Solid and Physical Modeling with Chain Complexes

Antonio DiCarlo *
Dipartimento di Strutture
Università “Roma Tre”
Via Segre, 6
00146 Rome, Italy

Franco Milicchio †
Dip. Informatica e Autom.
Università “Roma Tre”
Via Vasca Navale, 79
00146 Rome, Italy

Alberto Paoluzzi ‡
Dip. Informatica e Autom.
Università “Roma Tre”
Via Vasca Navale, 79
00146 Rome, Italy

Vadim Shapiro §
Dept. of Mech. Eng. & Comp. Science
University of Wisconsin
1513 University Avenue
Madison, WI 53706, USA

Abstract

In this paper we show that the (co)chain complex associated with a decomposition of the computational domain, commonly called a mesh in computational science and engineering, can be represented by a block-bidiagonal matrix that we call the Hasse matrix. Moreover, we show that topology-preserving mesh refinements, produced by the action of (the simplest) Euler operators, can be reduced to multi-linear transformations of the Hasse matrix representing the complex.

Our main result is a new representation of the (co)chain complex underlying field computations, a representation that provides new insights into the transformations induced by local mesh refinements. This paper is a further contribution towards bridging the subject of computer representations for solid and physical modeling—which flourished border-line between computer graphics, engineering mechanics and computer science with its own methods and data structures—under the general cover of linear algebra and algebraic topology. The main advantage of such an approach is that topology, geometry and physics may coexist in one and the same formalized framework, concurring together to define, represent and simulate the behavior of the model.

Our approach is based on first principles and is general in that it applies to most representational domains that can be characterized as cell complexes, without any restrictions on their type, dimension, codimension, orientability, manifoldness, connectedness. Contrary to what might appear at first sight, the theoretical complexity of the present approach is not greater than that of current methods, provided that sparse-matrix techniques with double element access (by rows and by columns) are employed. Last but not least, our tensor-based approach is a significant step forward in achieving close integration of geometrical representations and physics-based simulations, i.e., in the concurrent modeling of shape and behavior.

CR Categories: I.3.3 [Computational Geometry and Object Modeling]: Curve, surface, solid, and object representations; J.6 [Computer-aided design]: CAD

Keywords: topology representation, finite methods, computational mesh, Hasse matrix, split algorithm

1 Introduction

1.1 Motivation

The boundary representation has become the representation of choice in many academic and virtually all commercial solid modeling systems. As a consequence, most geometric, scientific and engineering applications have to be formulated in terms of boundary representations, often leading to nontrivial representation conversion problems. Well known examples of such problems include Boolean set operations, finite element meshing, and subdivision algorithms.

Formally, all boundary representations are widely recognized as graph-based data structures [Baumgart 1972, Guibas and Stolfi 1985, Mäntylä 1988, Brisson 1993] representing one of several possible cell complexes [Requicha 1977, Requicha 1980, Silva 1981, O’Connor and Rossignac 1990]. Space requirements and computational efficiency of such data structures have been studied in the literature (see, e.g., Woo 1985). Historically, such cell complexes have been restricted to (unions of) two-dimensional orientable manifolds, but a number of extensions to more general orientable cellular spaces have been proposed (see, e.g., Masuda 1993, Yamaguchi and Kimura 1995, O’Connor and Rossignac 1990). Depending on a particular choice of data structures, boundary representations are constructed, edited, and updated using a small set of basic operators on the graph representation, while preserving and/or updating the basic topological invariants of the cell complex. Such operators are commonly called Euler operators [Eastman and Weiler 1979, Mäntylä 1988, Masuda 1993], because they enforce the Euler-Poincaré formula. All higher-level algorithms and applications of boundary representations are implemented in terms of such operators.

This evolutionary development of boundary representations also led to several fundamental difficulties:

- Variety of assumptions about the cell complexes and graph representations make standardization difficult. This in turns complicates the issues of data exchange and transfer, and leads to proliferation of incompatible algorithms.
• Boundary representation algorithms are dominated by graph searching algorithms (boundary traversals) that tend to force serial processing. Nor is it clear how to combine such graph representations with multi-resolution representations and algorithms.

• Extending boundary representations to more general cellular spaces has proved challenging. Despite many proposals, most commercial systems are still restricted to two-dimensional orientable surfaces.

• Last, but not least, solid modeling has developed into a highly specialized discipline that is largely disconnected from many standard computational techniques. In particular, boundary representations do not appear to be directly related to the methods for physical analysis and simulation such as finite differences, finite elements, and finite volumes.

In this paper, we claim that all representations of cell complexes are properly represented by a (co)chain complex [Munkres 1984; Hatcher 2002]. It captures all combinatorial relationship of inter-parcel of cells [


In Section 2 we review some standard concepts from algebraic topology and their representations using matrices and Hasse diagram. Section 3 introduces our block-matrix representation of a chain complex. In Section 4 we use algebraic-topological notions to define a minimal set of operators as transformations between cell complexes that preserve the Euler characteristics. These operators are shown to correspond to multi-linear transformations of the Hasse matrix in Section 5. Section 6 demonstrates how common algorithms for splitting a cell complex may be formulated in algebraic topological terms. Section 7 explains how the proposed representation may unify geometric and physical modeling in a common computational framework. The Appendix shows how easily local adjacency information, including the discrete Jacobians, can be computed using only some standard linear algebra.

1.2 Related Work

Algebraic-topological properties of boundary representations are well understood—see [Requicha 1977; Hoffmann 1989; Mántylä 1988; O’Connor and Rossignac 1990] for details. In particular, Brann [Brannin 1966] and Tonti [Tonti 1975] advocated a unified discrete view of all physical theories using concepts from algebraic topology and the De Rham cohomology. More recently, this early research led to new efforts in developing unified computational models and languages for analysis, simulation, and engineering design. Notably, Palmer and Shapiro [Palmer and Shapiro 1992] proposed a unified computational model of engineering systems that relies on concepts from algebraic topology. A number of researchers went beyond the use of chains and cochains as general-purpose data types, considering that a sound numerical method should reflect the algebraic-topological structure of the underlying physical theory in a faithful way. Notably, Strang [Strang 1988] observed that the FEM encodes a pervasive balance pattern, which is at the center of the classification in [Tonti 1975]. Mattiusi [Matiusi 1997] provided interpretations of FEM, FVM, and FDM in terms of the topological properties of the corresponding field theory. Tonti [Tonti 2001] presented his cell method as a direct discrete method, bypassing the underlying continuum model. In [Hyman and Shashkov 1997] FDMs that satisfy desired topological properties are discussed. In our previous work [Milicchio et al. 2006], physical information is attached to an adaptive, full-dimensional decomposition of the computational domain. Giving preeminence to the cells of highest dimension allows to generate the geometry and to simulate the physics simultaneously. Such a formulation removes artificial constraints on the shape of discrete elements and unifies commonly unrelated finite methods into a single computational framework [Milicchio 2007]. Our goal is to graft this approach to field modeling onto an already established computational framework for geometric modeling with cell complexes [Paoluzzi et al. 1995]. This framework has been recently extended to provide parallel and progressive generation of very large datasets using streams of continuous approximations of the domain with convex cells [Scorzelli et al. 2006]. The approach also supports progressive Boolean operations [Paoluzzi et al. 2004a], providing continuous streaming of geometrical features and adaptive refinement of their details.

