Homology and Cohomology Solver in Gmsh and its Applications

Matti Pellikka

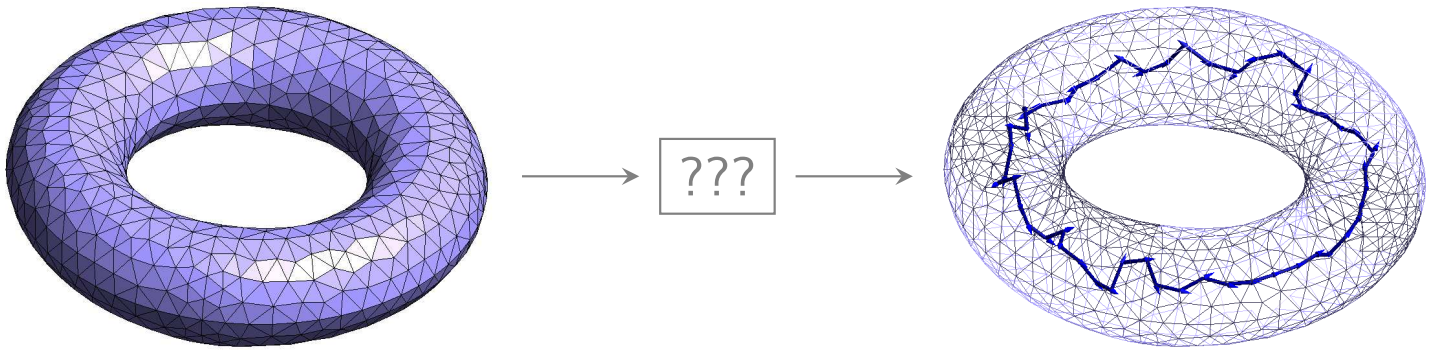
Tampere University of Technology, Electromagnetics, Finland

Second Gmsh Workshop, 2013

Introduction

- Homology and cohomology are global topological features of a domain: holes of various dimension

- Such topological features can be turned into properties of *groups* and *homomorphisms* between them
  - By the discretization of the domain (i.e. a mesh) we have computational problem
- Homology and cohomology solver implementation in Gmsh:



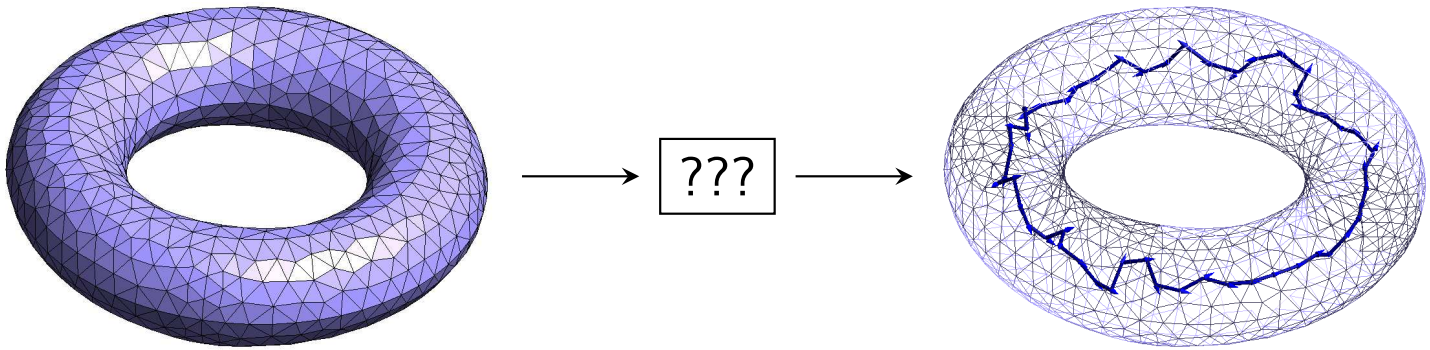
- Most notable application in solution of boundary value problems

Doughnut images by WestportWiki and Rainer Zenz, via Wikimedia Commons (Creative Commons), cropped

Introduction

- Homology and cohomology are global topological features of a domain: holes of various dimension

- Such topological features can be turned into properties of *groups* and *homomorphisms* between them
  - By the discretization of the domain (i.e. a mesh) we have computational problem
- Homology and cohomology solver implementation in Gmsh:

