Università degli Studi di Roma Tre
Dipartimento di Informatica e Automazione
Via della Vasca Navale, 79 – 00146 Roma, Italy

Stokes BSP: a data structure to compute coboundary and differential forms

Franco Milicchio

RT-DIA-77-2003 **Aprile 2003**

Università di Roma Tre,
Via della Vasca Navale, 79
00146 Roma, Italy.

ABSTRACT

The interest about the applications of algebraic topology tools are rapidly increasing. In particular, chains and cochains are useful in describing physical theories in a compact and elegant way. It has also been shown in [10] a strong relation between cochains and the Stokes theorem. The intent of this paper is not only to show a data structure capable of handling cochains and make operations over them, but also provide a versatile data structure useful for field calculus, not relying on fixed structures like grid decompositions, allowing also refinements and better approximation.

1 Introduction

This paper presents a data structure capable of handling cochains, calculate the coboundary over them, and calculate the Stokes theorem over a cell complex. Several applications use the Stokes theorem to calculate a field over a complex, and many of these softwares use finite element methods, thus using grids. These techniques can reach a fairly good accuracy if the grid involved in the process fits for the problem: a grid is so the main issue (many companies live on producing appropriate grids for fiels calculuses).

The main contribution of using BSP trees is that grids are not used anymore, in the sense that the accuracy can be reached just "going deeper" into the structure: if we want more accurate field calculus, we just choose the convex cell and partitionate it until we reach our approximation goal. This approach has some other advantages. A Binary Space Partition tree is a really well known data structure, long used in other computer graphic fields that require a great visualization responsiveness; it solves the point-in-or-out problem in logarithmic time with respect to the number of hyperplanes that define a complex; the size of the data structure for really complex objects is fairly small; binary operations can be made directly merging BSP trees as shown by Naylor, Amanatides e Thibault (cf. [6]).

The first sections of this documents intend to give a brief overview of some mathematical objects, like *differential k-forms* and their main operations, *cochains* and the *coboundary operation*, the *Stokes theorem* and its cochain-based representation. The last sections are dedicated to the SBSP data structure definition and its relation with cochains and the Stokes theorem.

2 Mathematical preliminaries

The data structure that will be shown in this note has not only the intent of storing informations about cochains, but it is also useful when calculating a field over a cell complex — that is, calculating a differential form over the complex.

So it will be a good idea to propose a little compendium regarding differential forms. For a better description of differential forms and their application, cf. [11].

Let p be a point in $\Re^n$. A **vector field** in $\Re^n$ is a map v that associate $\forall p \in \Re^n$ a vector $v(p)$:

$$v(p) = \sum_i a_i(p) e_i$$

where the functions $a_i : \Re^n \longrightarrow \Re$ characterize the vector field v and e_i are the canonical basis vectors.

We can associate to the space $\Re^n$ its dual space $\Re^{n*}$, set of all the linear functions

$$\varphi : \Re^n \longrightarrow \Re$$

A basis for this space is obtained considering dx_i, where x_i is the function that for every point $P \in \Re^n$ gives the i-th coordinate. The set

$$\{(dx_i)\}$$

is the dual basis of $\{(e_i)\}$ since

$$(dx_i)(e_j) = \frac{\partial x_i}{\partial x_j} = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases} \tag{1}$$

Definition 1 (Linear differential form). *Let v be a vectorial field. The scalar product $v(P) \cdot dP$ has the form*

$$\varphi = \sum_i v_i dx_i \tag{2}$$

and it is defined as ***linear differential form*** *or* ***external 1-form****.*

Let us consider two linear differential forms φ_1 and φ_2. Let $\Lambda^2(\Re^{n*})$ be the set of all the bilinear alternant functions $\varphi : \Re^n \times \Re^n \longrightarrow \Re$. We can obtain an element $\varphi_1 \wedge \varphi_2 \in \Lambda^2(\Re^{n*})$ defining

$$(\varphi_1 \wedge \varphi_2)(v_1, v_2) = det(\varphi_i(v_j)) \tag{3}$$

It is clear that the set $\{dx_i \wedge dx_j, i < j\}$ is a basis for $\Lambda^2(\Re^{n*})$. Moreover, we have that

$$dx_i \wedge dx_j = -dx_j \wedge dx_i \quad i \neq j \tag{4}$$

$$dx_i \wedge dx_i = 0 \tag{5}$$

Definition 2 (Quadratic differential form). *A* ***quadratic differential form****, or* ***external 2-form****, is defined as*

$$\omega = \sum_{i<j} a_{ij} dx_i \wedge dx_j \tag{6}$$

Just like we did defining a 2-form, we can generalize the concept of differential forms. Let $\Lambda^k(\Re^{n*})$ be the set of all the alternating k-linear forms

$$\varphi : \underbrace{\Re^n \times \ldots \times \Re^n}_{\text{k times}} \longrightarrow \Re$$

An element $\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_k$ can be obtained considering

$$\varphi_1 \wedge \varphi_2 \ldots \wedge \varphi_k(v_1, v_2, \ldots, v_k) = \det(\varphi_i(v_j)) \tag{7}$$

From determinant's properties, we can say that $\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_k$ is k-linear and alternating. It also possible to demonstrate that the set

$$\{dx_{i_1} \wedge dx_{i_2} \wedge \ldots \wedge dx_{i_k}, \quad i_1 < i_2 < \ldots < i_k, \quad i_j \in \{1, \ldots n\}\}$$

is a basis for $\Lambda^k(\Re^{n*})$.

Definition 3 (Differential k-form). *We define* ***differential k-form*** *the form $\omega \in \Lambda^k(\Re^{n*})$:*

$$\omega = \sum_{i_1 < \ldots < i_k} a_{i_1 \ldots i_k} dx_{i_1} \wedge \ldots \wedge dx_{i_k} \quad i_j \in \{1, 2, \ldots, n\} \tag{8}$$

For our ease, we will denote the tuple $(i_1, \ldots, i_k), \quad i_1 < \ldots < i_k, \quad i_j \in \{1, 2, \ldots, n\}$ as i, as a consequence, σ_i will be the basis form $dx_{i_1} \wedge \ldots \wedge dx_{i_k}$. Thus it is possible to write a differential form in a quite easier way:

$$\omega = \sum_i a_i \sigma_i \tag{9}$$

Conventionally, we have that a differential 0-form is a differentiable function

$$f : \Re^n \longrightarrow \Re$$

It is interesting to show the tight connection between differential k-forms and k-vectors: all the operations possible on k-vectors have analogous operations on k-forms.

