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

A R T I C L E   I N F O
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

A B S T R A C T
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.

© 2012 Elsevier B.V. All rights reserved.

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] and 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
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 techniques 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 $K$ 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 $K$ belonging to the conducting region are stored in the sub-complex $K_c$, whereas the elements of $K$ belonging to the insulating region form the sub-complex $K_s$. Moreover, we assume that $K, K_c$ and $K_s$ 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 $K_s$ for $h$-oriented eddy-current formulations. The discrete Ampère’s law in the insulating region can be written as

$$\delta F = I = 0.$$  

where $I$ is the complex-valued electric current 2-cochain—being zero by hypothesis since $K_s$ models the insulating region—and $F$ is the magneto-motive force (mmf) complex-valued 1-cochain. Thanks to the discrete Ampère’s law, $F$ is a 1-cocycle in $K_s$ ($F \in Z^1(K_s, \mathbb{C})$). Consequently, the 1-cocycle $F$ can be expressed as a linear combination of a basis of the first cohomology group $H^1(K_s, \mathbb{C})$ plus a 1-coboundary $B^1(K_s, \mathbb{C})$. However, the basis $H^1(K_s, \mathbb{C})$ in this case can be obtained form a basis of $H^1(K_s, \mathbb{Z})$ where the elements of $\mathbb{Z}$ are treated as elements of $\mathbb{C}$. 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 $K_s$ is embedded in $\mathbb{R}^3$, the homology groups are torsion free. Therefore, from the Universal Coefficient Theorem for homology (see Th. 3.4 [43]), there is a bijective correspondence of generators of $H_1(K_s, \mathbb{Z})$ and $H_1(K_s, \mathbb{C})$. Therefore, from the Universal Coefficient Theorem for cohomology (see Th. 3.1 [43]), there is a bijective correspondence between generators of $H^1(K_s, \mathbb{Z})$ and $H^1(K_s, \mathbb{C})$. Again from the Universal Coefficient Theorem for cohomology, there is a bijective correspondence between generators of $H^1(K_s, \mathbb{Z})$ and $H^1(K_s, \mathbb{C})$. Putting those two together we get the bijective correspondence between generators of $H^1(K_s, \mathbb{Z})$ and $H^1(K_s, \mathbb{C})$. We are using cohomology computed over integers since—they unlike complex and real cohomologies—can be rigorously computed with a computer.
Let us fix the $H^1(K_a)$ 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 $\{t^i\}_{i=1}^{h_1(K_a)}$ being the representatives of the fixed $H^1(K_a)$ cohomology group basis, there exists a set of cycles $\{c_i\}_{i=1}^{b_1(K_a)}$ being the representatives of a $H_1(K_a)$ homology group basis such that $\langle t^i, c_j \rangle = \delta_{ij}$ hold, where by $\langle \cdot, \cdot \rangle$ we denote the dot product of a cochain on a chain.\footnote{We want to point out that the dot product of a cochain on a chain is a discrete analog of integration.} Since the 1-cycles $\{c_i\}_{i=1}^{b_1(K_a)}$ are $H_1(K_a)$ generators and $K$ is homologically trivial there exists 2-chains $\{s_i\}_{i=1}^{b_1(K_a)}$ whose boundaries are the 1-cycles $\{c_i\}_{i=1}^{b_1(K_a)}$. Moreover, the 2-chains $\{s_i\}_{i=1}^{b_1(K_a)} \in C_2(K)$ intersect $K_a$. Let us consider the exact sequence of the pair $(K_a, K)$ [43]:
$$\cdots \to H_2(K_a) \to H_2(K) \to H_2(K, K_a) \to H_1(K_a) \to \cdots$$

Since $H_2(K) = 0$, there is an isomorphism between $H_2(K, K_a)$ and $H_1(K_a)$ induced by the boundary map. From the excision property [43], $H_2(K, K_a)$ is isomorphic to $H_2(K_a, \partial K_a)$. Therefore, the restrictions $\{\sigma_i\}_{i=1}^{b_1(K_a)}$ of the 2-chains $\{s_i\}_{i=1}^{b_1(K_a)}$ to $K_a$ are generators of the $H_2(K_a, \partial K_a)$ homology group.

