

Let us fix the $H^1(\mathcal{K}_a)$ generators in (2). In [44] it has been shown that in case of complexes embedded in $\mathbb{R}^3$ there is a straightforward correspondence between cohomology and homology group generators. Namely, for $\{\boldsymbol{t}^j\}_{j=1}^{\beta_1(\mathcal{K}_a)}$ being the representatives of the fixed $H^1(\mathcal{K}_a)$ cohomology group basis, there exists a set of cycles $\{c_i\}_{i=1}^{\beta_1(\mathcal{K}_a)}$ being the representatives of a $H_1(\mathcal{K}_a)$ homology group basis such that $\langle \boldsymbol{t}^j, c_i\rangle = \delta_{ij}$ hold, where by $\langle \cdot, \cdot \rangle$ we denote the dot product of a cochain on a chain.[4] Since the 1-cycles $\{c_i\}_{i=1}^{\beta_1(\mathcal{K}_a)}$ are $H_1(\mathcal{K}_a)$ generators and $\mathcal{K}$ is homologically trivial there exists 2-chains $\{s_i\}_{i=1}^{\beta_1(\mathcal{K}_a)} \in C_2(\mathcal{K})$ whose boundaries are the 1-cycles $\{c_i\}_{i=1}^{\beta_1(\mathcal{K}_a)}$. Moreover, the 2-chains $\{s_i\}_{i=1}^{\beta_1(\mathcal{K}_a)} \in C_2(\mathcal{K})$ intersect $\mathcal{K}_c$. Let us consider the exact sequence of the pair $(\mathcal{K}, \mathcal{K}_a)$ [43]:

$$\cdots \to H_2(\mathcal{K}_a) \to H_2(\mathcal{K}) \to H_2(\mathcal{K}, \mathcal{K}_a) \to H_1(\mathcal{K}_a) \to \cdots.$$

Since $H_2(\mathcal{K}) = 0$, there is an isomorphism between $H_2(\mathcal{K}, \mathcal{K}_a)$ and $H_1(\mathcal{K}_a)$ induced by the boundary map. From the excision property [43], $H_2(\mathcal{K}, \mathcal{K}_a)$ is isomorphic to $H_2(\mathcal{K}_c, \partial\mathcal{K}_c)$. Therefore, the restrictions $\{\sigma_i\}_{i=1}^{\beta_1(\mathcal{K}_a)}$ of the 2-chains $\{s_i\}_{i=1}^{\beta_1(\mathcal{K}_a)}$ to $\mathcal{K}_c$ are generators of the $H_2(\mathcal{K}_c, \partial\mathcal{K}_c)$ homology group.

The current linked by the 1-cycle c_i is defined as the dot product of the current 2-cocycle $\boldsymbol{I}$ and a 2-chain[5] $s_i \in C_2(\mathcal{K})$ whose boundary is c_i, see Fig. 1b,

$$i_j = \langle \boldsymbol{I}, s_j \rangle. \tag{3}$$

The independent currents $\{i_j\}_{j=1}^{\beta_1(\mathcal{K}_a)}$ are defined as the currents linked by the $H_1(\mathcal{K}_a)$ generators $\{c_i\}_{i=1}^{\beta_1(\mathcal{K}_a)}$. The independent currents have been defined in (3) as the currents flowing through the 2-chains $\{s_i\}_{i=1}^{\beta_1(\mathcal{K}_a)}$. Since the contribution to the current is zero in $\mathcal{K}_a$, the independent currents as defined in (3) match the currents that flow in the branches of the conductor identified by the $H_2(\mathcal{K}_c, \partial\mathcal{K}_c)$ generators $\{\sigma_i\}_{i=1}^{\beta_1(\mathcal{K}_a)}$ previously introduced. This gives a physical interpretation of complex numbers $\{i_j\}_{j=1}^{\beta_1(\mathcal{K}_a)}$ used in (2). An independent current may be known in some cases (i.e. a current-driven coil), but in general they are extra unknowns to be determined by solving the eddy-current problem. In the first case, it is clear that by setting the value of an independent current one can impose a desired value of current in the branch of the conductor identified by the corresponding $\{\sigma_i\}_{i=1}^{\beta_1(\mathcal{K}_a)}$.

3. The thinned current technique (TCT)

In this section a novel technique, referred to as *Thinned Current Technique* (TCT), to compute cohomology generators of the first integer cohomology group of the insulating region sub-complex $H^1(\mathcal{K}_a)$ is introduced. The TCT stems from an original approach with respect to the ones presented in the literature survey in the Introduction. In particular, all the proposed techniques to compute the $H^1(\mathcal{K}_a)$ generators work directly on the $\mathcal{K}_a$ sub-complex. The novel idea of the proposed approach is to put the emphasis on the sub-complex $\mathcal{K}_c$ taking advantage—as it is described further in the paper—of the feature that the whole complex $\mathcal{K}$ is homologically trivial.

The idea behind the algorithm has a physical root. In fact, let us "compress" the conductors as much as possible. At the end of this thinning process, let us imagine the same independent currents of the original configuration that flows in "tubes" formed by pillars of tetrahedra, see Fig. 2a to visualize the thinned conductive region in the proposed example. The skeleton of these pillars on the dual complex [5,18] is a graph, see Fig. 2b. Since the linked currents in the case

[4] We want to point out that the dot product of a cochain on a chain is a discrete analog of integration.

[5] The 2-chain s_i is not unique (it is defined up to its boundary c_i) but, since the current 2-cochain $\boldsymbol{I}$ is a 2-cocycle, the linked current does not depend on the particular 2-chain s_i used in the dot product, see [23]. We want to point out that s_i is a 2-chain in the whole (homologically trivial) complex $\mathcal{K}$.

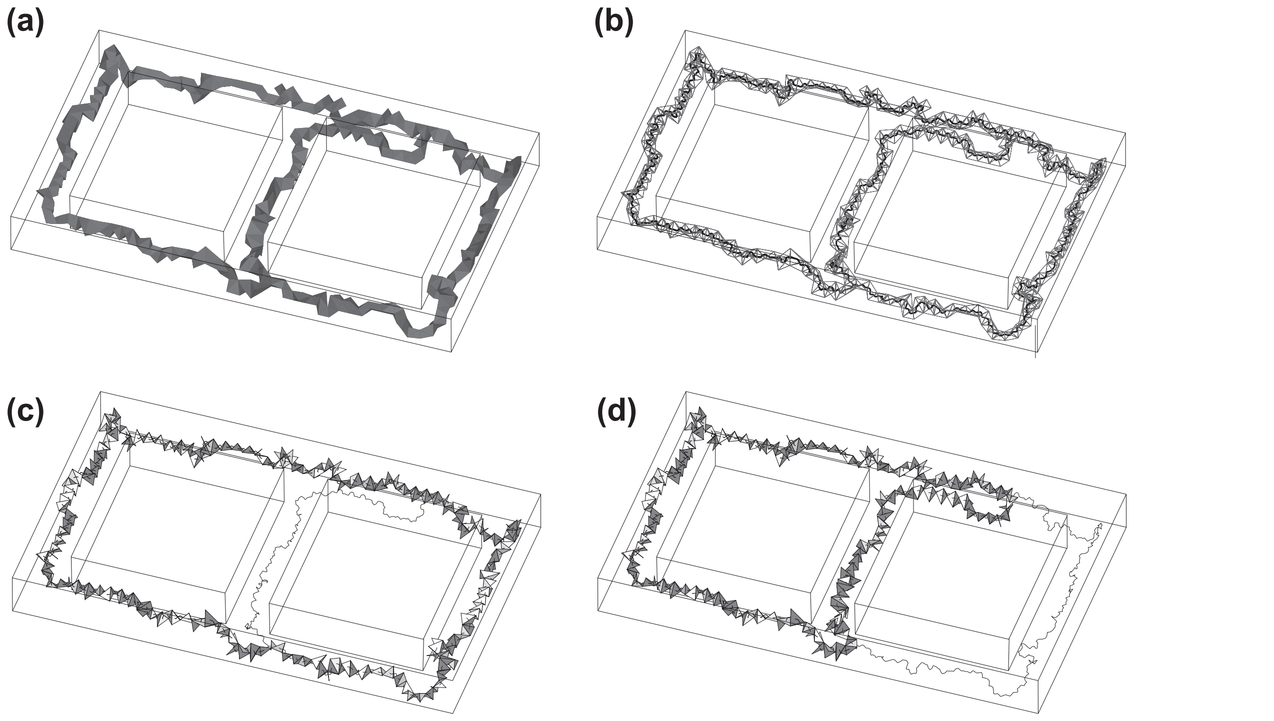

Fig. 2. (a) The pillars of tetrahedra obtained after the thinning process on $\mathcal{K}_c$. (b) The skeleton of $\mathcal{K}_c$ on the so called dual complex, which pierces the pillars of the tetrahedra. (c) and (d) The two $\mathbf{c}^1$ and $\mathbf{c}^2$ *independent cycles.*

of thinned conductors are the same as the ones in the original configuration by hypothesis, it is clear that even though the current distribution inside $\mathcal{K}_c$ is different, cohomology generators relative to the new current distribution restricted to $\mathcal{K}_a$ are cohomology generators for the original configuration (see Section 3.8 for the proof).

In the following the TCT algorithm is described in detail.