1.3 Overview

Let $K$ be a cell complex representing a finite partition of a compact set $D \subseteq \mathbb{R}^d$. We call $p$-skeleton $K_p \subseteq K$ the subset of oriented $p$-cells of $K$, and denote with $k_p$ the number of $p$-cells: $k_p := \#K_p$, hence

$$\#K = k_0 + k_1 + \cdots + k_d.$$  

The $p$-skeleton $K_p$ will be ordered by labeling each $p$-cell $\sigma_p$ with a positive integer: $K_p = (\sigma_p^0, \ldots, \sigma_p^{k_p})$. In the following, the ordinal and/or the dimension of cells will be dropped from notation whenever convenient. In its turn, the complex $K$ will be identified with the tuple of its ordered $p$-skeleta ($p = 0, \ldots, d$): $K \cong (K_p)$.  

2.1 Chains and Cochains

Let $(G, +)$ be an abelian (i.e., commutative) group. A $p$-chain of $K$ with coefficients in $G$ is a mapping $c_p : K_p \rightarrow G$ such that, for each $\sigma \in K_p$, reversing a cell orientation changes the sign of the chain value:

$$c_p(-\sigma) = -c_p(\sigma).$$

Chain addition is defined by addition of chain values: if $c_p, d_p$ are $p$-chains, then $(c_p + d_p)(\sigma) = c_p(\sigma) + d_p(\sigma)$, for each $\sigma \in K_p$. The resulting group is denoted $C_p(K; G)$. In the following the group $G$ will often left implied, writing $C_p(K)$.  

2.1.1 Chain groups

Let $(G, +)$ be an abelian (i.e., commutative) group. A $p$-chain of $K$ with coefficients in $G$ is a mapping $c_p : K_p \rightarrow G$ such that, for each $\sigma \in K_p$, reversing a cell orientation changes the sign of the chain value:

$$c_p(-\sigma) = -c_p(\sigma).$$  

Chain addition is defined by addition of chain values: if $c_p, d_p$ are $p$-chains, then $(c_p + d_p)(\sigma) = c_p(\sigma) + d_p(\sigma)$, for each $\sigma \in K_p$. The resulting group is denoted $C_p(K; G)$. In the following the group $G$ will often left implied, writing $C_p(K)$.  

2.1.1 Chain groups  

Let $(G, +)$ be an abelian (i.e., commutative) group. A $p$-chain of $K$ with coefficients in $G$ is a mapping $c_p : K_p \rightarrow G$ such that, for each $\sigma \in K_p$, reversing a cell orientation changes the sign of the chain value:

$$c_p(-\sigma) = -c_p(\sigma).$$

Chain addition is defined by addition of chain values: if $c_p, d_p$ are $p$-chains, then $(c_p + d_p)(\sigma) = c_p(\sigma) + d_p(\sigma)$, for each $\sigma \in K_p$. The resulting group is denoted $C_p(K; G)$. In the following the group $G$ will often left implied, writing $C_p(K)$.  

2.1.1 Chain groups  

Let $(G, +)$ be an abelian (i.e., commutative) group. A $p$-chain of $K$ with coefficients in $G$ is a mapping $c_p : K_p \rightarrow G$ such that, for each $\sigma \in K_p$, reversing a cell orientation changes the sign of the chain value:

$$c_p(-\sigma) = -c_p(\sigma).$$

Chain addition is defined by addition of chain values: if $c_p, d_p$ are $p$-chains, then $(c_p + d_p)(\sigma) = c_p(\sigma) + d_p(\sigma)$, for each $\sigma \in K_p$. The resulting group is denoted $C_p(K; G)$. In the following the group $G$ will often left implied, writing $C_p(K)$.  

2.1.1 Chain groups  

Let $(G, +)$ be an abelian (i.e., commutative) group. A $p$-chain of $K$ with coefficients in $G$ is a mapping $c_p : K_p \rightarrow G$ such that, for each $\sigma \in K_p$, reversing a cell orientation changes the sign of the chain value:

$$c_p(-\sigma) = -c_p(\sigma).$$  

Chain addition is defined by addition of chain values: if $c_p, d_p$ are $p$-chains, then $(c_p + d_p)(\sigma) = c_p(\sigma) + d_p(\sigma)$, for each $\sigma \in K_p$. The resulting group is denoted $C_p(K; G)$. In the following the group $G$ will often left implied, writing $C_p(K)$.  

2.1.1 Chain groups
Let $\sigma$ be an oriented cell in $K$ and $g \in G$. The elementary chain whose value is $g$ on $\sigma$, $-g$ on $-\sigma$ and 0 on any other cell in $K$ is denoted $g\sigma$. Each chain can then be written in a unique way as a finite sum of elementary chains:

$$c_p = \sum_{k=1}^{k_p} g_k\sigma_{p,k}.$$ 

Chains may be thought of as attaching multiplicity to cells; if coefficients are taken from the smallest nontrivial group, i.e. $G = \{-1, 0, 1\}$, then cells can only be discarded or selected, possibly inverting their orientation.

### 2.1.2 Cochain groups

By definition, the set of $p$-cochains of $K$, with coefficients in $G$, is the group of all homomorphisms of $C_p(K)$ into $G$:

$$C^p(K) := \text{Hom}(C_p(K), G)$$

Cochains may be thought of as measuring the content of $G$-valued additive quantities in chains. If $\gamma^p$ is a $p$-cochain, its content in the $p$-chain $c_p$ is often denoted as a pairing:

$$\langle \gamma^p, c_p \rangle := \gamma^p(c_p).$$

### 2.2 Boundary and Coboundary

#### 2.2.1 Boundary operator

The boundary operator $\partial_p : C_p(K) \rightarrow C_{p-1}(K)$ is first defined on simplices. If $\sigma_p$ is a $p$-simplex, then

$$\partial_p \sigma_p := \sum_{k=0}^{p} (-1)^k\sigma_{p-1,k}$$

where $\sigma_{p-1,k}$ denotes the $k$-th face of $\sigma_p$. The next move is to extend $\partial_p$ to cells, by partitioning them into simplices and assuming $\partial_p$ to be additive. This is then extended to elementary chains, by taking

$$\partial_p(g\sigma) := g(\partial_p \sigma)$$

and finally to all chains by additivity.