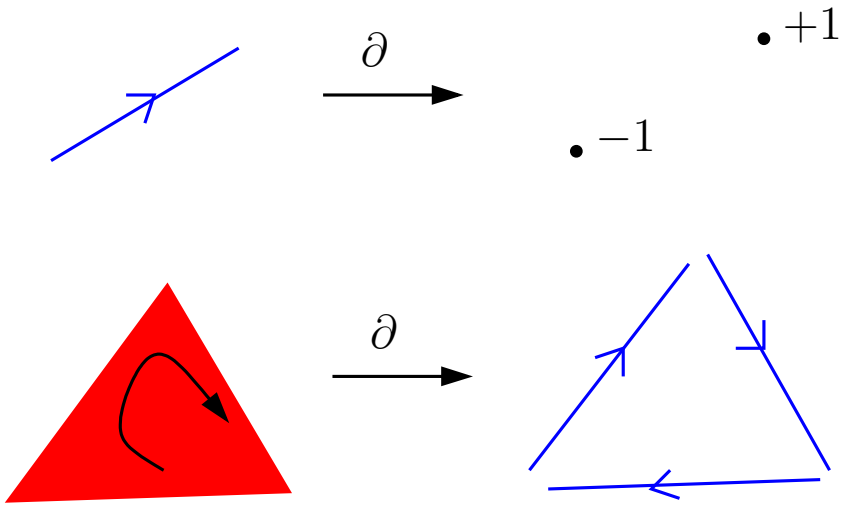


- Most notable application in solution of boundary value problems

Doughnut images by WestportWiki and Rainer Zenz, via Wikimedia Commons (Creative Commons), cropped

Finite element mesh is a cell decomposition of a domain

- A finite element mesh can be considered as a set of oriented volumes, faces, edges, and vertices: k-cells of the mesh
- Boundary of a k-cell is an integer combination of $k-1$-cells



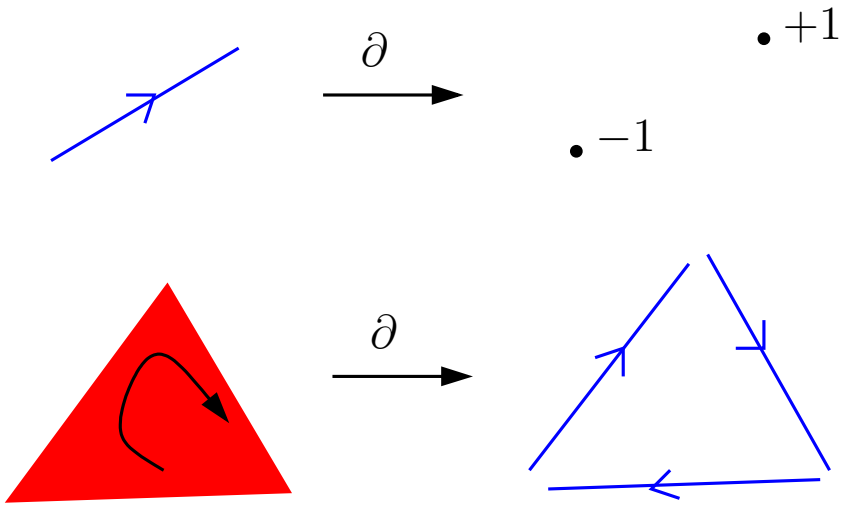
- We consider *integer combinations* of k-cells σ_i

$$c = \sum_i z_i \sigma_i, \quad z_i \in \mathbb{Z}$$

  - Represented by the integer vector $\mathbf{z}$
  - Boundary operator ∂ is represented by an integer matrices D_k that operate on $\mathbf{z}$: $\partial c \to \mathrm{D}_k \mathbf{z}$

Finite element mesh is a cell decomposition of a domain

- A finite element mesh can be considered as a set of oriented volumes, faces, edges, and vertices: k-cells of the mesh
- Boundary of a k-cell is an integer combination of $k-1$-cells