3.1. Data structures

The key feature of the presented Fortran 90 implementation of the TCT algorithm is that it does not require the construction of the whole 0-, 1- and 2-dimensional data structure needed to store the cell complex.[6] We assume that the nodes of the tetrahedra are enumerated with integers by the mesh generator. Then, each tetrahedron is an ordered vector of the indices of its four nodes. The increasing ordering of element indices fixes its orientation [5,20]. The algorithms operate in most cases on the list L of tetrahedra, where $L = L_c \cup L_a$. L_c denotes the tetrahedra in $\mathcal{K}_c$, whereas L_a denotes tetrahedra in $\mathcal{K}_a$. Moreover, both conducting and insulating regions are modeled in the solid modeling sense as combinatorial manifolds with boundaries, i.e. a link[7] of every vertex is a sphere or semi-sphere of dimension two.

3.2. Connected components of $\mathcal{K}_c$

Algorithm `findConnectedComponent` finds the connected components of $\mathcal{K}_c$ once the list L_c is given as input. Let H be a hash table with integer keys. Let us assume that $H[i], i \in \mathbb{N}$, is the list of all tetrahedra possessing the node i. This hash table, which can be trivially constructed in linear time, is used to efficiently find the neighbors of a given tetrahedron. For practical meshes the length of every list in the hash table H is constant. The set `tetrahedraAlreadyAssigned` in the line 2 of the `findConnectedComponent` Algorithm can be also coded as a boolean vector of length as the number of cells. In that case we can assume that the instructions at lines 4 and 13 of the algorithm take $O(1)$ time. The `while` loop starting at line 8 and ending at line 15 takes a time proportional to the cardinality of `cc`. Therefore, it is clear that the whole `for` loop at line 3 requires $O(card(L_c))$ time.

Algorithm 1. `findConnectedComponent`

```
Ensure: vector of vectors of tetrahedra in connected
   components of $L_c$
Require: list $L_c$
 1: vector of vector of tetrahedra result;
 2: set tetrahedraAlreadyAssigned = $\emptyset$;
 3: for $i = 0$ to size of $L_c$
 4:   if $L_c[i] \in$ tetrahedraAlreadyAssigned then
 5:     continue;
 6:   vector of tetrahedra cc;
 7:   queue $Q$
 8:   Enqueue($Q, L_c[i]$);
 9:   while $Q$ is not empty
10:     tetrahedra $T := pop(Q)$;
11:     tetrahedraAlreadyAssigned $\leftarrow$
   tetrahedraAlreadyAssigned $\cup T$;
12:     cc $\leftarrow$ cc $\cup T$;
13:     for every tetrahedron $T' \in L_c$ such that $T'$ is a neighbor
   (use $H$ hash table to determine neighbors of $T$) of $T$ do
14:       if $T' \notin$ tetrahedraAlreadyAssigned then
15:         enqueue ($Q, T'$);
16:   result $\leftarrow$ result $\cup$ cc;
17: return result;
```

[6] If the cohomology computations are not required.

[7] Let us consider all maximal elements $T_1, \ldots, T_s$ containing a given node n. The link of n is the set of all nodes, edges and faces of $T_1, \ldots, T_s$ that do not contain n.

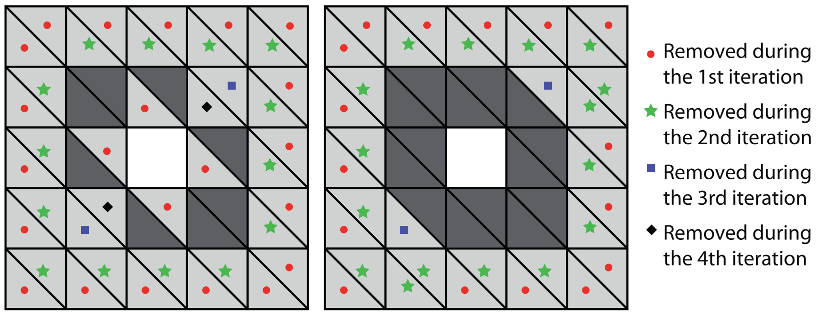
Removed during the 1st iteration
Removed during the 2nd iteration
Removed during the 3rd iteration
Removed during the 4th iteration

Fig. 3. A 2-dimensional conductor $\mathcal{K}_c$ is shown in grey. For the sake of clarity, the insulator $\mathcal{K}_a$ is not shown. In real 3-dimensional cases, a set of tetrahedra should be considered in place of the triangles. On left and right, 2-dimensional examples of the two strategies are shown. On the left, the result of the thinning when acyclicity of the intersection of triangles with the conductor is checked. It is clearly not a skeleton. On the right, the result of the thinning when acyclicity of the intersection of triangles with the insulating region is checked. The markers on triangles denote which triangles are removed during each iteration of the algorithm.

3.3. Skeletonization of the conductor

Let us first define formally the concept of skeleton of a complex. For this purpose the concept of dual complex is required. The formal definition can be found in [18]. In short, elements dual to tetrahedra A are the vertices $D(A)$ of the dual complex. A triangle B between two tetrahedra A_1, A_2 correspond to an edge of the dual complex $D(B)$ joining $T(A_1)$ and $T(A_2)$. An edge E incidental to triangles $T_1, \ldots, T_n$ correspond to a dual face $D(E)$ bounded by $D(T_1), \ldots, D(T_n)$. Finally, a vertex V incidental to edges $E_1, \ldots, E_n$ corresponds to dual volume $D(V)$ bounded by $D(E_1), \ldots, D(E_n)$. We want to point out that the complex dual to a simplicial complex is not a simplicial complex.

In this paper by a skeleton of a complex we mean a sequence of tetrahedra whose dual complex is a retract of the support of the initial simplicial complex. Moreover, we require that each tetrahedron in the skeleton, with the exception of tetrahedra with faces on the external boundary of $\mathcal{K}$, has at least two neighboring tetrahedra (i.e. there are no "hanging" tetrahedra). We also assume that the complex dual to the skeleton is a graph. This definition, at a first glance, may seem hard to check algorithmically. However, later in the paper we show that in most practical cases such a skeleton can be easily obtained. In Fig. 3 two possible homotopical retracts of the whole conductor are depicted as dark triangles. Only the one on the right is a valid skeleton according to the presented definition.

Let us now present the Algorithm 2 to *skeletonize* a conductor. Standard skeletonization procedures can be found in many papers [19]. For instance, one can produce a skeletonization by using coreduction [40]. However, we do not want to store the global structure of simplicial complex for the sake of maximum efficiency, therefore, a novel skeletonization technique is sought. In this paper we introduce a novel effective one based on the acyclicity test tables introduced in [20] and presented in [48].

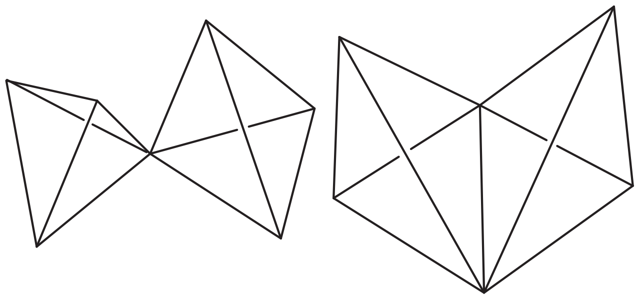

Fig. 4. On the left, two tetrahedra intersecting in a node. On the right, two tetrahedra intersecting in an edge.

Algorithm 2. `Skeletonization`

```
Ensure: List of tetrahedra in the skeleton of the conductor
  result and a boolean array computationStatus (the
  value on ith position denotes if the skeletonization process
  was successful for ith connected component.)
Require: vector of vectors cc of tetrahedra in connected
  components of $\mathcal{K}_c$
 1: boolean[size of cc] computationStatus = true;
 2: vector of vectors of result;
 3: for i = 1 to size of cc do
 4:   set removedTets = $\emptyset$;
 5:   list candidatesToRemove = $\emptyset$;
 6:   for every tetrahedron $T \in$ cc[$i$] do
 7:     if $T$ has faces not shared by other elements in cc[$i$] then
 8:       candidatesToRemove ← $T$;
 9:   while true do
10:     boolean wasTetrahedronRemoved = false;
11:     for every tetrahedron $T \in$ candidatesToRemove do
12:       Let $f$ be the set of all nodes, edges and faces of $T$
  that belong to the external boundary of cc[$i$] or
  removedTets;
13:       if $f$ is an acyclic configuration (Testing acyclicity of
  a configuration is based on [48]. The procedure is described
  in [20].) then
14:         removedTets ← $T$;
15:         wasTetrahedronRemoved = true;
16:     if wasTetrahedronRemoved = false then
17:       break;
18:     candidatesToRemove = $\emptyset$;
19:     for every tetrahedron $T \in$ cc[$i$]—$T \notin$ removedTets do
20:     if $T$ has faces not shared by other elements in cc[$i$] or
  faces shared by elements in removedTets then
21:         candidatesToRemove ← $T$;
22:   vector of tetrahedra skeleton ← {$T \in$ cc [$i$]—
  $T \notin$ removedTets};
23:   boolean wasSkeletonizationSuccessfull = true;
24:   for every tetrahedron $T \in$ result do
25:     for every edge $E \in T$ do
26:       if all tetrahedrons incidental to $E$ in $\mathcal{K}$ are in
  skeleton then
27:         wasSkeletonizationSuccessfull = false;
28:   if wasSkeletonizationSuccessfull then
29:     result ← skeleton;
30:     computationStatus[$i$] = true;
31:   else
32:     computationStatus[i] = false;
33: return result, computationStatus;
```

As one can see, Algorithm 2 may not provide a valid skeleton for some conductor, which is consequently marked by setting the flag (corresponding to the number of the connected component for which skeletonization fails) in `computationStatus` vector to `false`. In fact, in quite rare cases the skeletonization procedure may end up in some configuration similar to the Bing's House [45] or it may not be possible to find the skeleton of the conductor when the conductor cannot be retracted to a graph.[8] However, when the `Skeletonization` algorithm returns `false` at ith position of the `computationStatus` vector, one may use the general algorithm presented in [20] to compute the $H^1(\mathcal{K} \setminus \mathcal{K}_c^i)$ cohomology group generators, where $\mathcal{K}_c^i$ is the sub-complex with the elements of $\mathcal{K}_c$ belonging to the considered ith connected component. We emphasize that the computation is performed for the complement of the considered connected component $\mathcal{K}_c^i$ (it suffices to treat all remaining conductors as part of the complement). After the computations, the obtained cocycles representing the first cohomology group basis of $\mathcal{K}_c^i$ complement restricted to $\mathcal{K}_a$ are—as demonstrated in the following—cohomology generators of $H^1(\mathcal{K}_a)$.