The current linked by the 1-cycle $c_j$ is defined as the dot product of the current 2-cocycle $I$ and a 2-chain\footnote{The 2-chain $s_i$ is not unique (it is defined up to its boundary $c_j$) but, since the current 2-cocycle $I$ 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 $K$.} $s_i \in C_2(K)$ whose boundary is $c_j$, see Fig. 1b,
$$i_j = \langle I, s_i \rangle. \tag{3}$$

The independent currents $\{i_j\}_{j=1}^{b_1(K_a)}$ are defined as the currents linked by the $H_1(K_a)$ generators $\{c_j\}_{j=1}^{b_1(K_a)}$. The independent currents have been defined in (3) as the currents flowing through the 2-chains $\{s_i\}_{i=1}^{b_1(K_a)}$. Since the contribution to the current is zero in $K_a$, the independent currents as defined in (3) match the currents that flow in the branches of the conductor identified by the $H_2(K_a, \partial K_a)$ generators $\{\sigma_i\}_{i=1}^{b_1(K_a)}$ previously introduced. This gives a physical interpretation of complex numbers $\{i_j\}_{j=1}^{b_1(K_a)}$ 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 $\langle \sigma_i, I \rangle$.

### 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 $H^1(K_a)$ 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 $H^1(K_a)$ generators work directly on the $K_a$ sub-complex. The novel idea of the proposed approach is to put the emphasis on the sub-complex $K_a$, taking advantage—as it is described further in the paper—of the feature that the whole complex $K$ 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...
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 $K_c$ is different, cohomology generators relative to the new current distribution restricted to $K_a$ 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. 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 $L = L_c \cup L_o$. $L_c$ denotes the tetrahedra in $K_c$, whereas $L_o$ denotes tetrahedra in $K_o$. 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 $K_c$

Algorithm findConnectedComponent finds the connected components of $K_c$ once the list $L_c$ is given as input. Let $H$ be a hash table with integer keys.

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 ← tetrahedraAlreadyAssigned $\cup T$;
12:      $cc ← cc \cup T$;
13:   for every tetrahedron $T' \in L_c$ such that $T'$ is a neighbor of $T$ do
14:      if $T' \notin$ tetrahedraAlreadyAssigned then
15:         enqueue ($Q, T'$);
16:      result ← 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 $n$. 

Fig. 2. (a) The pillars of tetrahedra obtained after the thinning process on $K_c$. (b) The skeleton of $K_c$ on the so called dual complex, which pierces the pillars of the tetrahedra. (c) and (d) The two $c^1$ and $c^2$ independent cycles.
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 $T_1, ..., T_n$ correspond to a dual face $D(E)$ bounded by $D(T_1), ..., D(T_n)$. Finally, a vertex $V$ incidental to edges $E_1, ..., E_n$ corresponds to dual volume $D(V)$ bounded by $D(E_1), ..., D(E_n)$. 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 is not a simplicial complex.

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].

**Algorithm 2. Skeletonization**

**Ensure:** List of tetrahedra in the skeleton of the conductor $\mathcal{K}$.

1. Boolean array computationStatus[0..size of $\mathcal{K}$] = true;
2. Vector of vectors result[0..size of $\mathcal{K}$];
3. For $i = 1$ to size of $\mathcal{K}$ do:
   4. Set removedTets = $\emptyset$;
   5. List candidatesToRemove = $\emptyset$;
   6. For every tetrahedron $T \in \mathcal{K}[i]$ do:
      7. If $T$ has faces not shared by other elements in $\mathcal{K}[i]$ then:
         8. candidatesToRemove += $T$;
      9. While $T$ is true do:
         10. Boolean wasTetrahedronRemoved = false;
         11. For every tetrahedron $T \in \mathcal{K}[i]$ do:
            12. Let $f$ be the set of all nodes, edges and faces of $T$ that belong to the external boundary of $\mathcal{K}[i]$ 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 += $T$;
               19. For every tetrahedron $T \in \mathcal{K}[i]$ do:
                  20. If $T$ has faces not shared by other elements in $\mathcal{K}[i]$ or faces shared by elements in removedTets then:
                     21. candidatesToRemove += $T$;
               22. Vector of tetrahedra skeleton = [$T \in \mathcal{K}[i] - T \notin removedTets$];
               23. Boolean wasSkeletonizationSuccessfull = true;
               24. For every tetrahedron $T \in \mathcal{K}[i]$ 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;
                        else:
                           32. computationStatus[i] = false;
               33. Return result, computationStatus;

Fig. 3. A 2-dimensional conductor $\mathcal{K}$, is shown in grey. For the sake of clarity, the insulator $\mathcal{K}$ 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.