Let ω and φ be two differential k-forms.

$$\omega = \sum_i a_i \sigma_i \qquad \varphi = \sum_i b_i \sigma_i$$

We define the **sum**:

$$\omega + \varphi = \sum_i (a_i + b_i)\sigma_i \tag{10}$$

Now, let ω be a k-form and φ a s-form. We can define the **external product** $\omega \wedge \varphi$ the differential $(k+s)$-form:

$$\omega \wedge \varphi = \sum_{ij} a_i b_j \sigma_i \wedge \sigma_j \tag{11}$$

where

$$\omega = \sum_i a_i \sigma_i, \quad i \in (i_1, \ldots, i_k), \quad i_1 < \ldots < i_k$$

$$\varphi = \sum_j b_j \sigma_j, \quad j \in (j_1, \ldots, j_s), \quad j_1 < \ldots < j_s$$

The external product of two differential forms has properties similar the external product of two k-vectors.

Property 1. *Let ω, φ and θ be respectively a k-form, a s-forma and a r-form. Then*

1. $(\omega \wedge \varphi) \wedge \theta = \omega \wedge (\varphi \wedge \theta)$
2. $(\omega \wedge \varphi) = (-1)^{ks}(\varphi \wedge \omega)$
3. $\omega \wedge (\varphi + \theta) = \omega \wedge \varphi + \omega \wedge \theta, \qquad$ *if* $r = s$

It is important to remark that $dx_i \wedge dx_i = 0$ does not lead to $\omega \wedge \omega = 0$ for every k-form. In fact, if we consider

$$\omega = x_1 dx_1 \wedge dx_2 + x_2 dx_3 \wedge dx_4$$

the external product will be

$$\omega \wedge \omega = 2x_1 x_2 dx_1 \wedge dx_2 \wedge dx_3 \wedge dx_4$$

Now we can define an operation over k-forms directly involved in the Stokes theorem:

Definition 4 (External derivative). *The* ***external derivative****, or* ***external differentiation*** *of a differential form f is*

$$df = \sum_i \frac{\partial f}{\partial x_i} dx_i \tag{12}$$

being f a 0-form, in other words, a scalar function. Considering a k-form $\omega = \sum_i a_i \sigma_i$, with $k \geq 1$, the external derivative is

$$d\omega = \sum_i da_i \wedge \sigma_i \tag{13}$$

It is clear that the external derivative increases the dimension of the form, thus producing a $(k+1)$-form from a k-form.

Example of external differentiation

Let's see in detail with an example how the external derivarive works. Let ω be the differential 1-form

$$\omega = xyzdx + yzdy + (x+z)dz \tag{14}$$

obtained by a linear combination of elements of the orthogonal basis

$$\sigma_1 = dx \quad \sigma_2 = dy \quad \sigma_3 = dz$$

with coefficients:

$$a_1 = xyz \quad a_2 = yz \quad a_3 = x + z$$

By definition, we calculate:

$$da_1 = yzdx + xzdy + zydz \tag{15}$$

$$da_2 = zdy + ydz \tag{16}$$

$$da_3 = dx + dz \tag{17}$$

So the external differentiation is:

$$d\omega = (yzdx + xzdy + xydz) \wedge dx + (zdy + ydz) \wedge dy + (dx + dz) \wedge dz \tag{18}$$

Applying the distributive property of the external product we obtain:

$$d\omega = yzdx \wedge dx + xzdy \wedge dx + xydz \wedge dx + zdy \wedge dy + ydz \wedge dy + dx \wedge dz + dz \wedge dz \tag{19}$$

Knowing that $dx_i \wedge dx_i = 0$ and $\varphi \wedge \theta = -\theta \wedge \varphi$, we have:

$$d\omega = -xzdx \wedge dy - xydx \wedge dz - ydy \wedge dz + dx \wedge dz = \tag{20}$$

$$= -xzdx \wedge dy + (1 - xy)dx \wedge dz - ydy \wedge dz \tag{21}$$

3 Cochains

Let us now focus our attention on a particular object from the algebraic topology.

Definition 5 (Cochain). *A **cochain** on a cell complex C, with coefficient in K, with K being an abelian group, is a formal sum:*

$$\sum_{C_i \in C^k} g_i C_i \tag{22}$$

where $g_i \in K$ and C^k is the subset of k-dimensional cells from the complex C.

The group of the k-dimensional cochains, or *k-cochains*, over a complex C with coefficient in the abelian group G is denotated

$$C^k(C; G)$$

On the cochains is defined a fundamental operation:

Definition 6 (Coboundary). *The **coboundary** of a k-cochain c over a cell complex K with coefficients in G, is the operation*

$$\delta : C^k(K; G) \longrightarrow C^{k+1}(K; G)$$

defined as

$$\delta(c) = \sum_i g_i C_i \tag{23}$$

where g_i are the coefficients associated to the $(k+1)$-cochain, defined as

$$g_i = \sum_{f \in F^k(C_i)} \sigma(f, C_i) g_f \tag{24}$$

where f are the faces of the cell C_i and $\sigma(f, C_i)$ gives the relative orientation sign between the face and the higher dimensional cell C_i.

The definition of cochain and coboundary actually derive from the definition of *chain* and *boundary*, duals of chains and coboundary. These definitions, as well as the relative duals, are defined as formal sum. For more details, see [9], [5].

The coboundary on a k-dimensional cochain is possible only if the cochain is defined over a complex of dimensions at least $k+1$. In the example pictured, we show a 1-cochain over a 2-complex. The coboundary will produce a 2-cochain.

4 Stokes theorem

In most of the physical problems we solve, we use one of the most important theorems, the **Stoke's theorem**:

Theorem 1 (Stokes). *Let ω be a differential form and R a cell complex. We have that:*

$$\int_{\partial R} \omega = \int_R d\omega \tag{25}$$

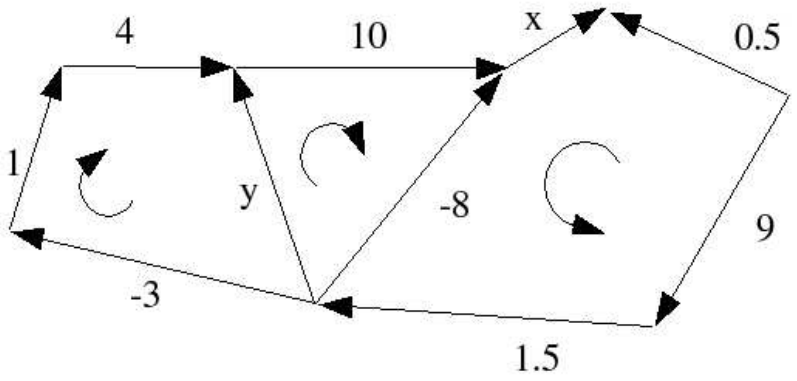


Figure 1: Example of a 1-cochain. Cells from left to right are C_1, C_2, C_3.