Let us call by *thinned conductor* the output of `Skeletonization` Algorithm. We show that the obtained thinned conductor is a skeleton of the conductor. First, let us show that the homology of the thinned conductor obtained by the `Skeletonization` Algorithm is preserved with respect to $\mathcal{K}_c$. At a fixed stage of the algorithm execution, let us denote by $\hat{\mathcal{K}}_a$ the union of the insulating domain $\mathcal{K}_a$ with `removedTets`. In the same spirit, $\hat{\mathcal{K}}_c$ denotes tetrahedra in the conductor $\mathcal{K}_c$ that are not in `removedTets`. It is clear from the Mayer–Vietoris theorem [43] that in the line 2 and the succeeding line of the `Skeletonization` algorithm we ensure that homology of $\hat{\mathcal{K}}_a$ does not change. To show this, let us write a Mayer–Vietoris sequence for reduced homology for the pair $(\hat{\mathcal{K}}_a, T)$, where T is a tetrahedron for which the test in the line 2 returns true (i.e. T has an acyclic intersection with $\hat{\mathcal{K}}_a$):

$$\cdots \to \widetilde{H}_n(\hat{\mathcal{K}}_a \cap T) \to \widetilde{H}_n(\hat{\mathcal{K}}_a) \oplus \widetilde{H}_n(T) \to \widetilde{H}_n(\hat{\mathcal{K}}_a \cup T) \to \cdots.$$

Due to the line 2 of the algorithm, $\widetilde{H}_n(\hat{\mathcal{K}}_a \cap T) = 0$. Therefore, we have an isomorphism between $\widetilde{H}_n(\hat{\mathcal{K}}_a) \oplus \widetilde{H}_n(T)$ and $\widetilde{H}_n(\hat{\mathcal{K}}_a \cup T)$. Since $\widetilde{H}_n(T) = 0$, we get an isomorphism between $\widetilde{H}_n(\hat{\mathcal{K}}_a)$ and $\widetilde{H}_n(\hat{\mathcal{K}}_a \cup T)$. As a consequence, one can add T to $\hat{\mathcal{K}}_a$ without changing the homology of $\hat{\mathcal{K}}_a$.

Due to the test in line 2 of the algorithm, we also know that $H_n(\hat{\mathcal{K}}_a, \partial\hat{\mathcal{K}}_a) = H_n(\mathcal{K}, \hat{\mathcal{K}}_c)$ remains unchanged. The rest is a simple consequence from the exact sequence of the pair $(\mathcal{K}, \hat{\mathcal{K}}_c)$ [43]:

$$\cdots \to H_n(\hat{\mathcal{K}}_c) \to H_n(\mathcal{K}) \to H_n(\mathcal{K}, \hat{\mathcal{K}}_c) \to H_{n-1}(\hat{\mathcal{K}}_c) \to \cdots.$$

Since $H_n(\mathcal{K})$ is homologically trivial, we have an isomorphism between $H_n(\mathcal{K}, \hat{\mathcal{K}}_c)$ and $H_{n-1}(\hat{\mathcal{K}}_c)$. Therefore, the topology of $\hat{\mathcal{K}}_c$ is also preserved (with respect to $\mathcal{K}_c$) during skeletonization.

Let us show that the thinned conductor is a skeleton. Let us first show two possible ways of obtaining two different, homologically equivalent skeletons. One can in fact check the acyclicity of the intersection of a tetrahedron with $\hat{\mathcal{K}}_c$ or with $\hat{\mathcal{K}}_a$. The two strategies applied to a simple example are visualized in Fig. 3. It is clear from Fig. 3 that, in order to obtain a skeleton as defined in this paper, one should perform the acyclicity test where the intersection of the tetrahedra with the insulating region is considered (as it is done in the loops 2 and 2 of the Algorithm). Let us demonstrate that in this case one always obtain a valid skeleton or a `false` value is returned in the corresponding position of the `computationStatus` vector.

We want to point out that the situation we consider is different from the acyclic subspace method considered in [39]. In [39] the relative homology of a space modulo an acyclic set is computed, while in this paper we are using a topological criterion to perform a topology preserving thinning of the connected components of $\mathcal{K}_c$.

In order to show that if `computationStatus`$[i]$ = `true` the complex dual to the thinned conductor obtained by the skeletonization of $\mathcal{K}_c^i$ is one dimensional (i.e. a graph), let us analyze the `for` loop at line 2 of the algorithm. If the test fails for an edge E, the 2-dimensional cell $D(E)$ would appear in the dual complex. It is clear that, if the test passed, neither 2- nor 3-dimensional elements can occur in the dual complex. Therefore, let us now assume that the algorithm does not return `false` at any position of the `computationStatus` vector. We give a rather informal argument why hanging tetrahedra cannot occur in this setting. By doing a simple case study, one can easily be convinced that if a tetrahedra T is a hanging tetrahedron, then the intersection of the boundary of T with the thinned conductor complement is acyclic. Therefore, T should have been removed in the `while` loop at the line 2 of the algorithm. It remains to show that the tetrahedra in the thinned conductor intersect on a common face or do not intersect at all. To show this, a simple and intuitive case study is presented in Fig. 4. On the left, let us consider a situation where two tetrahedra of the thinned conductor touch in a node. We show that this situation cannot occur. In the initial mesh there was another tetrahedron incidental to this node (since we started from a combinatorial manifold, and the presented configuration cannot occur in combinatorial manifolds.) Let us consider the first tetrahedron T incidental to this node removed in the course of the `Skeletonization` algorithm whose removal causes this configuration not to be a combinatorial manifold. Checking the acyclicity gives the information whether the topology of the complement does not change. However, removing T changes the topology of the complement. This is because T is closed and its removal changes either β_0 or β_1 of the conductor. Therefore, the suitable Betti number of the complement is changed (as a consequence of exact sequence of the pair), that gives a contradiction. Exactly the same situation holds for the situation represented on the right of Fig. 4. Again, the situation on the right of Fig. 4 cannot occur in combinatorial manifolds, and the same idea can be used to show that this situation cannot happen.

This intuitive demonstration shows that indeed a skeleton is obtained as a result of the `Skeletonization` algorithm. A formal proof, due to technical difficulties in the exposition, will be published elsewhere.

3.4. Minimal independent cycles in the skeleton and integer 2-cocycle construction

The next step of the TCT algorithm, when the skeletons of conductors are provided, is to find a suitable cycle basis. In many cases minimal length cycle basis of the obtained graph suffices. An algorithm to find the minimal cycle basis in a graph can be found in [50]. The time complexity of the algorithm is $O(m^2n)$, where m is the number of edges and n is the number of nodes in the graph. The intuition is that the numbers m and n for a skeleton should be very small comparing to the cardinality of $\mathcal{K}$.

For simplicity, in this paper we use a simple technique based on a tree-cotree decomposition of the skeleton to obtain a cycle basis. The two independent cycles on the dual complex obtained in the proposed example are represented in Fig. 2c and d.

Later, let us assume that the algorithm `Skeletonization` returns a graph G. For every connected component of G a set of cycles

[8] Unfortunately, we cannot give sufficient conditions for a tree dimensional complex to be skeletonizable. This is a consequence of the well-known problem of testing the contractibility of complexes. In two dimensions this problem can be solved in linear time. In dimension three this problem is at least as hard as finding optimal discrete Morse function which is an NP-hard problem, see [46] (a sequence of collapses on a contractible space gives an optimal Discrete Morse function). In dimension four and higher this problem is undecidable [49].

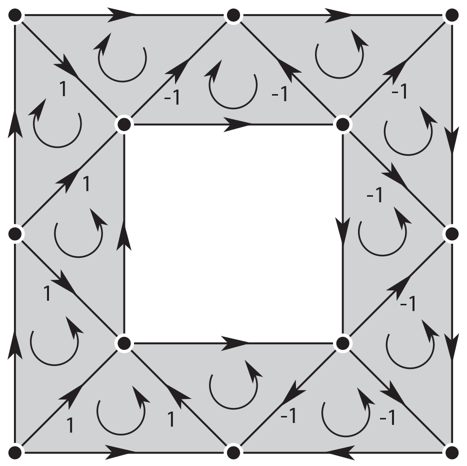

Fig. 5. A simple 2-dimensional example with a 1-dimensional analogous of the procedure. We want to obtain a 1-dimensional cocycle c from dual edges around the annulus. To do this, the coefficient of one edge in c is fixed. Then, the coefficients of all remaining edges are in such a way that the coboundary of the resulting cocycle is void (note that the depicted orientation have to be taken into account as it is done by using incidence index κ).