#### 2.2.2 Coboundary operator

The coboundary operator $\delta^p$ is defined as the dual of the boundary operator $\partial_{p+1} : C_{p+1}(K) \rightarrow C_p(K)$, so that

$$\delta^p : C^p(K) \rightarrow C^{p+1}(K)$$

in such a way that, for all $\gamma \in C^p$ and $c \in C_{p+1}$,

$$\{\delta^p \gamma, c\} = \langle \gamma, \partial_{p+1} c \rangle.$$

The pairing notation makes transparent that this defining property is a combinatorial prototype of the Stokes theorem.

#### 2.2.3 Matrix representation of (co)chains

A very simple and powerful abstraction consists in representing $p$-chains and $p$-cochains as matrices indexed on the cells of $K$ and parameterized in the underlying $G$ group.

Let $K$ be a $d$-complex, with $k_p = \#K_p$, $0 \leq p \leq d$. We may represent a $p$-chain $c_p \in C_p(K)$ as a column matrix $x_p \in G^{k_p}$, and we write $x_p = [c_p]$, or $x_p^\top = [c_p]^\top$. Analogously, we may represent a $p$-cochain $\gamma^p \in C^p(K)$ as a row matrix $y^p \in G^{k_p}$, and we write $y^p = [\gamma^p]^\top$, or $y^p = [\gamma^p]$. The content of the $p$-cochain $\gamma^p$ in the $p$-chain $c_p$ is given by the matrix product

$$y^p x_p = \langle \gamma^p, c_p \rangle.$$

#### 2.2.4 Incidence matrices

The intersection between $p$-cells and $(p + 1)$-cells may be characterized by the $p$-incidence matrix $(a_{ij}^p)$ defined by:

$$a_{ij}^p = 0 \quad \text{if} \quad \sigma_i \cap \sigma_j = \emptyset \quad (\sigma \text{ being the closure of } \sigma)$$

$$a_{ij}^p = \pm 1 \quad \text{otherwise}, \quad \text{with } +1 (-1) \text{ if the orientation of } \sigma_i \text{ is equal (opposite) to that of the corresponding face of } \sigma_j.$$

Of course, the transpose of $(a_{ij}^p)$ describes how $(p + 1)$-cells intersect with $p$-cells.

It is easy to check that $(a_{ij}^p)$ represents through matrix multiplication the action of the boundary operator $\partial_{p+1} : C_{p+1} \rightarrow C_p$, while its transpose represents the action of the coboundary operator $\delta^p : C^p \rightarrow C^{p+1}$:

$$\sum_{j=1}^{k_{p+1}} a_{ij}^p [c_{p+1}]^j = [\partial_{p+1} c_{p+1}]^i,$$

$$\sum_{i=1}^{k_p} a_{ij}^p [y^p]_i = [\delta^p y^p]^j\,.$$

**Example 1** (Boundary and Coboundary). Let the 2-chain $c \in C_2(K)$ be defined by

$$c(\sigma_1) = 1, \quad c(\sigma_2) = 1, \quad c(\sigma_3) = 1, \quad c(\sigma_4) = 1,$$

where $K$ is the 2-complex given in Figure 7. The boundary 1-chain

$$\partial_2 c = \partial_2 (\sigma_1 + \sigma_2 + \sigma_3 + \sigma_4) = \tau_1 + \tau_3 + \tau_4 + \tau_8 + \tau_9$$

is represented by

$$\begin{pmatrix}
1 \\
1 \\
1 \\
1
\end{pmatrix} \begin{pmatrix}
0 \\
1 \\
0 \\
1
\end{pmatrix} = \begin{pmatrix}
1 \\
0 \\
1 \\
0
\end{pmatrix},$$

where the incidence matrix $[\partial_2] = [\delta^1]^\top$, and

$$[\delta^1] = \begin{pmatrix}
1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & -1 & -1 & 1 & 0 & 0
\end{pmatrix}.$$
In order theory, a Hasse diagram is a graph \( \mathcal{H} = (N, E) \), where \( N \) is a finite poset, such that for any \( x, y \in N \), there exists \( (x, y) \in E \) if and only if \( x < y \), and there is no \( z \in N \) such that \( x < z < y \).

If, given a \( d \)-complex \( K \), the sets \( N \) and \( E \) are defined as follows, then the graph \( \mathcal{H}(K) = (N, E) \) provides a complete representation of the topology of \( K \):

1. \( N := K_0 \cup K_1 \cup \cdots \cup K_d \),
2. \( E := E_1 \cup \cdots \cup E_d \), with
3. \( E_p := \{ (\sigma_p, \sigma_{p-1}) \mid \sigma_{p-1} \in \partial \sigma_p \}, 1 \leq p \leq d \).

Attaching a label from \( \{-1, 1\} \) to the arc \((x, y) \in E_p \), denoted \( \text{sgn}(x, y) \), suffices to specify the relative orientation between the \( p \)-cell represented by the node \( x \) and the \((p - 1)\)-cell represented by the node \( y \).

Given a Hasse graph \( \mathcal{H}(K) = (N, E) \), with \( N = \cup_p K_p \), for each node \( x \in N \) define:

1. \( E^x := \{ (x, y) \} \mid y \in N, (x, y) \in E \}, \)
2. \( N^x := \{ y \mid y \in N, (x, y) \in E^x \} \).

Let \( \sigma \in K \) be the cell represented by the node \( x \). Then, the boundary of the elementary chain \( g \sigma \) is obtained by transferring the (properly signed) coefficient from the node \( x \) to its “children” in \( \mathcal{H}(K) \):

\[
\partial(g \sigma) = g(\partial \sigma) = g \sum_{y \in N^x} \text{sgn}(x, y) \tau(y) = \sum_{y \in N^x} \text{sgn}(x, y) g \tau(y)
\]

Where \( \tau(y) \) denotes the cell represented by the node \( y \). The computation of the boundary \( \partial c \) of a general chain \( c \) follows by linearity.