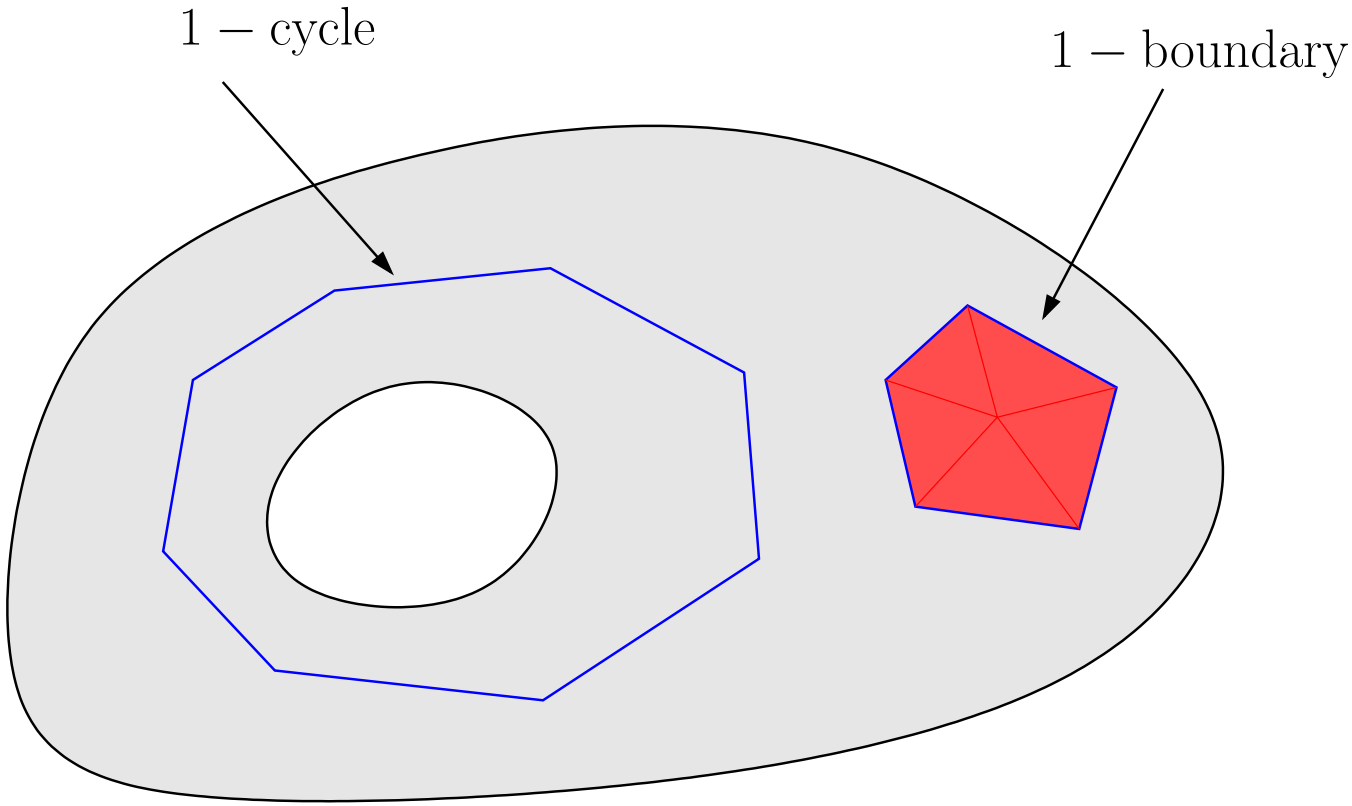
- We consider *integer combinations* of k-cells σ_i

$$c = \sum_i z_i \sigma_i, \quad z_i \in \mathbb{Z}$$

  - Represented by the integer vector $\mathbf{z}$
  - Boundary operator ∂ is represented by an integer matrices D_k that operate on $\mathbf{z}$: $\partial c \to \mathrm{D}_k \mathbf{z}$

Cycles and boundaries

- A k-cycle is an integer combination of k-cells whose boundary vanishes: $\mathrm{D}_k \mathbf{z} = 0$ holds
- A k-boundary is a k-cycle which bounds an integer combination of $k+1$-cells: $\mathbf{z} = \mathrm{D}_{k+1}\mathbf{q}$ holds for some $\mathbf{q}$

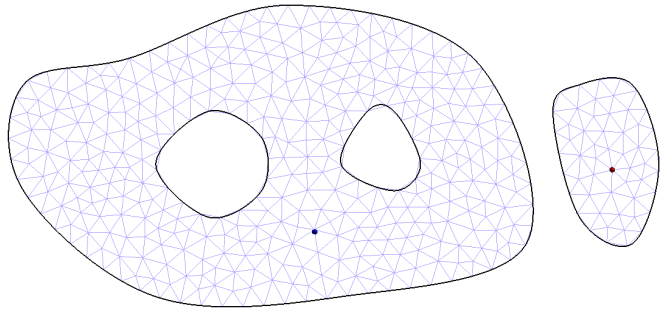


What is homology

- A k-Homology group H_k contains classes of *non-bounding* k-cycles in a domain
- I.e. classes of k-cycles that are *not* k-boundaries
  - H_0: one class per a connected component
  - H_1: one class per a tunnel
  - H_2: one class per a void