Fig. 4. On the left, two tetrahedra intersecting in a node. On the right, two tetrahedra intersecting in an edge.
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. 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 $H^i(K \setminus K_i)$ cohomology group generators, where $K_i$ is the sub-complex with the elements of $K$ belonging to the considered ith connected component. We emphasize that the computation is performed for the complement of the considered connected component $K_i$ (it suffices to treat all remaining conductors as part of the complement). After the computations, the obtained cocycles representing the first cohomology remaining conductors as part of the complement). After the computations, the obtained cocycles representing the first cohomology group generators, where $K$ is the union of the insulating domain $K_a$ with removedTets. In the same spirit, $K_c$ denotes tetrahedra in the conductor $K_c$ 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 $K_a$ does not change. To show this, let us write a Mayer–Vietoris sequence for reduced homology for the pair $(K_a, T)$, where $T$ is a tetrahedron for which the test in the line 2 returns true (i.e. $T$ has an acyclic intersection with $K_a$):

$$\cdots \to H_3(K_a \cap T) \to H_3(K_a) \oplus H_3(T) \to H_3(K_a \cup T) \to \cdots$$

Due to the line 2 of the algorithm, $H_3(K_a \cap T) = 0$. Therefore, we have an isomorphism between $H_3(K_a) \oplus H_3(T)$ and $H_3(K_a \cup T)$. Since $H_3(T) = 0$, we get an isomorphism between $H_3(K_a)$ and $H_3(K_a \cup T)$. As a consequence, one can add $T$ to $K_a$ without changing the homology of $K_a$.

Due to the test in line 2 of the algorithm, we also know that $H_3(K_a \cap \partial K_a) = H_3(K_a \cap K_c)$ remains unchanged. The rest is a simple consequence from the exact sequence of the pair $(K_a, K_c)$ [43]:

$$\cdots \to H_3(K_c) \to H_3(K_a) \to H_3(K_c) \to H_3(K_c) \to \cdots$$

Since $H_3(K_c)$ is homologically trivial, we have an isomorphism between $H_3(K_a \cap K_c)$ and $H_3(K_a \cap K_c)$. Therefore, the topology of $K_c$ is also preserved (with respect to $K_c$) 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 $K_c$ or with $K_a$. 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 $K_c$.

In order to show that the complex dual to the thinned conductor obtained by the skeletonization of $K_c$ 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 $\partial_0$ or $\partial_1$ 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 tetrahedron in 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 $K$.

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
{c_1, \ldots, c_5} 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 (&kappa;). Therefore, due to the line 4 in which the loop at the line 4 in which the

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, h) &isin; cocycle;
8: b_j := −&kappa;(cycles[i][j], T_1)h; k(cycles[i][j], T_2);
9: cocycle := (T_2, b);
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 K.

Once a 2-cochain c obtained by the Algorithm CreateCocycles is provided at input, it finds a 1-cocycle d such that ad = c. 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(\text{card } 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 K. At first, we need to construct a spanning tree on 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 K.

Require: Vector of all tetrahedra in 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 ∈ T such that vert[i] = false;
5: vert[i] := true;
6: Queue Q ← H[i];
7: while q ≠ ∅ do
8: T = pop (Q);
9: for Every node j ∈ T do
10: if vert[j] = true then
11: continue;
12: Let k ∈ T be such that vert[k] is equal true;
13: result ← (j, k);
14: vert[j] := true;
15: Q ← Q − (j, k);
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 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 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-cochain c ∈ C^2(K) produces a 1-cocycle d ∈ C^1(K) such that for every 1-cycle c ∈ Z_1(K) and 2-chain f ∈ C_2(K) with d(f) = e we have (c, f) = (d, e). As it is shown in [22], it always works provided that a cocycle is given as input, which

Such a chain f always exists, since K is homologically trivial.
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 $K$. 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: $K = K_c \cup K_a$ list of tetrahedra, $K_c$ conductive region, $K_a$ insulating region

1: Vector of vectors of pairs (edge, value) result, cohomologyGens;
2: Vector of vectors of tetrahedra cc = findConnectedComponent($K_c$);
3: boolean array skeletonizationSuccess;
4: List of tetrahedra in skeletons of conductors skeleton;
5: (skeleton, skeletonizationSuccess) = Skeletonization(cc);
6: if skeletonizationSuccess[i] = false then
    7: Compute first cohomology generators of $K \setminus c[i]$ 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 $K_a$;
12: return result;

Let $t^1$ and $t^2$ 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 $d^1$ and $d^2$ related with the example are represented in Fig. 6a and b, respectively. The corresponding cohomology $H^1(K_a)$ generators $t^1$ and $t^2$ are represented in Fig. 6c and d.

Using only maximal elements (tetrahedra) in place of a simplcial 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 $O(\text{card}(K_a)^3)$.

If the skeletonization is performed, the ESTT algorithm may take $O(\text{card}(K_a)^3)$ time in the worst case. That gives $O(\text{card}(K_a)^3)$ 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. 6. (a) and (b) Dual faces dual to edges in the support of the cochains $d^1$ and $d^2$, respectively. (c) and (d) Dual faces dual to edges in the support of the cochains $t^1$ and $t^2$, respectively.](image-url)
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 \( H^1(K_a) \) generators. First, we show that in the analysis it suffices to restrict to a complement of a single connected component of \( K_a \) 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 \( K_a \) whose cohomology is to be computed is connected, but its internal complements \( K_c = K_1 \cup \ldots \cup K_n \) are not. We show that it suffices to compute the cohomology of \( K \setminus K_a^i \) for \( i \in \{1, \ldots, n\} \) and restrict the output to \( K_a \) to obtain a cohomology basis of \( K_a \).