### 2.3.1 Chain and cochain complexes

A Hasse diagram, together with the above representation of the boundary operator \( \partial \), is a convenient representation of a chain complex, whose formal definition is as follows.

A chain complex \( C = (C_p, \partial_p) \) is a sequence

\[
\cdots \longrightarrow C_{p+1} \xrightarrow{\partial_{p+1}} C_p \xrightarrow{\partial_p} C_{p-1} \longrightarrow \cdots \longrightarrow C_1 \xrightarrow{\partial_1} C_0
\]

of abelian groups \( C_p \), paired with homomorphisms \( \partial_p, p \geq 1 \), that satisfies the relation \( \partial_p \circ \partial_{p+1} = 0 \), for each \( p \geq 1 \).

The dual cochain complex \( C' = (C^p, \delta^p) \) is the sequence

\[
\cdots \longleftarrow C^{p+1} \xleftarrow{\delta^{p+1}} C^p \xleftarrow{\delta^p} C^{p-1} \longleftarrow \cdots \longleftarrow C^1 \xleftarrow{\delta^0} C_0
\]

The relations \( \delta^p \circ \delta^{p-1} = 0 \) are satisfied by duality.

### 2.3.2 Chain maps

Let \( C(C_p, \partial_p) \) and \( Ĉ(Ĉ_p, \partial_Ĉ) \) be two chain complexes. A chain map \( \phi : C \rightarrow Ĉ \) is a \( p \)-family of homomorphisms

\[
\phi_p : C_p \rightarrow Ĉ_p
\]

such that \( \partial_Ĉ \circ \phi_p = \phi_{p-1} \circ \partial_p \), i.e., the following diagram is commutative:

\[
\begin{array}{ccc}
C_p & \xrightarrow{\phi_p} & Ĉ_p \\
\partial_p & \downarrow & \partial_Ĉ \\
C_{p-1} & \xrightarrow{\phi_{p-1}} & Ĉ_{p-1}
\end{array}
\]

### 3 Matrix representation

In this section we introduce a block-matrix representation of the topology of the chain complex associated to a decomposition of the computational domain, and call it Hasse matrix. Later we show that, since all blocks transform by a given pattern of transformations, so also the Hasse matrix transforms by the same pattern.

#### 3.1 Block-Matrix Decomposition

A chain complex \( C(C_p, \partial_p) \) and its dual \( C'(C^p, \partial^p) \) can be represented by a block-bidiagonal matrix. Since the boundary operators \( \partial_p (p \geq 1) \) are well represented by incidence matrices and the coboundary operators \( \delta^{p-1} \) by their transposes, we may represent the \( p \)-families of homomorphisms \( \delta_p, \delta^{p-1} (p \geq 1) \) by a block-structured matrix. Notice that, from now on, we shall often write \( \delta_p \) instead of \( \delta^p \).

Let \( K \) be a \( d \)-complex and \( \mathcal{H}(K) \) its Hasse graph. The Hasse matrix will have the block structure shown in Figure 2.

![Figure 1: A 2-complex K, whose 2-cells are coherently oriented.](image)

![Figure 2: The whole scheme holds for d odd; for d even, the last block-row should be discarded.](image)
of independent refining (coarsening) operators for a d-space that do not change its Euler characteristic has to increase (decrease) by one both \( k_{p-1} \) and \( k_p \), for \( p \in \{1, \ldots, d\} \). There are therefore \( d \) elementary refining operators and the same number of elementary coarsening operators.

In order to change the Euler characteristic, i.e. to change the shape of a space, it is appropriate to use some Boolean operator, according to the properties [Alexandrov 1998, Baez 2003] recalled below.

### 4.1.1 Properties of the Euler characteristic

Let \( \chi(M) \) and \( \chi(N) \) be the Euler characteristics of any two topological spaces \( M \) and \( N \). Then, their sum is the Euler characteristic of the disjoint union of \( M \) and \( N \):

\[
\chi(M \cup N) = \chi(M) + \chi(N).
\]

More generally, if \( M \) and \( N \) are subspaces of a larger space \( X \), then so are their union and intersection, and the Euler characteristic obeys the rule:

\[
\chi(M \cap N) = \chi(M) + \chi(N) - \chi(M \cup N).
\]

Moreover, the Euler characteristic of any product space is

\[
\chi(M \times N) = \chi(M) \chi(N).
\]

### 4.2 Make and Kill operations

The simplest Euler operators that transform a cell complex \( K \) into a new complex \( \tilde{K} \) without changing its Euler characteristic \( \chi \), add (remove) just two cells to (from) the complex, with dimensions \( p \) and \( (p+1) \). They will be denoted as \( \beta \) and \( \kappa \), from the Greek words “blastos” and “klastos”, referring to construction and destruction, respectively.

By definition, the operator \( \beta^p \) adds a \( p \)-cell and a \( (p+1) \)-cell to \( K \), thus transforming it into \( \tilde{K} \). The reverse operator \( \kappa^q \) deletes a \( p \)-cell and a \( (p-1) \)-cell.

In this section we discuss how the coboundary operators transform under the action of a refinement operation \( \beta^q \):

\[
\delta_p \circ \beta^q : \delta_p(K) \mapsto \delta_q(\beta^q(K)) , \quad p = 0, \ldots, n-1.
\]

It is easily seen that \( \beta^q \) affects in a nontrivial way only the coboundary operators whose domain and/or codomain change under its action, namely:

1. \( \delta_q \mapsto \bar{\delta}_q \)
2. \( \bar{\delta}_q \mapsto \delta_{q+1} \)
3. \( \delta_{q+1} \mapsto \bar{\delta}_{q+1} \)

as shown by the commutative diagram:

\[\begin{array}{ccccccc}
\tilde{C}^{q+2} & \xrightarrow{\delta_{q+1}} & C^{q+1} & \xrightarrow{\beta^q} & C^q & \xleftarrow{\delta_q} & \tilde{C}^q \\
\bar{\delta}_{q+1} & \xleftarrow{\delta_{q+1}} & \tilde{C}^{q+2} & \xrightarrow{\delta_q} & C^{q+1} & \xleftarrow{\delta_q} & C^q
\end{array}\]

Three different computations have to be performed, depending on whether only the domain changes (case 1), or only the codomain (case 2), or both change (case 3).
4.2.1 Addition of a column \((\delta_{q+1} \mapsto \tilde{\delta}_{q+1})\)