$\{c_1,\dots,c_n\}$ being a cycle basis of this connected component of G is obtained. Let us moreover interpret each 1-cycle c_i in the dual complex as a sequence of succeeding tetrahedra such that a pair of neighboring tetrahedra share a triangle. Each cycle needs to be oriented by using the standard incidence index κ for simplices (the obvious technical details are left to the Reader)

$$\kappa(A,B) := \left\{\begin{array}{ll} (-1)^i & \text{if } A=[a_1,\dots,a_{i-1},a_i,a_{i+1},\dots,a_n], \\ & \quad B=[a_1,\dots,a_{i-1},a_{i+1},\dots,a_n] \\ 0 & \text{otherwise} \end{array}\right\}.$$

Then, the Algorithm `CreateCocycles` is used to construct the vector of 2-cocycles $\{\boldsymbol{c}^i\}$. It is clear that the Algorithm `CreateCocycles` works in a linear time $O(card\ \mathcal{K})$. Orienting the dual cycle ensures that the output cochain is a cocycle. The idea of the procedure is presented in Fig. 5.

Algorithm 3. `CreateCocycles`

```
Ensure: Vector of vectors of pairs (triangle, value)
  representing the 2-cocycles
Require: Vector of vectors of tetrahedra, cycles representing
  the cycle basis
 1: Vector of vectors of pairs (triangle, value) result;
 2: for i = 1 to size of cycles do
 3:   vectors of pairs (triangle, value) cocycle;
 4:   Let $T_1$ be a triangle between first and last tetrahedron in
      cycles;
 5:   cocycle ← $(T_1, 1)$;
 6:   for $j$ = 1 to length of cycles[i] do
 7:     Let $T_1$ and $T_2$ be two triangles in cycles[$i$][$j$], let
        $(T_1, l_1) \in$ cocycle;
 8:     $l_2 := -\kappa($cycles[i][j]$, T_1) l_1 \kappa($cycles[i][j]$, T_2)$;
 9:     cocycle ← $(T_2, l_2)$;
10:   result ← cocycle;
11: return result;
```

3.5. ESTT algorithm

The *Extended Spanning Tree Technique* (ESTT) algorithm [22] is a general version of the so-called Spanning Tree Technique (STT) [12,21,22]. It works on the whole (homologically trivial) complex $\mathcal{K}$. Once a 2-cochain $\boldsymbol{c}^j$ obtained by the Algorithm `CreateCocycles` is provided at input, it finds a 1-cocycle $\boldsymbol{d}$ such that $\delta\boldsymbol{d} = \boldsymbol{c}^j$. The details of the ESTT algorithm, together with a proof of its generality, can be found in [22]. Concerning the time complexity of the ESTT algorithm, it is linear $O(\mathcal{K})$ in all practical cases, see [22].

The ESTT algorithm used in the following is different from the one presented in [22]. The version presented in this paper does not use the global data structure of a complex, which yields a more efficient implementation. It works only on the list of all tetrahedra in the complex $\mathcal{K}$. At first, we need to construct a spanning tree on $\mathcal{K}$. It is important to remind that the best strategy to forming a tree for our purpose is to use *minimal diameter trees* [21,22].

Algorithm 4. `ConstructSpanningTree`

```
Ensure: Vector of edges (i.e. pairs of node indices)
  representing a spanning tree of $\mathcal{K}$
Require: Vector of all tetrahedra in $\mathcal{K}$
 1: Vector of edges result;
 2: Let H be a hash table assigning to a given index $i$ a list of
    tetrahedra possessing a node $i$;
 3: Let vert be a boolean array, initially set to false. The
    value at position $i$ indicates if a node $i$ is already in the
    constructed tree;
 4: Let $T$ be a tetrahedron for which there exists a node $i \in T$
    such that vert [$i$] =false;
 5: vert [$i$] := true;
 6: Queue Q ← H[$i$];
 7: while Q $\neq \emptyset$ do
 8:   $T$ = pop (Q);
 9:   for Every node $j \in T$ do
10:     if vert [$j$] =true then
11:       continue;
12:     Let $k \in T$ be such that vert[$k$] is equal true;
13:     result ← $(j,k)$;
14:     vert [$j$] :=true;
15:     Q ← $H[j]$;
16: if There exists a vertex $i$ such that vert[$i$] is false then
17:   GOTO 4;
18: return result;
```

Theorem 1. *The algorithm* `ConstructSpanningTree` *constructs a spanning tree of edges of* $\mathcal{K}$.

Due to the condition 4, it is clear that all vertices are set to `true` in the vector `vert`. Therefore, due to the line 4 of the Algorithm, they are all connected in their connected components (although in the considered case complex $\mathcal{K}$ is connected, the `ConstructSpanningTree` Algorithm works in the general case, for non connected complexes.) To show that no cycle can occur in the output graph, let us suppose by contrary that the output graph contains a cycle c. Let an edge (k,l) be the last edge added to the cycle c. In the iteration of the `for` loop at the line 4 in which the edge (k,l) is added to the output, we have `vert`[k] = `vert`[l] = `true`. Therefore, the edge (k,l) cannot be added to the output due to the condition in the line 4 of the algorithm. We get a contradiction and complete the proof.

The remaining part of the TCT algorithm is the ESTT algorithm. The algorithm together with its detailed description can be found in [22]. In the following the raw idea of the algorithm is presented. ESTT algorithm for a given 2-cocycle $c \in C^2(\mathcal{K})$ produces a 1-cocycle $d \in C^1(\mathcal{K})$ such that for every 1-cycle $e \in Z_1(\mathcal{K})$ and 2-chain $f \in C_2(\mathcal{K})$ with $\partial f = e$[9] we have $\langle c,f\rangle = \langle d,e\rangle$. As it is shown in [22], it always works provided that a cocycle is given as input, which

[9] Such a chain f always exists, since $\mathcal{K}$ is homologically trivial.

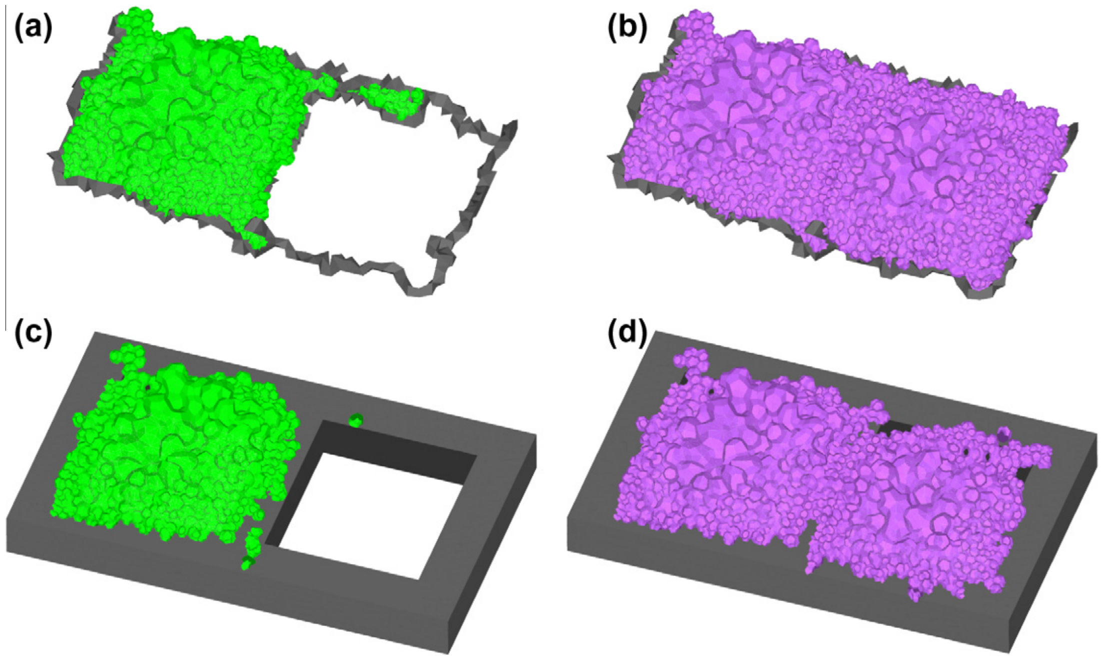

Fig. 6. (a) and (b) Dual faces dual to edges in the support of the cochains $\boldsymbol{d}^1$ and $\boldsymbol{d}^2$, respectively. (c) and (d) Dual faces dual to edges in the support of the cochains $\boldsymbol{t}^1$ and $\boldsymbol{t}^2$, respectively.

is the case of the TCT algorithm. The implementation required here is a little bit different with respect to the one presented in [22]. However, it is fairly easy by using hash tables to produce a list of all triangles in the complex $\mathcal{K}$. The details are left to the reader.

3.6. TCT algorithm

In this short section we basically put together all the intermediate algorithms in order to present the `TCT` algorithm.