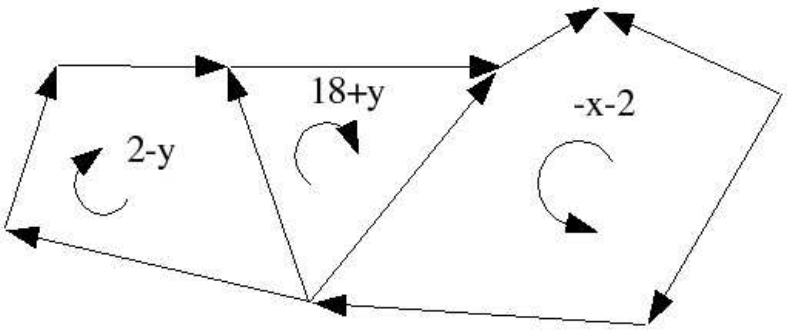


Figure 2: Result of the coboundary: $\sum_i g_i C_i = (2-y)C_1 + (18+y)C_2 + (-x-2)C_3$

This is actually a fundamental theorem, since it is *dimension-independent*, *coordinate-free* and *metric-independent*. Moreover, this theorem shows how to have a finite balace law from an infinitesimal one (cf. [10]).

Let us now consider the left side of the Stokes theorem and write a diagram:

$$\int_{\partial R} \omega = \begin{cases} \text{k-cochain} & \xrightarrow{\delta} & (k+1)\text{-cochain} \\ \int \uparrow & & \\ \text{k-form} & & \end{cases} \tag{26}$$

where the vertical step corresponds to an integration over the boundary ∂R. This operation, being R a cell complex, is done cell-by-cell. The final result is to associate to every cell a coefficient given by the integration: in other words, we create a cochain. The horizontal step transfers the signed sum of all the coefficient to the adjacent $(k+1)$-cells.

On the other hand, we can write a diagram for the other theorem's part:

$$\int_R d\omega = \begin{cases} & & (k+1)\text{-cochain} \\ & & \uparrow \int \\ \text{k-form} & \xrightarrow{d} & (k+1)\text{-form} \end{cases} \tag{27}$$

So, the Stokes theorem can be written with a commutative diagram, obtained by the union of the two diagrams shown above:

$$\begin{array}{ccc} \text{k-cochain} & \xrightarrow{\delta} & (k+1)\text{-cochain} \\ \int \uparrow & & \uparrow \int \\ \text{k-form} & \xrightarrow{d} & (k+1)\text{-form} \end{array} \tag{28}$$

This diagram introduces a **type** for all the coefficients. A coefficient F from a cell $\mathbb{C}$ taken from the cochain row, that is from the upper row, represents a *quantity*. A coefficient g from an infinitesimal cell of the lower row, that is the differential forms row, represents a density. This densities will have clearly a dimension-dependent type: Quantity/Length, Quantity/Area, Quantity/Volume, ...

5 Cochains and BSP trees

In order to use BSP trees, we must define two operations over cochains derived from the intrinsic structure of a binary space partition. When building a BSP tree, a hyperplane can intersect a cell, in this case, the cell must be "splitted" in two parts. This operation must preserve the coboundary value.

Definition 7 (Coherent partition). *Let C be a $(n-1)$-cochain in E^n and let H be a n-cell adjacent to the cohain's cells — the $(n-1)$-cells are so faces of H. We define* ***coherent partition*** *of the cochain* ***inducted by the hyperplane*** *h the cochain defined as:*

- *The cell H becomes a cell complex of two cells H' and H'' individuated by the hyperplane and oriented as the original cell H*
- *The hyperplane generates a $(n-1)$-cell α, adjacent to H' and H'' coherently oriented*

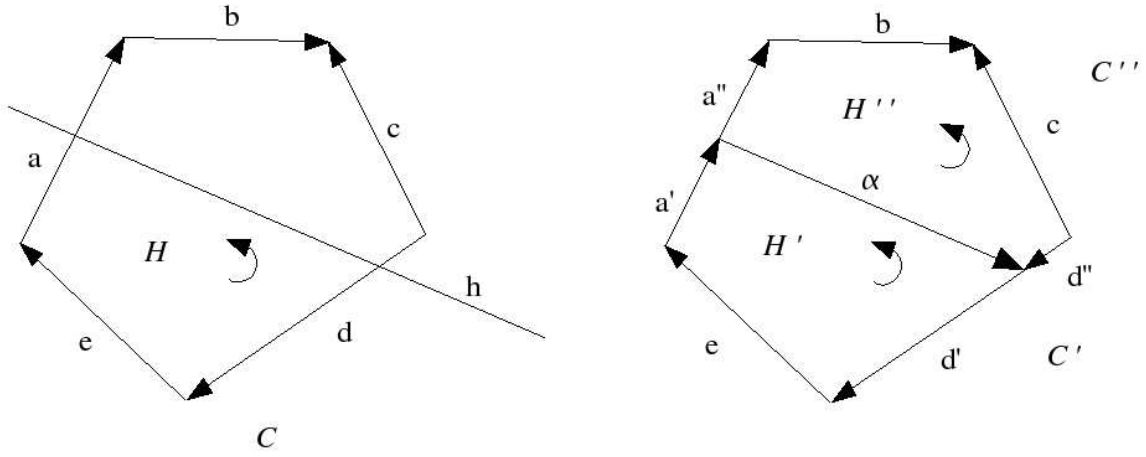


Figure 3: Example of a coherent partition

- *The cells that intersect the hyperplane are divided into two cells oriented as the original cell*
- *To every divided cell resulted by the "cut" of a cell K with coefficient k, are associated the coefficients k' and k'' with the constraint $k' + k'' = k$*

A coherent partition of a cochain creates two adjacent cochains coherently oriented. In figure 3 and 4 we provide an example of a coherent partition in 2D and 3D. The newly generated cell α has a coefficient over which we didn't put any constraint, except the fact that it must belong to the same abelian group of other coefficients.

Definition 8 (Coherent union). *Let C_1 and C_2 be two n-cells in E^n, $(n-1)$-adjacents, coherently oriented, over which is defined a n-cochain. We define* ***coherent union*** *of the two cells, the cell*

$$C = C_1 \cup C_2 \tag{29}$$

obtained by the "deletion" of the adjacency, oriented as the original cells, with a coefficient sum of the two coefficients of cells C_1 and C_2. On C is so defined a n-cochain.