Let the matrix \([\delta_{q+1}]\) be \(m \times n\); then, the matrix \([\tilde{\delta}_{q+1}]\) will be \(m \times (n + 1)\). The column to be added to \([\tilde{\delta}_{q+1}]\) represents the cochain in \(\beta^q(C^{q+2})\) incident on the new cell \(\tilde{\sigma}_{q+1}\). It is a linear combination of the columns of \([\tilde{\delta}_{q+1}]\), i.e., of the preexistent chains in \(C^{q+2}\). We have:

\[
[\tilde{\delta}]_{m \times (n+1)} = [\delta]_{m \times n} \begin{pmatrix} I_{n \times n} & c_1 & \cdots & c_n \end{pmatrix} = [\delta]_{m \times n} C
\]

4.2.2 Addition of a row \((\delta_{q-1} \mapsto \tilde{\delta}_{q-1})\)

The row to be added to \([\delta_{q-1}]\) represents the chain of \(\beta^q(C_{q-1})\) incident on the new cell \(\tilde{\sigma}_q\). It is a linear combination of the rows of \([\delta_{q-1}]\). We have:

\[
[\tilde{\delta}]_{(m+1) \times n} = \begin{pmatrix} I_{m \times m} \\ \vdots \\ I_{m \times m} \end{pmatrix} \cdot [\delta]_{m \times n} = R[\delta]_{m \times n}
\]

4.2.3 Addition of a column and a row \((\delta_q \mapsto \tilde{\delta}_q)\)

One of the rows of \([\delta_q]\) (one chain in \(C_q\)) is substituted by two rows (two chains in \(\beta^q(C_q)\)), whose components on the added cell \(\tilde{\sigma}_q\) sum up to zero. The matrix \([\tilde{\delta}_q]\) is obtained as the sum

\[
[\tilde{\delta}_q]_{(m+1) \times n} = \sum_{i=1}^{3} S_i [\delta_q]_{m \times n} T_i,
\]

where the first term \((i = 1)\) provides the contribution of the split cell \(\sigma_{q+1}\), the second one \((i = 2)\) the contribution of the added cell \(\tilde{\sigma}_{q+1}\), and the third one \((i = 3)\) the contribution of all of the other cells in \(K_{q+1}\).

4.3 Examples

In Figs. 4-6 we show a very simple 2-complex \(K\) and its refinement \(\tilde{K}\), obtained by applying first the operator \(\beta^0\) to split the 1-cell \(\sigma_1\), then the operator \(\beta^1\) to split the 2-cell \(\sigma_2\).

![Figure 4: Coarse complex \(K = (K_0, K_1, K_2)\).](image)

Let us compute the matrix representation of the coboundary operators \(\delta_0, \delta_1\), on \(K\) and on their refinements \(\tilde{K} = \beta^0(K)\) and \(\tilde{\tilde{K}} = \beta^1(\tilde{K})\). The boundary operators \(\delta_0, \delta_1, \delta_2\), as well as their refinements, are obtained by transposition.

**Example 5** (Coboundary \(\delta_0 : C^2(K) \to C^1(K)\)). Both domain and codomain have dimension 3. From Figure 4 it is seen that the matrix representation of \(\delta_0\) is

\[
[\delta_0] = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix}.
\]

**Example 6** (Coboundary \(\delta_1 : C^1(K) \to C^0(K)\)). In this case we have \(k_1 = 3\) and \(k_2 = 1\), so that

\[
[\delta_1] = \begin{pmatrix} -1 & 1 & 1 \end{pmatrix}.
\]

**Example 7** (Coboundary \(\delta_0 : C^0(\tilde{K}) \to C^1(\tilde{K})\)). We have \(k_0 = k_1 = k_2 = 3 + 1\). In Figure 6, the new 0-cell and 1-cell are displayed in red. Since both domain and codomain dimensions increase, the new operator has to be computed as the sum of three contributions (see Section 4.2.3).

\[
[\tilde{\delta}_0] = \begin{pmatrix} S_1 & S_2 & S_3 \end{pmatrix} [\delta_0] \begin{pmatrix} T_1 \\ T_2 \\ T_3 \end{pmatrix}
\]

where \(\begin{pmatrix} S_1 & S_2 & S_3 \end{pmatrix}\) and \(\begin{pmatrix} T_1 \\ T_2 \\ T_3 \end{pmatrix}^\top\) are block-matrices, and

\[
S_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad S_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad S_3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}.
\]

Matrices \(S_1, S_2\) extract the row of \([\delta_0]\) that corresponds to the 1-cell \(\sigma_1\) to be split (recall that a row of \([\delta_0]\) equals a column of \([\delta_1]\)); \(S_3\) associates that row to \(\tilde{\sigma}_1\), while \(S_2\) associates it to the added cell \(\tilde{\sigma}_2\); matrix \(S_3\) keeps all other rows of \([\delta_0]\) unchanged. The actions
of \( S_1 \), \( S_2 \), and \( S_3 \) on \( \delta_0 \) are explicitly given below:

\[
S_1 [\delta_0] = \begin{pmatrix}
-1 & 0 & 1 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{pmatrix}, \quad S_2 [\delta_0] = \begin{pmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
-1 & 0 & 1
\end{pmatrix}, \quad S_3 [\delta_0] = \begin{pmatrix}
0 & 0 & 0 \\
-1 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix}.
\]

Each column of matrix \( T_i \) (\( i = 1, 2, 3 \)) corresponds to a 1-cell in \( K_1 \). Each \( T_i \) matrix represents the linear transformation that maps one or more chains of \( K_0 \) elements into the corresponding chains of \( K_0 \) elements:

\[
S_1 [\delta_0] T_1 = \begin{pmatrix}
-1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix},
\]

\[
S_2 [\delta_0] T_2 = \begin{pmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1
\end{pmatrix},
\]

\[
S_3 [\delta_0] T_3 = \begin{pmatrix}
-1 & 1 & 0 & 0 \\
0 & -1 & 1 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}.
\]

In conclusion, we get:

\[
[\tilde{\delta}_1] = \begin{pmatrix}
-1 & 0 & 0 & 1 \\
-1 & 1 & 0 & 0 \\
0 & -1 & 1 & 0 \\
0 & 0 & 1 & -1
\end{pmatrix}.
\]

The reader may check this result looking at Figure 8.