Algorithm 5. `TCT`

```
Ensure: Vector of vectors of pairs (edge, value) representing
   1-cocycles (thick cuts)
Require: $\mathcal{K} = \mathcal{K}_c \cup \mathcal{K}_a$ list of tetrahedra, $\mathcal{K}_c$ conductive region,
   $\mathcal{K}_a$ insulating region
 1: Vector of vectors of pairs (edge, value) result,
    cohomologyGens;
 2: Vector of vectors of tetrahedra cc =
    findConnectedComponent($\mathcal{K}_c$);
 3: boolean array skeletonizationSucces;
 4: List of tetrahedra in skeletons of conductors skeleton;
 5: (skeleton,skeletonizationSucces) =
    Skeletonization(cc);
 6: if skeletonizationSucces[i] = false then
 7:    Compute first cohomology generators of $\overline{\mathcal{K} \setminus cc[i]}$ by
    using standard first cohomology group computations [20]
    and add the resulting cohomology generators to
    cohomologyGens;
 8: Vector of vectors of tetrahedra cycles – cycle basis of the
    graph cc;
 9: Vector of 2-cocycles cocycles = CreateCocycles
    (cycles);
10: result := ESTT (cocycles) + cohomologyGens;
11: Restrict supports of cocycles in result to $\mathcal{K}_a$;
12: return result;
```

Let $\boldsymbol{t}^1$ and $\boldsymbol{t}^2$ be the 1-cocycles obtained as output of TCT algorithm for the example considered in Fig. 1. The dual faces, dual to the edges in the support of the cochains $\boldsymbol{d}^1$ and $\boldsymbol{d}^2$ related with the example are represented in Fig. 6a and b, respectively. The corresponding cohomology $H^1(\mathcal{K}_a)$ generators $\boldsymbol{t}^1$ and $\boldsymbol{t}^2$ are represented in Fig. 6c and d.

Using only maximal elements (tetrahedra) in place of a simplicial complex data structure not only does not bring any performance penalty but speeds up the algorithm a lot and enables a dramatic reduction of memory usage. The timings of the presented algorithm are presented in Section 5.

3.7. Complexity analysis of the TCT algorithm

In this short section a summary of the results on the complexity analysis of TCT algorithm is given. First of all, we would like to point out that there is a huge difference between the typical and the worst case complexity estimates of this algorithm. This should not discourage the reader since dependence on heuristics is frequent in engineering and computer science. The worst case complexity analysis for the conjugate gradient method, optimization, and quicksort may be discouraging, while those algorithms perform perfectly well in practice.

The complexity of the TCT algorithm depends on the following factors:

1. Are we able to skeletonize the conductor?
2. How many extra symbolic variables (see [22]) are used during the ESTT algorithm run?

In a very unlucky case when the skeletonization process fail, we are in fact using pure cohomology computations as described in [20]. In this case the worst case computational complexity is $O(card(\mathcal{K}_a)^3)$.

If the skeletonization is performed, the ESTT algorithm may take $O(card(\mathcal{K}_a)^3)$ time in the worst case. That gives $O(card(\mathcal{K}_a)^3)$ as the theoretical worst case bound for the computational complexity of the algorithm. However, we would like to remark that those cases were never seen in practical computations provided BFS trees are used.

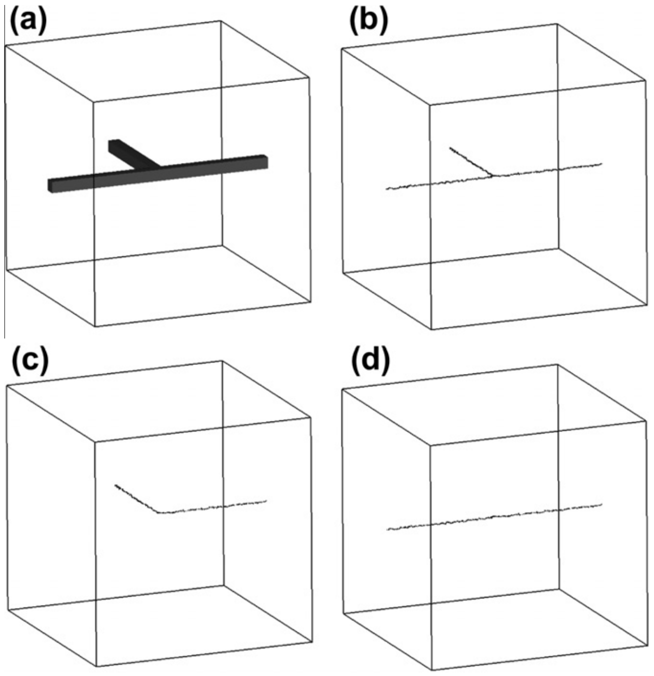

Fig. 7. An Y-shaped conductor.

Typically, in engineering practice, one should expect that the skeletonization algorithm does not fail and the ESTT algorithm takes $O(card(\mathcal{K}_a))$ time. Then, the typical overall complexity of the TCT algorithm is $O(card(\mathcal{K}))$, since:

(1) Finding connected components of $\mathcal{K}_c$ takes $O(card(\mathcal{K}_c))$.
(2) Skeletonization procedure takes $O(card(\mathcal{K}_c))$.
(3) Finding the cycle basis by using spanning tree technique and creating cocyles takes $O(card(\mathcal{K}_c))$.
(4) ESTT algorithm and restricting support of generators takes $O(card(\mathcal{K}))$.

No problem has been reported in the skeletonization procedure and no symbolic variable has been introduced for ESTT propagation in every of the benchmarks tested (that are much more than the ones reported in this paper.) Therefore, the linear time complexity behavior is visible in all tested examples.

3.8. Proofs

In this section we are going to show that the 1-cocycles obtained with the TCT algorithm are $H^1(\mathcal{K}_a)$ generators. First, we show that in the analysis it suffices to restrict to a complement of a single connected component of $\mathcal{K}_c$ in the standard cohomology computations. To be precise, (co) homology of a set consisting of many connected components are direct sum of (co) homologies of its components. Here the situation is different, since the set $\mathcal{K}_a$ whose cohomology is to be computed is connected, but its internal complements $\mathcal{K}_c = \mathcal{K}_c^1 \cup \ldots \cup \mathcal{K}_c^n$ are not. We show that it suffices to compute the cohomology of $\overline{\mathcal{K} \setminus \mathcal{K}_c^i}$ for $i \in \{1, \ldots, n\}$ and restrict the output to $\mathcal{K}_a$ to obtain a cohomology basis of $\mathcal{K}_a$.

Let $\{c_1^j, \ldots, c_{n_j}^j\}$ denote the $H^1(\overline{\mathcal{K} \setminus \mathcal{K}_a^j})$ generators restricted to $\mathcal{K}_a$. We have the following theorem:

Theorem 2. *$\{c_1^j, \ldots, c_{n_j}^j\}_{j=1}^n$ generates $H^1(\mathcal{K}_a)$.*

In order to avoid too many technicalities we give here the idea instead of the whole formal proof. First, let us show that a somehow similar property holds in case of homology generators. Let us take a single connected component $\mathcal{K}_c^i$. It is clear that one can find the representatives of generators of $H_1(\overline{\mathcal{K} \setminus \mathcal{K}_c^i})$ that lies on the boundary of $\mathcal{K}_c^i$. The raw intuition of this fact is the following: Let us take any representatives $c_1, \ldots, c_n \in C_1(\overline{\mathcal{K} \setminus \mathcal{K}_c^i})$ of generators of $H_1(\overline{\mathcal{K} \setminus \mathcal{K}_c^i})$. Since $\mathcal{K}$ is topologically trivial, one can find $b_1, \ldots, b_n \in C_2(\mathcal{K})$ such that $\partial b_i = c_i$ for $i \in \{1, \ldots, n\}$. Let us now take the cycles $\hat{c}_1, \ldots, \hat{c}_n \in C_1(\overline{\mathcal{K} \setminus \mathcal{K}_c^i})$ being intersections of $b_1, \ldots, b_n$ with the boundary of $\mathcal{K}_c^i$. They are clearly cycles, since $\partial\partial = 0$. They are also in the same class as c_i, so they form the desired set of $H_1(\overline{\mathcal{K} \setminus \mathcal{K}_c^i})$ generators.

Then, by using the exact sequence of the pair $(\mathcal{K}, \mathcal{K}_a)$ [43]

$$\cdots \xrightarrow{\partial_2} H_2(\mathcal{K}_a) \to H_2(\mathcal{K}) \to H_2(\mathcal{K}, \mathcal{K}_a) \xrightarrow{\partial_1} H_1(\mathcal{K}_a) \to \cdots,$$

since $\mathcal{K}$ is topologically trivial, we get that $\partial_1 : H_2(\mathcal{K}, \mathcal{K}_a) \to H_1(\mathcal{K}_a)$ is an isomorphism. Moreover, thanks to the excision property, we get

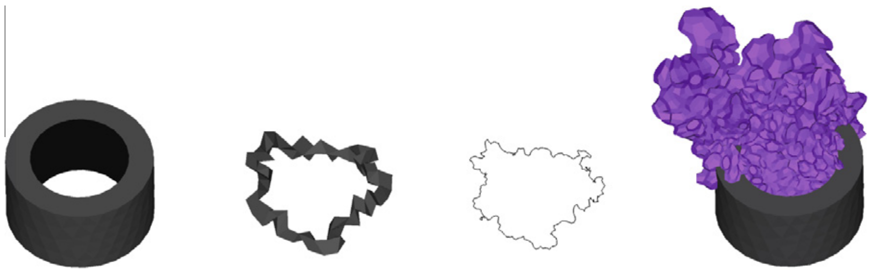
Fig. 8. b1: A circular coil.