With these operations, we can define a property of cochains:

Theorem 2 (Coboundary invariance). *The coboundary of a k-cochain in E^{k+1} is equal to the coherent union of cells resulting from the coboundary over the cochains obtained from a coherent partition of the original k-cochain.*

Proof. Let C be a $(k+1)$-cell over whose faces is defined a k-cochain in E^{k+1}. The coboundary of this cochain is a $(k+1)$-cochain with a single cell C^{k+1} with coefficient

$$\sum_{C_i \in C^k} g_{C_i} \sigma(C_i, C) \tag{30}$$

Let us coherently partition C with a hyperplane h. This partitioning will create two adjacent cochains C' and C''. Let K' and K'' be respectively the set of cells belonging

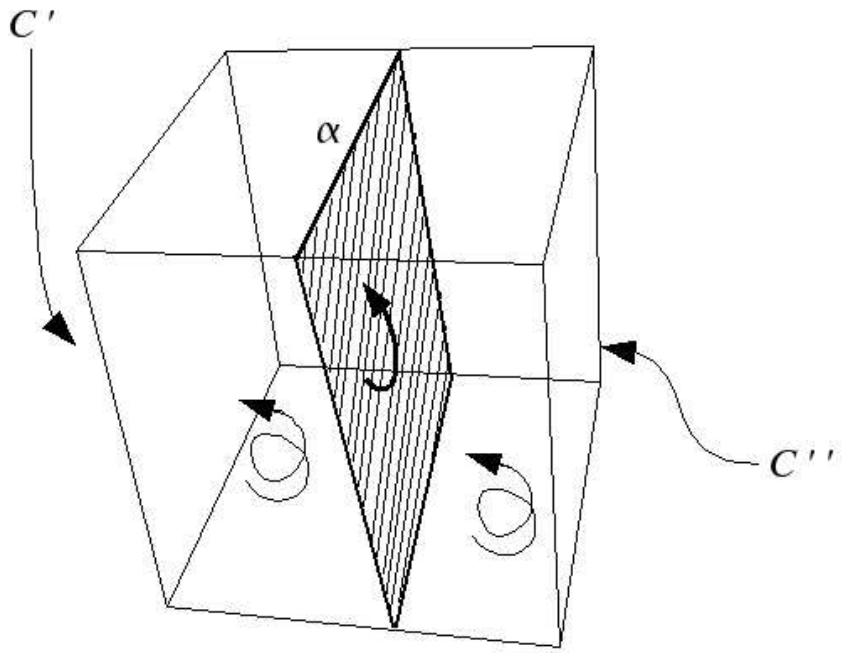


Figure 4: Coherent partition of a 2-cochain

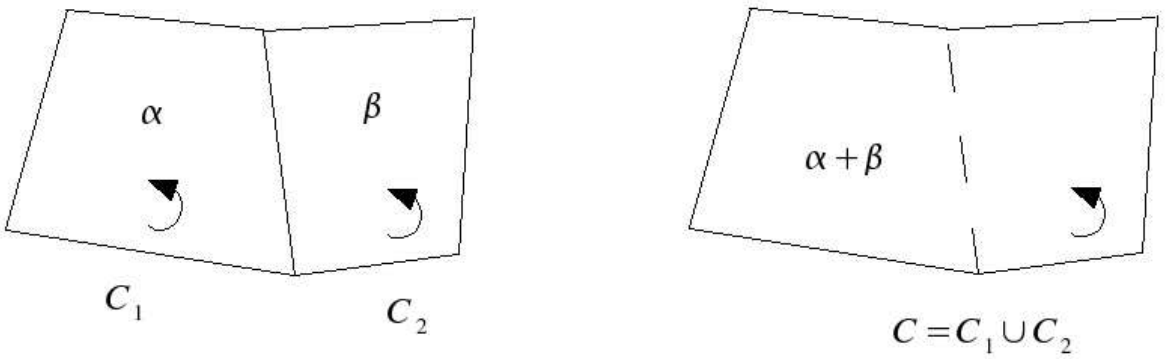


Figure 5: Coherent union of two cells of a cochain

to C' and C''"cut" by the hyperplane h. The coboundary on C' is a single cell with coefficient

$$\bar{\delta}(C') = \sum_{C'_i \in C'} g_{C'_i} \sigma(C'_i, C') \tag{31}$$

Let us separate the contribute of the cut cells:

$$\bar{\delta}(C') = \sum_{C'_i \in (C' \setminus K')} g_{C'_i} \sigma(C'_i, C') + \sum_{C'_i \in (C' \cap K')} g_{C'_i} \sigma(C'_i, C') \tag{32}$$

We can do the same on the cochain C'':

$$\bar{\delta}(C'') = \sum_{C''_i \in (C'' \setminus K'')} g_{C''_i} \sigma(C''_i, C'') + \sum_{C''_i \in (C'' \cap K'')} g_{C''_i} \sigma(C''_i, C'') \tag{33}$$

These are the coefficient associated to the higher-order cells H' and H''. The coherent union of these cells is the cell

$$C = H' \cup H'' \tag{34}$$

with the associated coefficient:

$$\bar{\delta}(C') + \bar{\delta}(C'') \tag{35}$$

Thus, we have

$$\begin{aligned} \bar{\delta}(C') + \bar{\delta}(C'') = & \sum_{C'_i \in (C' \setminus K')} g_{C'_i} \sigma(C'_i, C') + \sum_{C'_i \in (C' \cap K')} g_{C'_i} \sigma(C'_i, C') + \\ & + \sum_{C''_i \in (C'' \setminus K'')} g_{C''_i} \sigma(C''_i, C'') + \sum_{C''_i \in (C'' \cap K'')} g_{C''_i} \sigma(C''_i, C'') \end{aligned} \tag{36}$$

Let us consider the two sums over $C' \setminus K'$ and $C'' \setminus K''$. In this sets is included the newly created cell α. So we have:

$$\sum_{C' \setminus (K' \cup \{\alpha\})} g_{C'_i} \sigma(C'_i, C') + \sum_{C'' \setminus (K'' \cup \{\alpha\})} g_{C''_i} \sigma(C''_i, C'') + \alpha\sigma(\alpha, C') + \alpha\sigma(\alpha, C'') \tag{37}$$

Analyzing the sets over which are defined the sums, we have

$$(C' \setminus (K' \cup \{\alpha\}) \cup (C'' \setminus (K'' \cup \{\alpha\}) = C \setminus K \tag{38}$$

where K is the set of cut cells of C. Moreover, we have

$$\sigma(\alpha, C') = -\sigma(\alpha, C'') \tag{39}$$

because the cells are coherently oriented, so

$$\alpha\sigma(\alpha, C') + \alpha\sigma(\alpha, C'') = 0 \tag{40}$$

Thus, we have

$$\sum_{C_i \in (C \setminus K)} g_{C_i} \sigma(C_i, C) \tag{41}$$

because C' and C'' are by construction oriented as C.