**Example 8 (Coboundary \( \bar{\delta}_1 : C^1(\bar{K}) \to C^2(\bar{K}) \)).**

In this case, \( k_1 = 3 + 1 \) and \( k_2 = 1 \); one gets:

\[
[\tilde{\delta}_1] = [\delta_1] C = [\delta_1] \begin{pmatrix}
1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0
\end{pmatrix} = \begin{pmatrix}
-1 & 1 & 1 & -1
\end{pmatrix}.
\]

**Example 9 (Coboundary \( \tilde{\delta}_0 : C^0(\bar{K}) \to C^1(\bar{K}) \)).**

We have: \( k_0 = k_2 = 4 \), \( k_1 = k_1 + 1 = 5 \), and we get:

\[
[\tilde{\delta}_0] = [\tilde{\delta}_0] = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & -1
\end{pmatrix} \begin{pmatrix}
-1 & 0 & 0 & 1 \\
-1 & 1 & 0 & 0 \\
0 & -1 & 1 & 0 \\
0 & 0 & 1 & -1
\end{pmatrix} = \begin{pmatrix}
-1 & 0 & 0 & 1 \\
-1 & 1 & 0 & 0 \\
0 & -1 & 1 & 0 \\
0 & 0 & 1 & -1
\end{pmatrix}.
\]

**Example 10 (Coboundary \( \bar{\delta}_1 : C^1(\bar{K}) \to C^2(\bar{K}) \)).**

Now we have \( k_1 = k_1 + 1 = 5 \) and \( k_2 = k_2 + 1 = 2 \). Since both domain and codomain dimensions increase, by performing the same operations as in Example 7 we get:

\[
[\tilde{\delta}_1] = \begin{pmatrix}
\delta_1 \\
S_1 \\
S_2 \\
S_3
\end{pmatrix} \begin{pmatrix}
T_1 \\
T_2 \\
T_3
\end{pmatrix} \begin{pmatrix}
R^T \\
[\delta_1] \end{pmatrix}
\]

\[
H(K) \Rightarrow H(\bar{K}),
\]

\[
\eta^b_K : \mathcal{M}_m^m \to \mathcal{M}_m^{m+1},
\]

such that

\[
H(K) \Rightarrow H(\bar{K}),
\]

where the \( (p + 1) \)-cell \( \sigma^b_{p+1} \) is split by the blastos (or “make”) \( \beta^\ast \) operator into two cells:

\[
\bar{\sigma}^b_{p+1} \quad \text{and} \quad \bar{\sigma}^{(b_{p+1}+1)},
\]

and a new \( p \)-cell \( \sigma^b_{p+1} \) is added to the complex. Notice that, while \( m \) and \( n \) increase under topology-preserving refinements, their difference does not. Let us distinguish between even and odd values of \( d \), and assume, without loss of generality, that \( d = 3 \). In this case there are two diagonal blocks \( [\delta_0] \cdot [\delta_2] \) and one upper-diagonal block \( [\delta_1]^T \) in \( H \) (see Section 3) **Remark 1 (Make operators \( \beta^0, \beta^1 \) and \( \beta^2 \)).**

Different but similar computational patterns arise, depending on the order of the make operator:

\[
\beta^0(H) = \begin{pmatrix}
( S_1 \quad S_2 \quad S_3 ) [\delta_0] \begin{pmatrix}
T_1 \\
T_2 \\
T_3
\end{pmatrix} \\
R [\delta_1]^T
\end{pmatrix},
\]

\[
\beta^1(H) = \begin{pmatrix}
R [\delta_0] \\
( S_1 \quad S_2 \quad S_3 ) [\delta_2]^T \begin{pmatrix}
T_1 \\
T_2 \\
T_3
\end{pmatrix}
\end{pmatrix},
\]

\[
\beta^2(H) = \begin{pmatrix}
0 \\
[\delta_0] \\
[\delta_1]^T \\
( S_1 \quad S_2 \quad S_3 ) [\delta_0]^T \begin{pmatrix}
T_1 \\
T_2 \\
T_3
\end{pmatrix}
\end{pmatrix}.
\]

In 3D the only make operators are \( \beta^0, \beta^1, \beta^2 \). Each \( \beta^\ast \) inserts two new cells \( \bar{\sigma} \) and \( \bar{\sigma}_{p+1} \) into \( \bar{K} \). In order to specify the corresponding Hasse transformations, we need to extract the diagonal and upper-diagonal blocks of \( H \):

\[
H = \begin{pmatrix}
[\delta_0] \\
[\delta_1]^T
\end{pmatrix} + \begin{pmatrix}
0 \\
[\delta_0] \\
[\delta_1]^T
\end{pmatrix}
\]

Then, we need only to apply the elementary transformations already given for a single operator, and to add the resulting matrices:

\[
\beta^\ast(H) = \beta^\ast(H_1) + \beta^\ast(H_2).
\]
6 Hyperplane splitting

In this section we discuss a subdivision algorithm (SPLIT) developed by Bajaj and Pascucci in [Bajaj and Pascucci 1996], rephrasing it in terms of the algebraic machinery developed in the previous sections. This algorithm works efficiently on a single d-cell of a d-complex. Our algebraic formulation is general and easy to implement using standard packages for sparse-matrix computation [Davis 2006].

The SPLIT algorithm is a useful tool for refining cell complexes, providing the ability to compute Boolean operations when combined with BSP trees in a progressive way [Paoluzzi et al. 2004a]. The SPLIT algorithm is also useful to approximate continuous maps between cell complexes. A formal definition of subdivision of a complex goes this way [Munkres 1984]:

Definition 1. Let \( K \) be a cell complex. Then, a complex \( \tilde{K} \) is a subdivision of \( K \) if:

1. for each \( \tilde{c} \in \tilde{K} \) there exists \( c \in K \) such that \( \tilde{c} \subseteq c \);
2. for each \( c \in K \), there exists a finite subset \( \{ \tilde{c} \} \subseteq \tilde{K} \), such that \( c = \bigcup \tilde{c} \).

The SPLIT algorithm—to be detailed in the following—generates a subdivision, since for every cell \( \tilde{c} \in \tilde{K} \) we have by construction \( \tilde{c} \subseteq c \in K \). Property 2 is also satisfied, since every cell in \( K \) is mapped into the union of at most two halves \( \tilde{c}^- \) and \( \tilde{c}^+ \), produced by the operation \( \bar{K} \).

6.1 The split algorithm

Let us first introduce two auxiliary operators, to be used for the matrix formulation of the SPLIT algorithm.