Let \( \{c_1, \ldots, c_n\} \) denote the \( H^1(K \setminus K_a^i) \) generators restricted to \( K_a \). We have the following theorem:

**Theorem 2.** \( \{c_1, \ldots, c_n\}_{j=1}^n \) generates \( H^1(K_a) \).

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 \( K_x \). It is clear that one can find the representatives of generators of \( H_1(K \setminus K_x^i) \) that lies on the boundary of \( K_x^i \). The raw intuition of this fact is the following: Let us take any representatives \( c_1, \ldots, c_n \in C_1(K \setminus K_x^i) \) of generators of \( H_1(K \setminus K_x^i) \). Since \( K \) is topologically trivial, one can find \( b_1, \ldots, b_n \in C_0(K) \) such that \( \partial b_i = c_i \) for \( i \in \{1, \ldots, n\} \). Let us now take the cycles \( c_1, \ldots, c_n \in C_1(K \setminus K_x^i) \) being intersections of \( b_1, \ldots, b_n \) with the boundary of \( K_x^i \). 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 \( H_1(K \setminus K_x^i) \) generators.

Then, by using the exact sequence of the pair \((K, K_a)\) [43]

\[
\cdots \rightarrow H_2(K_a) \rightarrow H_2(K) \rightarrow H_2(K, K_a) \rightarrow H_1(K_a) \rightarrow \cdots,
\]

since \( K \) is topologically trivial, we get that \( \partial : H_2(K, K_a) \rightarrow H_1(K_a) \) is an isomorphism. Moreover, thanks to the excision property, we get
that the inclusion $H_2(K, K_a) \to H_2(K_c, \partial K_c)$ induces an isomorphism in homology. Hence, $\partial_1 : H_2(K_c, \partial K_c) \to H_1(K_a)$ is an isomorphism. And, since $K_c$ is composed with connected components $K_1 \cup \ldots \cup K_n$ we have $H_2(K_c, \partial K_c) = \bigoplus H_2(K_i, \partial K_i)$. Therefore, one can simply take for $i \in \{1, \ldots, n\}$ the boundaries of $H_2(K_i, \partial K_i)$ generators to obtain $H_1(K_a)$ generators.

We just demonstrated that by summing up the described homology generators of $H_1(K_c \backslash K_a)$, one obtains generators of $H_1(K_c \backslash K_a) = H_1(K_a)$. Now we would like to show the same property for $H^i(K_a)$ generators. As it is described in [21], in the considered setting, for a given $H_1(K_a)$ generator one can obtain the dual $H^i(K_a)$ 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 $K_c$. In [21] a characterization of $c_1, \ldots, c^k$ being the dual $H^i(K_a)$ generators is given. Let $c_1, \ldots, c^k$ be the chosen generators of $H_1(K_a)$. Then,
for every cycle $c$ homologous to $c_i$ we have $\langle c', c \rangle = \delta_{q_i}$. Let us therefore pick $K^i$ and a chosen $c_1, \ldots, c_n$ generators of $H_1(K \setminus K^i)$. Let the dual $H_1(K \setminus K^i)$ generators represented by $c_1, \ldots, c_n$ be obtained from TCT algorithm. It is clear that the restriction of those cocycles to $K_a$ are again cocycles. Moreover, the property $\langle c', c \rangle = \delta_{q_i}$ still holds for every cycle $c$ homologous to $c_i$ in $K_a$ (since it holds for $K \setminus K^i \supset K_a$). Therefore, by summing up cocycles generating $H_1(K \setminus K^i)$ one obtains a set of cocycles generating $H_1(K_a)$.

We still need to show that the ESTT algorithm returns the $H_1(K^i)$ 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 $H_1(K_a)$ generators) with skeletons. Those intersections are supported in a single triangles, and createCocycles algorithm enforce value $1$ (or $-1$) on this triangle. 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 $H_1(K \setminus K^i)$ 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 $K = K_c \cup K_a$ computes the $H^1(K_a)$ 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 $R^3$ 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.

### 4. Taking advantage of symmetries

When symmetries in the region can be exploited, it may happen that the conducive region touches the boundary of $K$. 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 $D(K_c)$ 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.
connected components of $D(K_c \cap \partial K)$. 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. $\partial K$) 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 $K$. 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 $K$. 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 GSTT 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 $K_c$ and the corresponding thinned regions are shown first, then the cohomology generators are visualized as the dual 2-chains with respect to the $H^1(K_c)$ 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.

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. $\beta_1(K_c') = 1$) 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