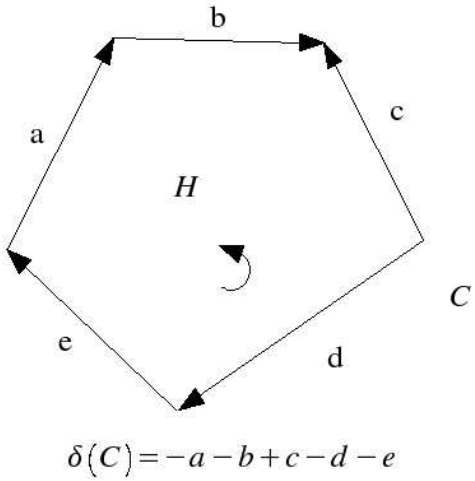


Figure 6: Example of proof — Original cochain

Let us examine the second part of the sums. These are sums over the cells which are cut by the hyperplane:

$$\sum_{C'_i \in (C' \cap K')} g_{C'_i} \sigma(C'_i, C') + \sum_{C''_i \in (C'' \cap K'')} g_{C''_i} \sigma(C''_i, C'') \tag{42}$$

By constructions, we have

$$g_{C'_i} = \theta'_i g_{C_i} \quad g_{C''_i} = \theta''_i g_{C_i} \tag{43}$$

with

$$\theta'_i + \theta''_i = 1 \tag{44}$$

and

$$\sigma(C'_i, C') = \sigma(C''_i, C'') = \sigma(C_i, C) \tag{45}$$

The sum is so

$$\sum_{C_i \in (C \cap K)} g_{C_i} \sigma(C_i, C) \tag{46}$$

Let us reconsider the two coboundaries, we have thus

$$\delta(C') + \delta(C'') = \sum_{C_i \in (C \setminus K)} g_{C_i} \sigma(C_i, C) + \sum_{C_i \in (C \cap K)} g_{C_i} \sigma(C_i, C) = \tag{47}$$

$$\sum_{C_i \in C} g_{C_i} \sigma(C_i, C) \tag{48}$$

So we have demonstrated the property for one cell. This is also valid for a cochain constituted by n cells, just repeating the same procedure for every cell in the considered cochain. □

This property leads directly to another one:

Property 2 (Simplicial cochains). *The coboundary over a k-cochain is equal to the coherent union of the cells originated by a triangulation of the cochain.*

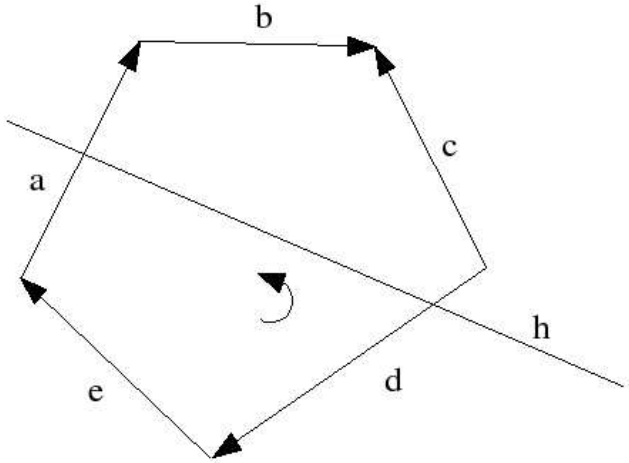


Figure 7: Example of proof — Cut of the cochain

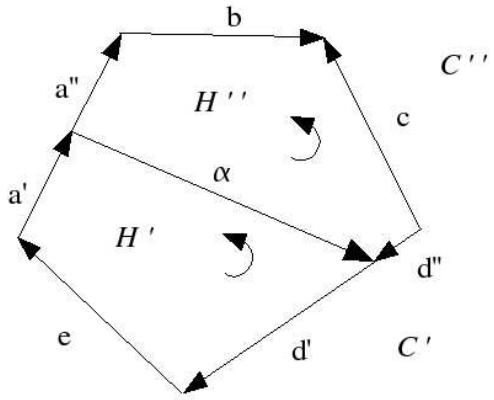


Figure 8: Example of proof — Cochains C' and C'' generated by the cut

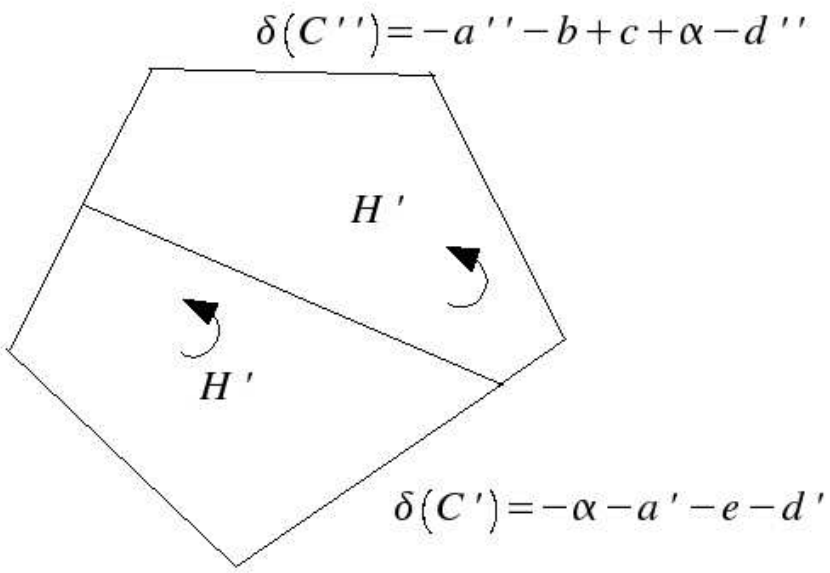
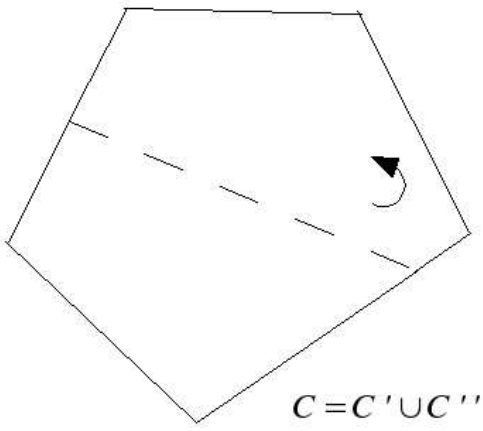