Definition 2 (Sign function).
The operator \( \text{sgn}_\epsilon : \mathbb{R}^d \rightarrow \{-1, 0, 1\}^d \) returns the matrix listing the signs of the elements \( v_i \) of a d-tuple \( v = (v_i) \), taking into consideration the numerical tolerance \( \epsilon > 0 \):

\[
(\text{sgn}_\epsilon v)_j = \begin{cases} 
-1, & v_j < -\epsilon \\
0, & -\epsilon \leq v_j \leq \epsilon \\
1, & v_j > \epsilon 
\end{cases}
\]

Definition 3 (Absolute value function).
The function \( \text{abs} \) operates on a matrix \( M = (m_{ij}) \) returning the matrix of the absolute values of its elements:

\[
\text{abs} \ M = ((|m_{ij}|))
\]

Consider the splitting hyperplane \( h \) characterized by the equation \( \sum_p h_p x_p = b \) as a linear (affine homogeneous) form \( \mathbb{R}^{d+1} \rightarrow \mathbb{R} \), represented by the row-matrix

\[
h = \begin{pmatrix} h_1 & h_2 & \ldots & h_d & -b \end{pmatrix}.
\]

Let \( v \) be the column-matrix representation formed by the homogeneous coordinates of the 0-cell \( \sigma_0 \):

\[
v = \begin{pmatrix} x_1 & x_2 & \ldots & x_d & 1 \end{pmatrix}^T.
\]

Clearly, \( \sigma_0 \) belongs to the above subspace \( h^+ \) if and only if \( h(\sigma_0) > 0 \), while it belongs to the below subspace \( h^- \) if and only if \( h(\sigma_0) < 0 \). The sign of the scalar product \( h \cdot v \) solves the point location problem.

Introducing the matrix

\[
V = \begin{pmatrix} v_1 & v_2 & \cdots & v_{k_0} \end{pmatrix}
\]

algorithm SPLIT (input: \( K, V, h \); output: \( \tilde{K}, \tilde{V} \));

1. \( p := 0 \)
2. Classify the 0-cells: \( c_0 := \text{sgn}_\epsilon(hV) \)
3. \( p := p + 1 \)
4. Classify the \( p \)-cells and find their “face” class:
   \[
   c_p := (\text{abs} [\delta_{p-1}]) c_{p-1}
   \]
   \[
   f_p := (\text{abs} [\delta_{p-1}]) (\text{abs} c_{p-1})
   \]
5. \( \text{foreach} \ |c_p| \neq f_p^e \text{ do} \) Update the cell complex:
   Split the \( e \)-th \( p \)-cell: \( K := \beta^{p-1}(K) \);
   Set the new element value: \( \sigma^{p-1}_V := 0 \)
6. Re-classify the \( p \)-cells of the updated cell complex:
   \( \sigma := \text{sgn}_\epsilon(hV) \)
7. \( \text{if} p < d \text{ then goto step 3, else stop.} \)

Figure 7: The SPLIT algorithm, implemented by using a classification chain and the coboundary operator.

that collects the homogeneous coordinates of all the 0-cells in \( K_0 \), their classification with respect to the \( h \) splitting hyperplane is coded by the 0-chain \( c : K_0 \rightarrow \{-1, 0, 1\} \), represented by the matrix:

\[
c_0 = \text{sgn}_\epsilon(hV).
\]

The SPLIT algorithm proceeds hierarchically from 0-cells up to \( d \)-cells by (a) classifying the cells with respect to the splitting hyperplane, and (b) updating the cell complex accordingly, including the new elements in the skeletons of all orders. The algorithm is sketched in Figure 7.

Remark 2. For each dimension \( p \), the absolute value \( |c_p^e| \) of \( c_p(\sigma^e) \) is compared with the value \( f_p^e = f_p(\sigma^e) \) (step 5). In fact, the only \( p \)-cells that intersect the splitting hyperplane \( h \) are characterized by the inequality \(|c_p^e| \neq f_p^e\).

6.2 Split example

Let us go back to the splitting example already discussed in Section 6.1, and refine the 2-complex with the hyperplane specified in Fig 8. The reader should recall Figs. 3, 4, and 5 and refer to them to locate by name the cells of the complex.

Figure 8: (a) The splitting hyperplane \( h \), and (b) the classification of vertices.

The SPLIT algorithm is initialized by setting \( p = 0 \) and by classifying the vertices through the 0-chain

\[
c_0 = \text{sgn}_\epsilon(h \begin{pmatrix} v_1 & v_2 & v_3 \end{pmatrix}) = \begin{pmatrix} -1 & 0 & 1 \end{pmatrix},
\]

as shown in Figure 8(a). Then \( p \) is increased to 1 and 1-cells are classified by computing the 1-chains:

\[
c_1 = (\text{abs} [\delta_0]) c_0 = \begin{pmatrix} 0 & -1 & 1 \end{pmatrix},
\]
\[
f_1 = (\text{abs} [\delta_0]) (\text{abs} c_0) = \begin{pmatrix} 2 & 1 & 1 \end{pmatrix}.
\]

Results are illustrated in Fig 8(b) we see that \( \sigma_1^+ \) should be split, since \(|c_1^+| \neq f_1^+\).
Let us denote the support space of the complex $\mathcal{K}$.

### 6.3 Subdivision of a complex

The application of the $\beta^0$ operator adds a new 0-cell (classified to 0) and a new 1-cell (see Figs. 10). The two 1-cells resulting from the split one, as shown in Fig. 10b, are reclassified.

Figure 10: The updated cell complex, with 1-cells reclassified.

Then, $p$ is increased to 2 and 2-cells are classified:

\[
e_2 = \text{abs } [\delta_1] e_1 = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} -1 & -1 & 1 & 1 \end{pmatrix} = 0
\]

\[
f_2 = \text{abs } [\delta_1] \text{abs } e_1 = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix} = 4
\]

(see Fig. 11a). Hence, the $\sigma^1$ cell gets split, the splitting being executed by the $\beta^1$ operator that creates one 1-cell and one 2-cell, as shown in Figs. 11b. Finally the algorithm reclassifies the 2-cells and terminates, since $p = d$. The result is illustrated in Fig. 11c, where the 2-chain generated on the refined complex $\tilde{\mathcal{K}}$ is illustrated.

Figure 11: (a) Classification of the 2-cells, (b) the classification 1-chain on the refined 1-skeleton, and (c) the refined 2-skeleton with the classification 2-chain.