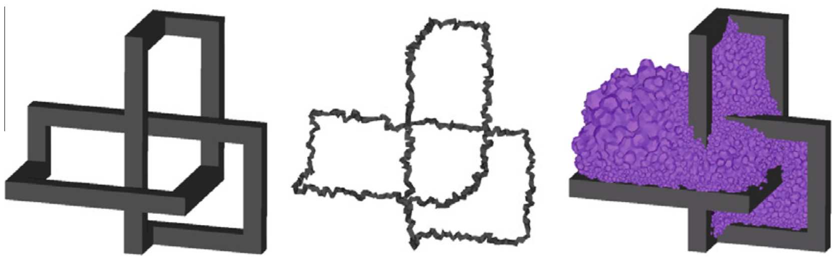
Fig. 9. b4: A trefoil knot conductor.

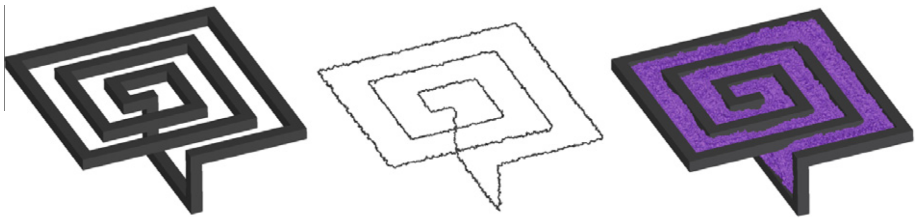
Fig. 10. b9: A spiral conductor.

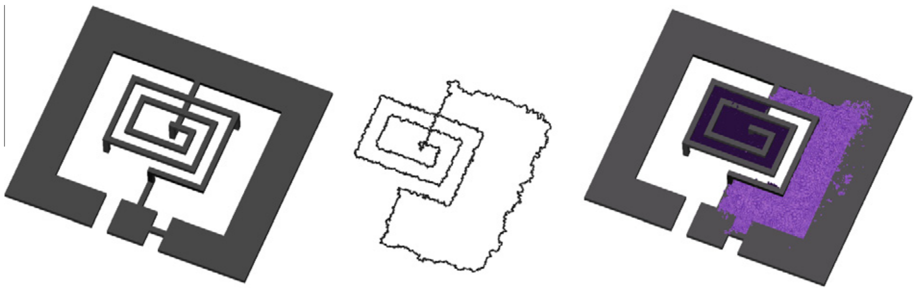
Fig. 11. b101: A micro-inductor.

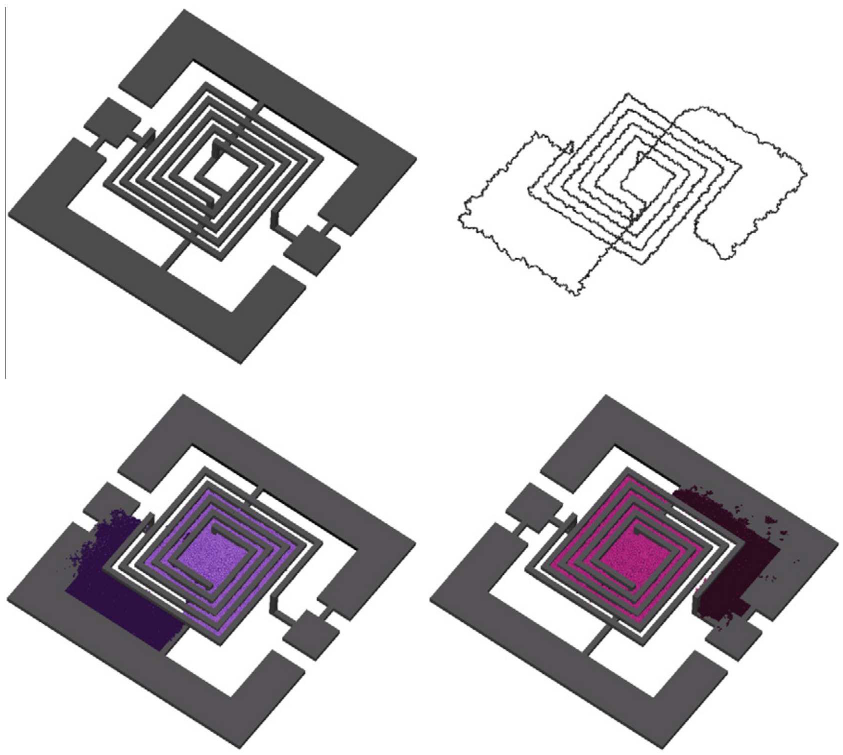
Fig. 12. b102: A micro-transformer.

that the inclusion $H_2(\mathcal{K},\mathcal{K}_a)\hookrightarrow H_2(\mathcal{K}_c,\partial\mathcal{K}_c)$ induces an isomorphism in homology. Hence, $\partial_1 : H_2(\mathcal{K}_c,\partial\mathcal{K}_c) \to H_1(\mathcal{K}_a)$ is an isomorphism. And, since $\mathcal{K}_c$ is composed with connected components $\mathcal{K}_c^1 \cup \ldots \cup \mathcal{K}_c^n$ we have $H_2(\mathcal{K}_c,\partial\mathcal{K}_c) = \bigoplus_{i=1}^n H_2(\mathcal{K}_c^i,\partial\mathcal{K}_c^i)$. Therefore, one can simply take for $i \in \{1,\ldots,n\}$ the boundaries of $H_2(\mathcal{K}_c^i,\partial\mathcal{K}_c^i)$ generators to obtain $H_1(\mathcal{K}_a)$ generators.[10]

We just demonstrated that by summing up the described homology generators of $H_1(\overline{\mathcal{K}\setminus\mathcal{K}_c^i})$, one obtains generators of $H_1(\overline{\mathcal{K}\setminus\mathcal{K}_c}) = H_1(\mathcal{K}_a)$. Now we would like to show the same property for $H^1(\mathcal{K}_a)$ generators. As it is described in [21], in the considered setting, for a given $H_1(\mathcal{K}_a)$ generator one can obtain the dual $H^1(\mathcal{K}_a)$ generator—which actually is what the TCT algorithm does. The only point that needs to be clarified is that this basis can be computed independently for each connected component of $\mathcal{K}_c$. In [21] a characterization of $c^1,\ldots,c^k$ being the dual $H^1(\mathcal{K}_a)$ generators is given. Let $c_1,\ldots,c_k$ be the chosen generators of $H_1(\mathcal{K}_a)$. Then,

[10] To cut a long story short, in the considered case we cannot use the axiom of the sum for $\mathcal{K}_a$, since $\mathcal{K}_a$ is connected. It has only disconnected internal complement, namely $\mathcal{K}_c$. Therefore, we use the axiom of sum for $\mathcal{K}_c$ instead. Since $\mathcal{K} = \mathcal{K}_c \cup \mathcal{K}_a$, once we pick a connected component $\mathcal{K}_c^i$ of $\mathcal{K}_c$, it suffices to compute cohomology of $\overline{\mathcal{K}\setminus\mathcal{K}_c^i}$ and later restrict the resulting cochain to $\mathcal{K}_a$.

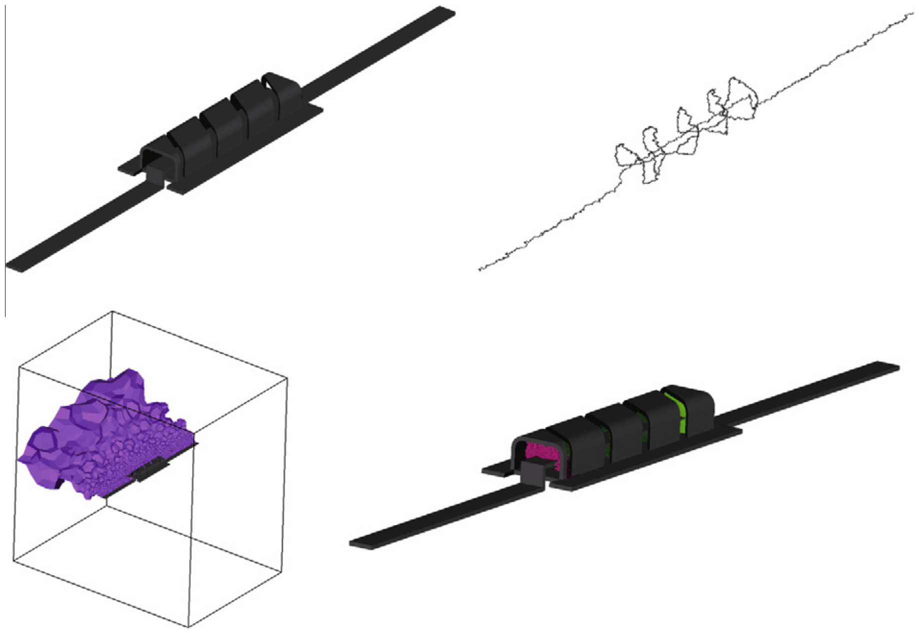

Fig. 13. b104: A micro-coaxial cable.

for every cycle c homologous to c_i we have $\langle c^j, c\rangle = \delta_{ij}$. Let us therefore pick $\mathcal{K}_c^i$ and a chosen $c_1,\ldots,c_n$ generators of $H_1(\overline{\mathcal{K}\setminus\mathcal{K}_c^i})$. Let the dual $H^1(\overline{\mathcal{K}\setminus\mathcal{K}_c^i})$ generators represented by $c^1,\ldots,c^n$ be obtained from TCT algorithm. It is clear that the restriction of those cocycles to $\mathcal{K}_a$ are again cocycles. Moreover, the property $\langle c^j, c\rangle = \delta_{ij}$ still holds for every cycle c homologous to c_i in $\mathcal{K}_a$ (since it holds for $\overline{\mathcal{K}\setminus\mathcal{K}_c^i} \supset \mathcal{K}_a$). Therefore, by summing up cocycles generating $H_1(\overline{\mathcal{K}\setminus\mathcal{K}_c^i})$ one obtains a set of cocycles generating $H^1(\mathcal{K}_a)$.