Figure 9: Example of proof — Coboundary over C' and C''



$C = C' \cup C''$

$$\delta(C') + \delta(C'') =$$

$$-a'' - b + c + \alpha - d'' - \alpha - a' - e - d' =$$
$$-(a' + a'') - b + c - (d' + d'') - e + \alpha - \alpha =$$

$$-\boldsymbol{a} - \boldsymbol{b} + \boldsymbol{c} - \boldsymbol{d} - \boldsymbol{e} =$$

$$= \delta(C)$$

Figure 10: Example of proof — Coherent union of H' and H''

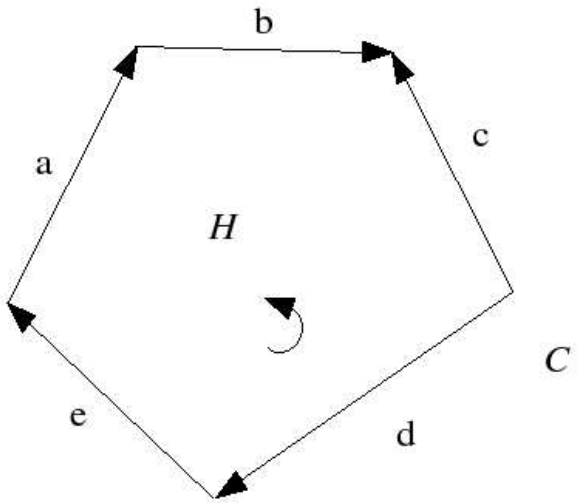


Figure 11: Example of simplicial decomposition — Original cochain

Proof. Let C be a k-cochain adjacent to a $(k+1)$-cochain H. A triangulation generates simplicial cells: these new cells "triangulate" H. We can extend these k-cells until we reach the cochain: we generate thus hyperplanes intersecting the original cochain. We're now in the previous situation, so we can iterate the process for every hyperplane generated by each new k-cell in the triangulation. □

6 Stokes BSP: SBSP

It has been proved that a cochain can be splitted in two by a hyperplane and that the coboundary is invariant with respect to this operation. This property makes it possible to use BSP trees to represent a cochain and calculate the coboundary.

We know that BSP trees cannot represent every object: they can represent only **full objects**. This is a restriction for cochain representation: we can represent $(n-1)$-cochains in E^n, generating n-cochains with the coboundary.

In the **SBSP** data structure we represent k-cochains associating a name to every adjacent $(k+1)$-cell; for every k-cell we associate the respective hyperplane, the $(k+1)$-cells it belongs to and the relative sign between the two. The BSP structure is thus modified: every node ν represents a convex region, with the following information:

1. The **coefficient** associated to the k-cell represented by the hyperplane g_ν
2. The adjacent $(k+1)$**-cells**, or better their **names** C_ν
3. For every $(k+1)$-cell, the **relative orientation**, that is the sign sgn_{C_ν}

Just like BSP trees, we can have a hyperplane cutting a cell. In this situation, just as it happens in BSP trees, the cell must be splitted in two preserving the coboundary, so with a **coherent partitioning**. We don't consider the newly created cell, so we do not store its information: this is equivalent to have a null coefficient associated to this cell.

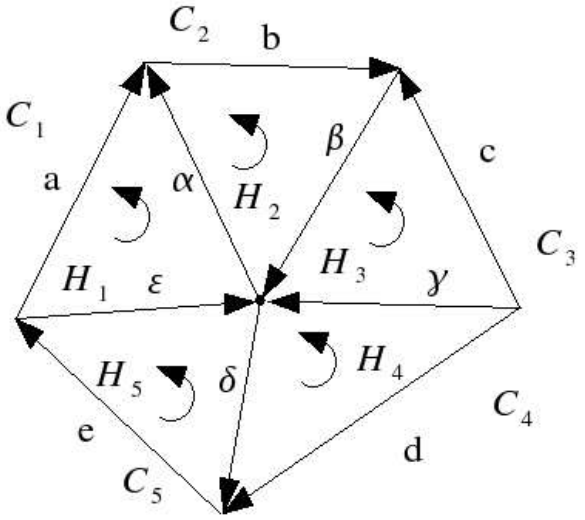


Figure 12: Example of simplicial decomposition — Triangulation

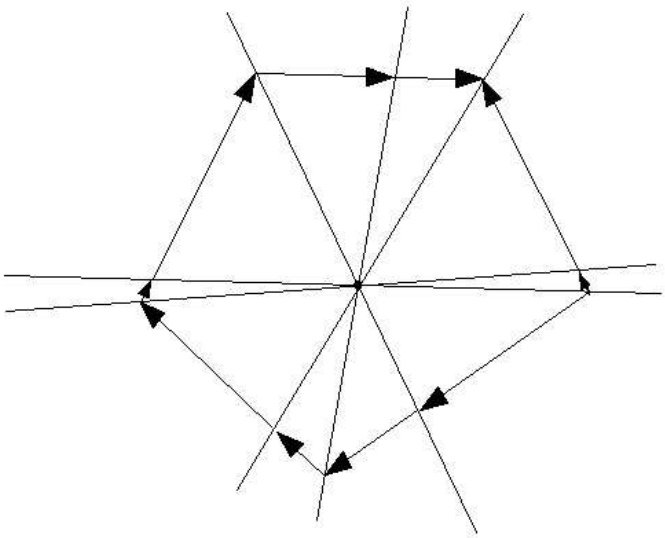


Figure 13: Example of simplicial decomposition — Example of proof

A binary space partition erases the topology, keeping only the essential informations. In order to reconstruct the topology of cochains we introduce an adjacency graph between the higher-order cells, with every link representing the adjacency hyperplane. This structure can also store the coefficients associated to the higher-dimension cells generated by a coboundary operation.

```
struct sbsp_cell_sign
{
    bool      relative_orientation;
    cell_name cell;
}

struct sbsp_node
{
    hyperplane            h;
    abelian_group         coeff;
    list<sbsp_cell_sign>  adjacence_list;

    sbsp_node             *above, *below;
}
```

Figure 14: SBSP data structure — C pseudo-code

The following pictures represent a cochain, followed by a possible representation using a SBSP tree.