Therefore, the induced chain map can be applied either to the subdivided chain or to the original chain complex, since

\[
\partial \circ \zeta = \zeta \circ \partial.
\]

The chain map can be summarized in the following commutative diagram:

\[
\cdots \to C_p \xrightarrow{\partial} C_{p-1} \to \cdots
\]

\[
\cdots \to \tilde{C}_p \xrightarrow{\partial} \tilde{C}_{p-1} \to \cdots
\]

As a result, boundaries in the refined cell complex $\tilde{\mathcal{K}}$ may be computed by applying the chain map $\zeta$ to boundaries evaluated in the coarse cell complex $\mathcal{K}$.

### 7 Geometry & physics modeling

The (co)chain-complex formalism and the Hasse-matrix representation generalize in a natural and straightforward way to physical modeling. Chains assign measures to cells, measures that may be tuned to represent the physical properties of interest (mass, charge, conductivity, stiffness, and so on). Cohomology, on the other side, may be used to represent all physical quantities associated to cells through integration with respect to a measure. The coboundary operator stays behind the basic structural laws (balance and compatibility) involving physically meaningful cochains [34, 25, 28]. It is also well known that $k$-cochains are the coarse-grained analogue of differential $k$-forms [36, 10]. Correspondingly, the cochain complex introduced in Section 2.3.1 is a discrete version of the De Rham complex [7, 23, 2], naturally represented by the Hasse matrix (or its transpose).

This view on physical modeling has been increasingly advocated [6, 2, 16] as a way to increase numerical stability and accuracy of various numerical methods. Even more important is the that a proper use of the Hasse matrix has the potential to bring both geometric and physical modeling within a unified computational framework. According to its definition (see Section 3.1), $H(K)$ provides a compact representation of purely topological operators, boundary $\partial$ and coboundary $\partial$, acting on chains or cochains defined on $K$. Such a representation is mediated by a metric structure which embodies far more information than the topology of the cell complex $K$ plus the measure-like properties imparted to it by the introduction of chains. This additional structure is brought in by the seemingly innocuous identification between elementary chains and elementary cochains. However, the “obvious” cell-wise identification we have performed in Section 2.2.3, is associated with a conventional metric structure, easy to use on $K$, but totally unrelated—in general—to the metric properties relevant to the physics under consideration. Of course, the underlying topology stays untouched. Therefore, as long as one is only interested in having an easy-to-use metrical representation of topological operators, the metric involved is instrumental and one is allowed to use whichever is found convenient. Nevertheless, when the object itself, not only its representation, does depend on the metric—as when introducing the notion of adjacency between cells and the related notion of Laplacian (see the Appendix)—then it is essential to import into the model the relevant, physics-based metric structure, through a well-tuned identification of chains with cochains. As a consequence, the elementary chain $\sigma^1$ will not be identified—in general—with the elementary cochain $1 \sigma$. Approaching these issues is basic to gain the possibility of transferring information from $K$ to its refinement $\tilde{K}$.

A deeper discussion on metric issues is out of scope; we simply stress here that the same data structures and algorithms may be used both for solid modeling and physics-based simulations. From our
vantage point, boundary representations and finite element meshes appear as two different aspects of the same Hasse representation. Furthermore, there is no fundamental distinction between different types of approximation methods: in [21, 20], by telling apart the metrical and topological properties embodied in the Hasse representation, we showed that all linear problems formulated by all finite methods are basically equivalent. Within our framework, the split algorithm described in Section 6 becomes a powerful method for progressive refinement not only of shapes, but also of the representation of fields living on those shapes.

8 Conclusions

Historically, the development of boundary representation schemes in solid modeling was driven by limited computational resources, and the usual space-time trade-offs [Woo 85]. A typical boundary representation was chosen (a) to save memory, when RAM was small and expensive, and (b) to spare disk access times, by giving efficient answers to topological queries. Contrary to what might appear at first sight, the present approach does not imply higher theoretical complexity, since the number of non-zero elements in the Hasse matrix $H(K)$ is essentially of the same order as the number of adjacency pointers in a typical graph-based representation of the cell complex $K$. Furthermore, the Hasse matrix serves as a unifying standard for all boundary representations; the difference between different graph structures amount to different methods [Davis 2006] for encoding a subset of the sparse matrix $H(K)$.

We also note that the chain complex is a standard tool for representing and analyzing topological properties of arbitrary cellular spaces. It follows that the proposed Hasse matrix and transformations may codify much more general models, without restrictions on orientability, (co)dimension, manifoldness, connectivity, homology, and so on. The resulting framework, centered on a matrix representation of the domain of interest, unifies several geometric and physical finite formulations, and supports local progressive refinement and coarsening. This approach is inspired by the applications to be developed within the next generation of computational sciences. In particular, the new “big science” of life need simulation models of field problems where geometric and physical properties are generated, detailed, and refined simultaneously and progressively.

References


The symmetric matrices $a_{ij}$ of a 1-complex $K$ are the representations of a graph's adjacency matrices. In Appendix: Adjacency matrices, $A$, $P$, and $G$ are defined. The well-known relation between the incidence matrix of a graph, its transpose and the adjacency matrix of its vertices can be generalized to boundary and coboundary operators of every order, and to the adjacency of $p$-cells in $K_p$, for any dimension $p$.

The topology of the 3-complex $K$ depicted in Fig. 12 is represented by the matrices $[\delta_0] = [\delta_1]^\top$, $[\delta_2] = [\delta_2]^\top$, and $[\delta_3] = [\delta_3]^\top$.

**Definition 4.** The symmetric matrices

$$[\delta_{p+1}][\delta_p] \quad \text{and} \quad [\delta_{p-1}][\delta_p]$$

define the adjacency between $p$-cells through $(p+1)$-cells and $(p-1)$-cells, respectively.

While leaving to the reader the straightforward construction of these matrices, we stress here that such a representation makes use of the standard metric on $K$, by which each elementary chain $1\sigma$ is identified with the elementary cochain $1\sigma$. The metric information introduced in this way becomes important when introducing and computing adjacency matrices, which imply the successive application of the boundary and coboundary operators (or vice versa).

It is worth mentioning that the discrete Laplace-De Rham operators

$$[\delta_{p+1}][\delta_p] + [\delta_{p-1}][\delta_p]$$

are just sums of adjacency matrices. They depend essentially on the metric carried by the matrix representation of boundary and coboundary operators.

### A Appendix: Adjacency matrices

In graph theory, the adjacency matrix of vertices is one of the possible representations of a graph $G = (N, E)$ which is, by definition, a 1-complex $K = (K_0, K_1)$. The adjacency matrices are defined as follows:

$$A_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$$
Figure 12: A 3-complex $K := (K_0, K_1, K_2, K_3)$ and its adjacency matrices.