We still need to show that the ESTT algorithm returns the $H^1(\mathcal{K}_c^i)$ generators. Let us therefore pick an intersection of cycles $b_1,\ldots,b_n$ (defined above such that $\partial b_1,\ldots,\partial b_n$ are fixed $H_1(\mathcal{K}_a)$ generators) with skeletons. Those intersections are supported in a single triangles, and `createCocycles` algorithm enforce value 1 (or -1) on this triangle.[11] Therefore, ESTT algorithm produces a 1-cocycle enforcing value 1 (or -1) on cycles $c_1,\ldots,c_n$ and all cycles homologous to them. Therefore, the cocycles returned by ESTT algorithm are $H^1(\overline{\mathcal{K}\setminus\mathcal{K}_c^i})$ generators.

If for a certain connected component the `Skeletonization` algorithm returns `false` at some position of `computationStatus` vector, then due to the line 5 of TCT algorithm, the algorithm presented in [20] is called that returns the cohomology generators of this component complement. If for a certain connected component the `Skeletonization` algorithm does not fail, then `CreateCocycles` algorithm produces valid 2-cocycles, and ESTT algorithm produces the first cohomology group generators of the complement of the connected component. The following theorem is therefore a straightforward consequence:

Theorem 3. *The* TCT *algorithm when executed on* $\mathcal{K} = \mathcal{K}_c \cup \mathcal{K}_a$ *computes the* $H^1(\mathcal{K}_a)$ *generators.*

At the end we would like to give one final remark. The presented theorem yields a way to parallelize cohomology computations when one wants to compute cohomology groups generators of an acyclic set in $\mathbb{R}^3$ with some elements removed. Cohomology computations can be performed either with standard tools as in [20] or with the TCT algorithm presented in this paper. The parallelized algorithm will be presented in a forthcoming paper.

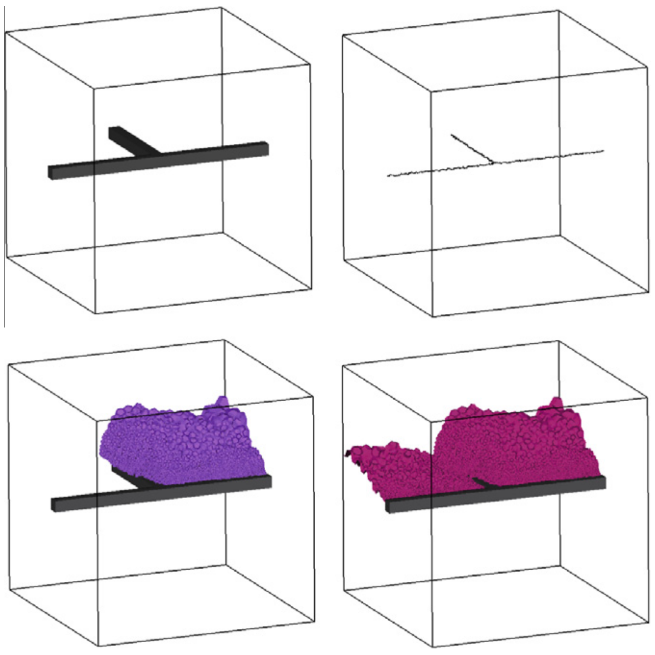

Fig. 14. b105: A Y-shaped conductor.

4. Taking advantage of symmetries

When symmetries in the region can be exploited, it may happen that the conducive region touches the boundary of $\mathcal{K}$. In this case there is no need to modify the definition of potentials but the `Skeletonization` procedure requires some minor modifications. Apart from finding a cycle basis as described previously, one should also add to the skeleton all paths in $D(\mathcal{K}_c)$ that join different

[11] In fact cycle b_i may intersect the skeleton many times but is clear that since we have a cocycle all but one intersections canceled out.

Table 1
Information about the mesh used for each benchmark.

Benchmark	Nodes	Edges	Faces	Tetrahedra	$\beta_0(\mathcal{K}_c)$	$\beta_1(\mathcal{K}_a)$
b1 (circular coil)	4000	27,047	45,776	22,728	1	1
b4 (trefoil knot)	34,057	233,479	398,631	199,208	1	1
b9 (spiral)	288,635	2,130,836	3,684,272	1,842,070	1	1
b101 (micro-inductor)	368,451	2,566,054	4,394,796	2,197,192	1	1
b102 (micro-transformer)	433,690	3,016,979	5,166,120	2,582,830	1	2
b104 (micro-coaxial line)	826,987	5,690,102	9,724,771	4,861,655	2	6
b105 (Y-shaped)	219,257	1,495,486	2,549,178	1,272,948	1	2

connected components of $D(\mathcal{K}_c \cap \partial\mathcal{K})$. An example of such paths can be found in the Fig. 7c and d for the Y-shaped conductor represented in Fig. 7a. Fig. 7b represents the skeleton of the conductor.

A modification that needs to be applied to the `Skeletonization` algorithm is the following. In the two instructions (line 2 and 2 in Algorithm `Skeletonization`) one needs to ensure that the faces of tetrahedra lying on the external boundary of the mesh (i.e. $\partial\mathcal{K}$) are not considered in those `if` statements. This trivial modification is left to the Reader.

Another modification is required for the algorithm to find the cycle basis. When a dual skeleton is obtained in each connected component, one needs to check if there exists a tetrahedron T with a face on the boundary of the complex $\mathcal{K}$. If such T is found, by using a BFS [47] algorithm, the dual graph is searched for any tetrahedron $T', T \neq T'$, which is placed on the boundary of $\mathcal{K}$. For every such a tetrahedron T', a path joining T and T' can be recovered from the BFS algorithm. Every such a path is added to the list containing the cycle basis. This procedure has to be run for each connected component of the skeleton. The obvious details of the algorithm are left to the Reader.

Finally, another modification has to be performed in the `CreateCocycles` algorithm. As `CreateCocycles` algorithm has performed the orientation of dual cycles, now it should also perform the orientation of the dual paths joining the external boundary. Again, the trivial details of the procedure are left to the Reader. The `ESTT` algorithm returns a number of extra cocycles and the proofs in Section 3.8 work without any change for the case described in this Section.

5. Numerical results

The `TCT` algorithm described in this paper has been integrated into the research software CDICE [41]. The software has been implemented in Fortran 90 and the Intel Fortran 90 Compiler has been used to produce the executable. The hardware used for the computations consists of an Intel Core 2 Duo T7700 2.4 GHz laptop with 4 GB of RAM.

To validate the results produced by the `TCT` algorithm and evaluate its performances, some benchmarks have been analyzed. Figs. 8–14 represent the conductive regions which have been analyzed. In particular, for each benchmark, the $\mathcal{K}_c$ and the corresponding thinned regions are shown first, then the cohomology generators are visualized as the dual 2-chains with respect to the $H^1(\mathcal{K}_a)$ 1-chain [18] to achieve a better perception. Table 1 contains the information on the mesh used in each benchmark. The results for all benchmarks obtained with various algorithms are compared in terms of computational time in Table 2. In particular, the other algorithms which have been tested are the GSTT [18,21], the cohomology computation with the coreduction (COR) shaving [20,40] and the cohomology computation with the acyclic sub-complex (ACC) shaving (the fastest version, which is the one implemented with lookup tables) [20,39]. As the mesh size increases, the cohomology computations end up in failures due to having exceeded the memory limit of the laptop. This is due to the fact that the current implementations of (co) homology computations are quite memory consuming. We note that we tested the `TCT` algorithm on the laptop up to five millions of tetrahedra without encountering any problem. In case of a failure, a bigger computer (64 GB of RAM and Intel Xenon E7-8830 2.13 GHz processors) was used for the cohomology computation. The numbers in brackets denote the time required for the cohomology computation with the bigger computer. As one can easily observe, even when run on a powerful computer, the timings required for the cohomology computations are not comparable with the `TCT` algorithm.

Table 2
Time required (s) for cohomology computation with various algorithms.