7 Coboundary and SBSP

We now know that a cochain can be represented by a SBSP data structure. This structure stores the coefficients associated to every cell in the cochain, the adjacent higher-order cells and their relative signs.

It is clear that we have all the informations we need to compute the coboundary over a cochain. A simple tree traversing permits to easily calculate the coboundary:

1. Associate a null coefficient to every higher-order cell

```
struct adjacency
{
    cell_name       cell_1, cell_2;
    hyperplane      h;
}

struct adjgraph
{
    list<adjacency> adj_list;
}
```

Figure 15: SBSP data structure: adjacency graph — C pseudo-code

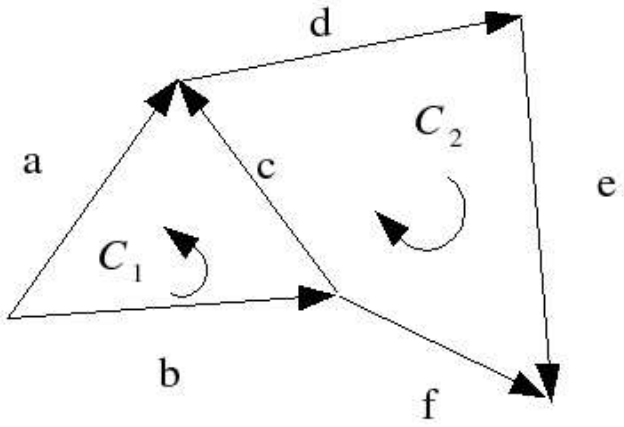


Figure 16: Cochain to be represented

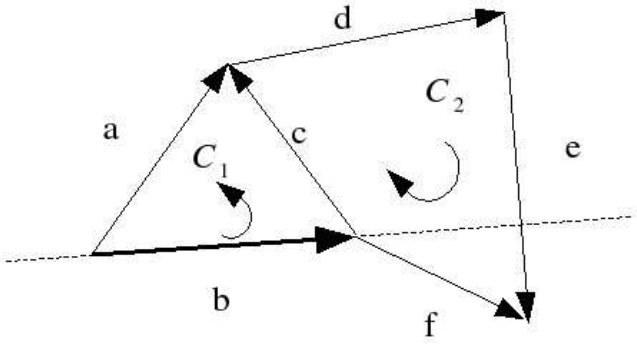


Figure 17: First step of partitioning algorithm

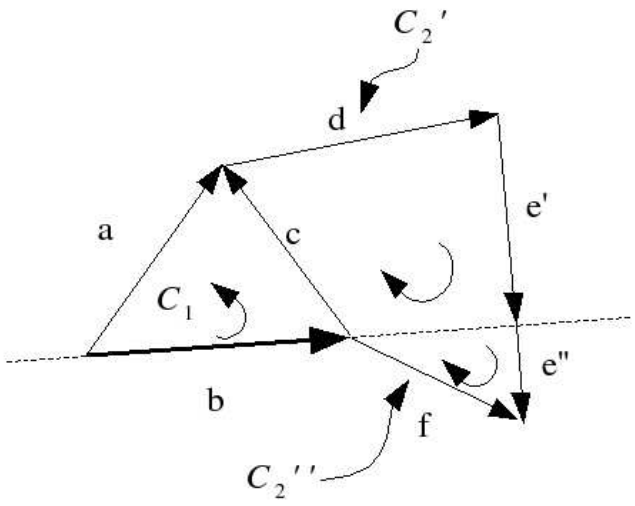


Figure 18: Coherent partitioning

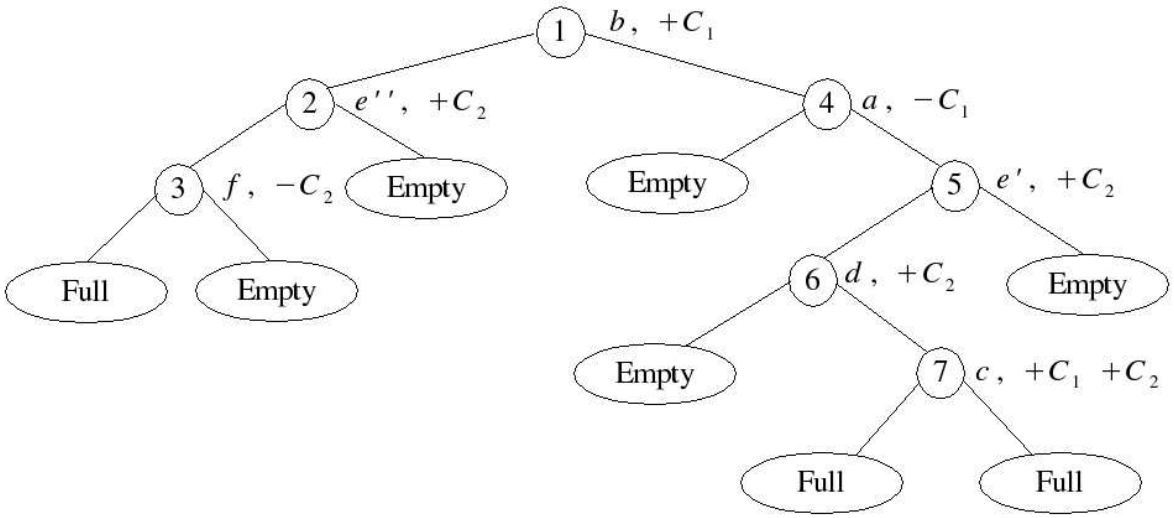


Figure 19: Possible representation using a SBSP tree

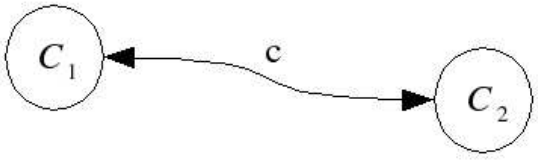


Figure 20: Adjacency graph

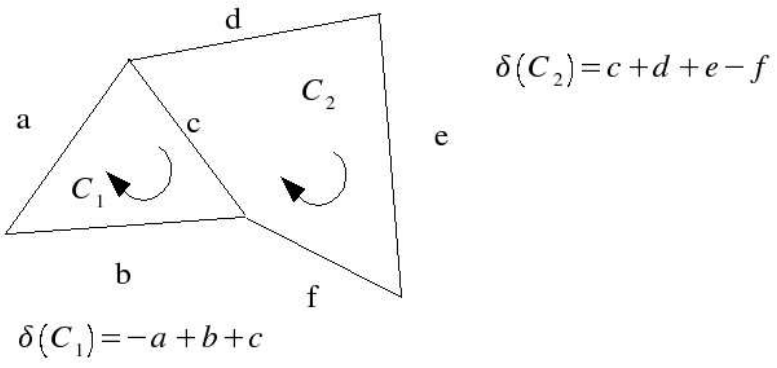


Figure 21: Cochain's coboundary

2. For every node ν, add the coefficient g_ν to every adjacent cell C_ν with sign sgn_{C_ν}
3. Continue on the sub-trees $above(\nu)$ and $below(\nu)$

We can store the coefficients associated to the higher-order cells in the adjacency graph.