Benchmark	GSTT [18,21]	H^1 COR [20]	H^1 ACC [20]	`TCT`
b1 (circular coil)	1.07	0.91	0.6	0.03
b4 (trefoil knot)	24.38	30	23	0.6
b9 (spiral)	(424.14)	(4040)	(612)	10.1
b101 (micro-inductor)	(59359)	(> 70000)	(> 70000)	24.5
b102 (micro-transformer)	(> 70000)	(> 70000)	(> 70000)	32.8
b104 (micro-coaxial line)	(612828)	(57714)	(6128)	86.1
b105 (Y-shaped)	(155.12)	(2631)	(289)	4.6

6. Conclusions

A novel technique for cohomology computations called Thinned Current Technique (TCT) has been introduced. The algorithm is automatic, general, straightforward to implement and parallelizable. The TCT compares quite favorably in term of speed with other state-of-the-art algorithms documented in literature and it can be directly extended to deal with a general polyhedral mesh. Moreover, it exhibits also important advantages in the quality of the cohomology basis obtained. In fact, typical source coils are torus-like conductors (i.e. $\beta_1(\mathcal{K}_c^i) = 1$) and with the TCT there is the guarantee that the current that flows inside a coil can be directly related to one and only one cohomology generator. This render the enforcement of current sources particularly trivial. Furthermore, if the user has the need of some specific basis for the cohomology group, the TCT is more appealing than any other methods since it is easier to implement a drag-and-drop user interface to allow the user to select the appropriate cycle basis with respect to accommodate 1-cocycles.

References

[1] H. Poincaré, Analysis Situs, Journal de l'École Polytechnique Ser 2 1 (1895) 1–123.
[2] H. Poincaré, Complément à l'Analysis Situs, Rendiconti del Circolo Matematico di Palermo 13 (1899) 285–343.
[3] H. Poincaré, Second complément à l'Analysis Situs, Proceedings of the London Mathematical Society 32 (1900) 277–308.
[4] H. Poincaré, Papers on Topology: Analysis Situs and Its Five Supplements (translated by J. Stillwell). Hisory of Mathematics vol. 37, 2010 American Mathematical Society, Providence, RI.
[5] J.R. Munkres, Elements of Algebraic Topology, Perseus Books, Cambridge, MA, 1984.

[6] P.R. Kotiuga, On making cuts for magnetic scalar potentials in multiply connected regions, J. Appl. Phys. 61 (1987) 3916–3918.

[7] P.R. Kotiuga, Toward an algorithm to make cuts for magnetic scalar potentials in finite element meshes, J. Appl. Phys. 63 (1988) 3357–3359. erratum: 64; 4257.

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

[9] A. Bossavit, Computational electromagnetism, Academic Press, San Diego, USA, 1998. Available online at <http://butler.cc.tut.fi/bossavit/>.

[10] L. Kettunen, K. Forsman, A. Bossavit, Discrete spaces for div and curl-free fields, IEEE Trans. Magn. 34 (5) (1998) 2551–2554.

[11] Z. Ren, T-Ω formulation for eddy-current problems in multiply connected regions, IEEE Trans. Magn. 38 (2002) 557–560.

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

[13] P.W. Gross, P.R. Kotiuga, Electromagnetic Theory and Computation: A Topological Approach, MSRI Publication, Cambridge University Press, MA, 2004. p. 48.

[14] S. Suuriniemi, Homological computations in electromagnetic modeling, Ph.D. Thesis, Tampere University of Technology, 2004, ISBN: 952-15-1237-7.

[15] M. Desbrun, E. Kanso, Y. Tong, Discrete differential forms for computational modeling, ACM SIGGRAPH 2006 Courses, 39–54.

[16] S. Elcott, Y. Tong, E. Kanso, P. Schröder, M. Desbrun, Stable, circulation-preserving, simplicial fluids, ACM Trans. Graph. 26 (1) (2007) 1–12.

[17] R. Specogna, S. Suuriniemi, F. Trevisan, Geometric t-ω approach to solve eddy-currents coupled to electric circuits, Int. J. Numer. Meth. Engrg. 74 (2008) 101–115.

[18] P. Dłotko, R. Specogna, F. Trevisan, Automatic generation of cuts on large-sized meshes for the T-Ω geometric eddy-current formutlation, Comput. Methods Appl. Mech. Engrg. 198 (2009) 3765–3781.

[19] P. Dłotko, R. Specogna, F. Trevisan, Voltage and current sources for massive conductors suitable with the A-χ geometric formulation, IEEE Trans. Magn. 46 (2010) 3069–3072.

[20] P. Dłotko, R. Specogna, Efficient cohomology computation for electromagnetic modeling, CMES: Comput. Model. Engrg. Sci. 60 (3) (2010) 247–278.

[21] P. Dłotko, R. Specogna, Critical analysis of the spanning tree techniques, SIAM J. Numer. Anal. 48 (4) (2010) 1601–1624.

[22] P. Dłotko, R. Specogna, Efficient generalized source field computation for h-oriented magnetostatic formulations, Eur. Phys. J.-Appl. Phys. (EPJ-AP) 53 (2011) 20801.

[23] P. Dłotko, R. Specogna, Cohomology in electromagnetic modeling, Comm. Comput. Phys. arXiv:1111.2374.

[24] M.I. Monastyrsky (Ed.), Topology in Molecular Biology, Springer, Berlin, 2007.

[25] P.G. Mezey, Group theory of electrostatic potentials: a tool for quantum chemical drug design, Int. J. Quantum Chem. 28 (1985) 113–122.

[26] T.H. Dey, K. Li, J. Sun, D. Cohen-Steiner, Computing geometry-aware handle and tunnel loops in 3D models, ACM Trans. Graph. 27 (3) (2008) 1–9.

[27] T.H. Dey, K. Li, J. Sun, Computing handle and tunnel loops with knot linking, Comput.-Aided Des. 41 (10) (2009) 730–738.

[28] R. Gonzalez-Diaz, P. Real, On the cohomology of 3D digital images, Discrete Appl. Math. 147 (2-3) (2005) 245–263.

[29] R. Gonzalez-Diaz, A. Ion, M. Iglesias-Ham, W.G. Kropatsch, Irregular graph pyramids and representative cocycles of cohomology generators, in: Proceedings of the 7th IAPR-TC-15 International Workshop on Graph-Based Representations in Pattern Recognition (GbRPR '09), Venice, Italy, Springer-Verlag, Berlin, 2009, pp. 263–272.

[30] X. Guo, X. Li, Y. Bao, X. Gu, H. Qin, Meshless thin-shell simulation based on global conformal parameterization, IEEE Trans. Vis. Comput. Graph. 12 (2006) 375–385.

[31] Y. Tong, P. Alliez, D. Cohen-Steiner, M. Desbrun, Designing quadrangulations with discrete harmonic forms, in: Proceedings of the ACM/Eurographics Symposium on Geometry Processing, 2006, 201–210.

[32] H. Whitney, On products in a complex, Ann. Math. 39 (1938) 397–432.

[33] G. Ellis, P. Smith, Computing group cohomology rings from the Lyndon–Hochschild–Serre spectral sequence, J. Symbol. Comput. 46 (4) (2011) 360–370.

[34] G. Ellis, I. Kholodna, Computing second cohomology of finite groups with trivial coefficients, Homol., Homot. Appl. 1 (1) (1999) 163–168.

[35] A. Storjohann, Near optimal algorithms for computing smith normal form of integer matrices, in: Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation, ISAAC, 1996, pp. 267–274.

[36] H. Edelsbrunner, J. Harer, Computational topology an introduction, Am. Math. Soc., Providence, Rhode Island, 2010.

[37] T. Kaczynski, M. Mrozek, M. Slusarek, Homology computation by reduction of chain complexes, Comput. Math. 35 (1998) 59–70.

[38] M. Pellikka, S. Suuriniemi, L. Kettunen, Homology in electromagnetic boundary value problems, Boundary Value Problems 2010 (2010) 1–18.

[39] M. Mrozek, P. Pilarczyk, N. Želazna, Homology algorithm based on acyclic subspace, Comput. Math. 55 (2008) 2395–2412.

[40] M. Mrozek, B. Batko, Coreduction homology algorithm, Dis. Comput. Geomet. 41 (2009) 96–118.

[41] R. Specogna, CDICE: Complementarity and Duality In Computational science and Engineering, 2008–2012. http://www.cdice.org.

[42] G.T. Herman, Geometry of Digital Spaces, Birkhäuser, Boston.

[43] A. Hatcher, Algebraic Topology, Cambridge Publishing Company, Cambridge, UK, 2002.

[44] W.S. Massey, Algebraic Topology: An Introduction (Graduate Texts in Mathematics), Springer, New York, 1967.

[45] R.H. Bing, Some aspects of the topology of 3-manifolds related to the Poincaré Conjecture, Lectures on Modern Mathematics II, T.L. Saaty ed., Wiley, 1964, pp. 93–128.

[46] T. Leviner, H. Lopes, G. Tavares, Optimal discrete Morse functions for 2-manifolds, J. Comput. Geomet.: Theory Appl. 26 (3) (2003) 221–233.

[47] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, The MIT Press, Cambridge, MA, 2001.

[48] P. Dłotko, Acyclicity tables for simplices. <http://www.ii.uj.edu.pl/dlotko/accconf.html>.

[49] P.S. Novikov, On the algorithmic unsolvability of the word problem in group theory, Trudy Mat. Inst. Steklov 44 (1955) 1–143.

[50] K. Mehlhorn, D. Michail, Minimum cycle bases: faster and simpler, ACM Trans. Algor. 6 (1) (2009) 1–12.