8 Stokes theorem and SBSP

With respect to the Stokes theorem, it is possible to split the theorem in two and build a diagram:

$$\int_{\partial R} \omega \quad = \begin{cases} \text{k-cochain} & \xrightarrow{\delta} & (k+1)\text{-cochain} \\ \int \uparrow & & \\ \text{k-form} & & \end{cases}$$

$$\int_{R} d\omega \quad = \begin{cases} & & (k+1)\text{-cochain} \\ & & \uparrow \int \\ \text{k-form} & \xrightarrow{d} & (k+1)\text{-form} \end{cases}$$

The SBSP data structure can handle the first part of the diagram. A SBSP tree represents a k-cochain on the complex ∂R in E^{k+1}. Thus, we can compute the first part of the theorem:

$$\text{k-form} \xrightarrow{\int} \text{k-cochain}$$

The cell-by-cell integral assigns to every k-cell a coefficient deriver from the integration of a k-form: in other words, it creates a k-cochain. We can compute this step first rebuilding the topology of k-cells from the SBSP tree, and then integrate.

The next step is a coboundary operation, possible just with a tree traversal:

$$\text{k-cochain} \xrightarrow{\delta} (k+1)\text{-cochain}$$

The other part of the diagram involves the external derivative and an integral. Since the external derivative is basicly a symbolic operation, the SBSP tree can't help here.

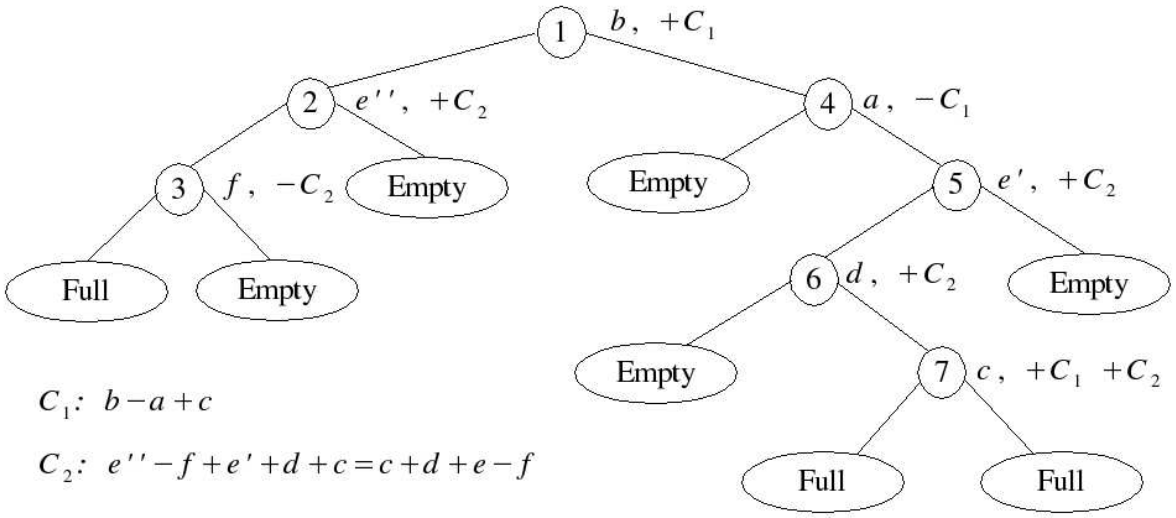


Figure 22: Coboundary calculated over the SBSP tree

It can be useful to reconstruct the topology about the $(k+1)$-cells and so calculate the integral.

9 Final valutations

The use of BSP trees has indeed some advantages. First, BSP trees are a well known data structure, used for a long time in computer graphics. One great advantage of BSP trees is the great flexibility in some trivial operations, such as the inner-outer point problem solving, boolean operations and tree merging.

A little modification to standard BSP trees makes it possible to store cochains, calculate the coboundary, and so to handle field calculus over a cell complex. The amount of memory needed by this structure is very small, this fact suggests its use against a more complicated data structure like the *winged edge* structure.

References

[1] A. A. G. Requicha. Mathematical models of rigid solid models. Production Automation Project, University of Rochester, 1977.

[2] A. A. G. Requicha. Representations for rigid solids: Theory, methods and systems. *ACM Computer Surveys*, 12(4),437–464, 1980.

[3] A. Brondsted. An introduction to convex polytopes. Graduate texts in mathematics, Vol. 90 — Springer-Verlag, New York, NY, 1983.

[4] Alberto Paoluzzi. *Geometric programming for computer aided design.* John Wiley & Sons, 2002.

[5] Allen Hatcher. *Algebraic topology.* Cambridge University Press, 2002.

[6] B. F. Taylor, J. Amantides, W. Thibault. Merging BSP trees yields polyhedral set operations. In *Computer Graphics*, volume 24(4):115–124. ACM Siggraph '90, 1990.

[7] Enzo Tonti. On the formal structure of physical theories. Technical report, Istituto di Matematica del Politecnico di Milano, 1975.

[8] F. H. Branin. The algebraic-topological basis for network analogies and the vector calculus. In *Proceeding of the Symposium on Generalized Networks*, volume 16:453–491. Polytechnic Institute of Brooklyn, 1966.

[9] James R. Munkres. *Elements of algebraic topology.* Addison Wesley Publishing Company, 1984.

[10] Jeffrey A. Chard Vadim Shapiro. A multivector data structure for differential forms and equations. *IMACS Transactions Journal, Mathematics and Computer in Simulation*, 54(1):33–64, 2000.

[11] Manfredo Perdigão do Carmo. *Differential forms and applications.* Springer-Verlag, 1971. English translation (1994).

[12] R. Bott, L. Tu. *Differential forms in algebraic topology.* Springer-Verlag (GTM 82), 1982.

[13] R. Egli, N. F. Stewart. A framework for system specification using chains on cell complexes. *Computer Aided Design*, 32(9):447–459, 2000.

[14] R. W. R. Darling. *Differential forms and connections.* Cambridge University Press, 1994.