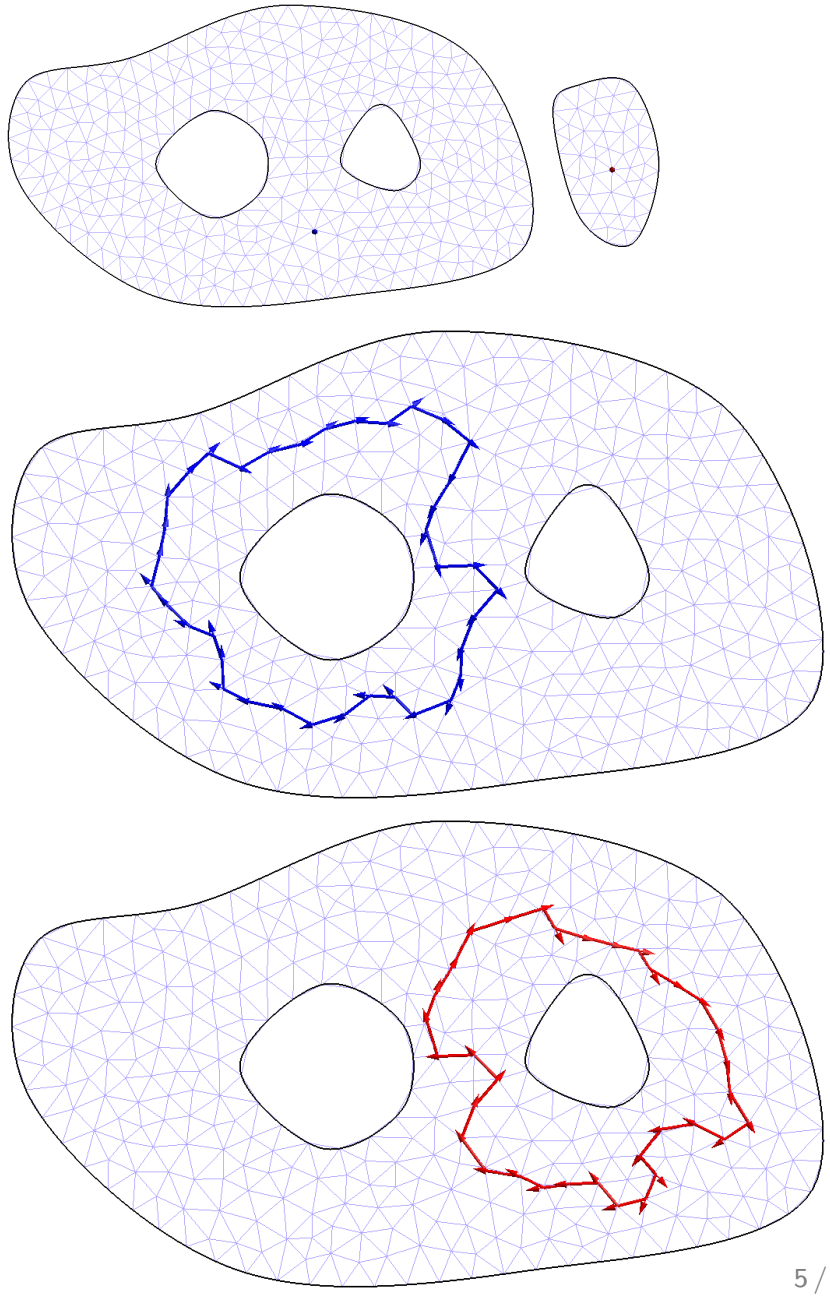
What is homology

- A k-Homology group H_k contains classes of *non-bounding* k-cycles in a domain
- I.e. classes of k-cycles that are *not* k-boundaries
  - H_0: one class per a connected component
  - H_1: one class per a tunnel
  - H_2: one class per a void



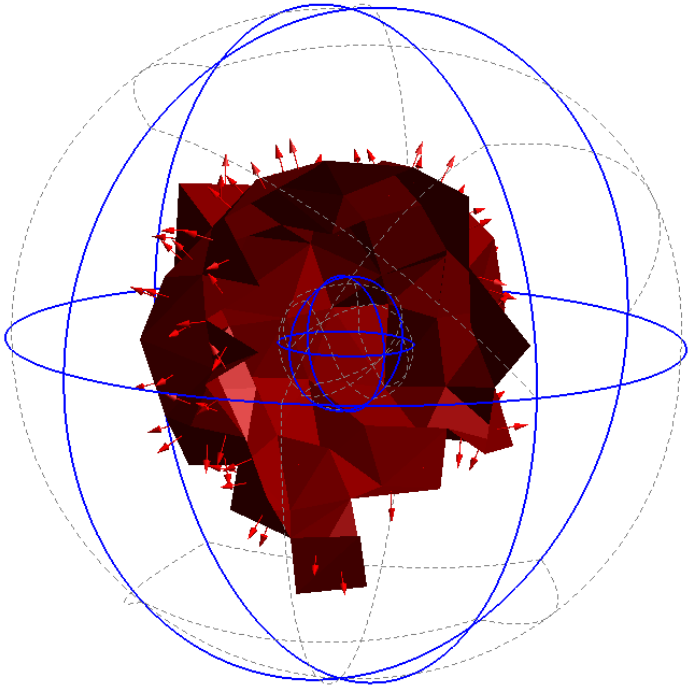
What is homology

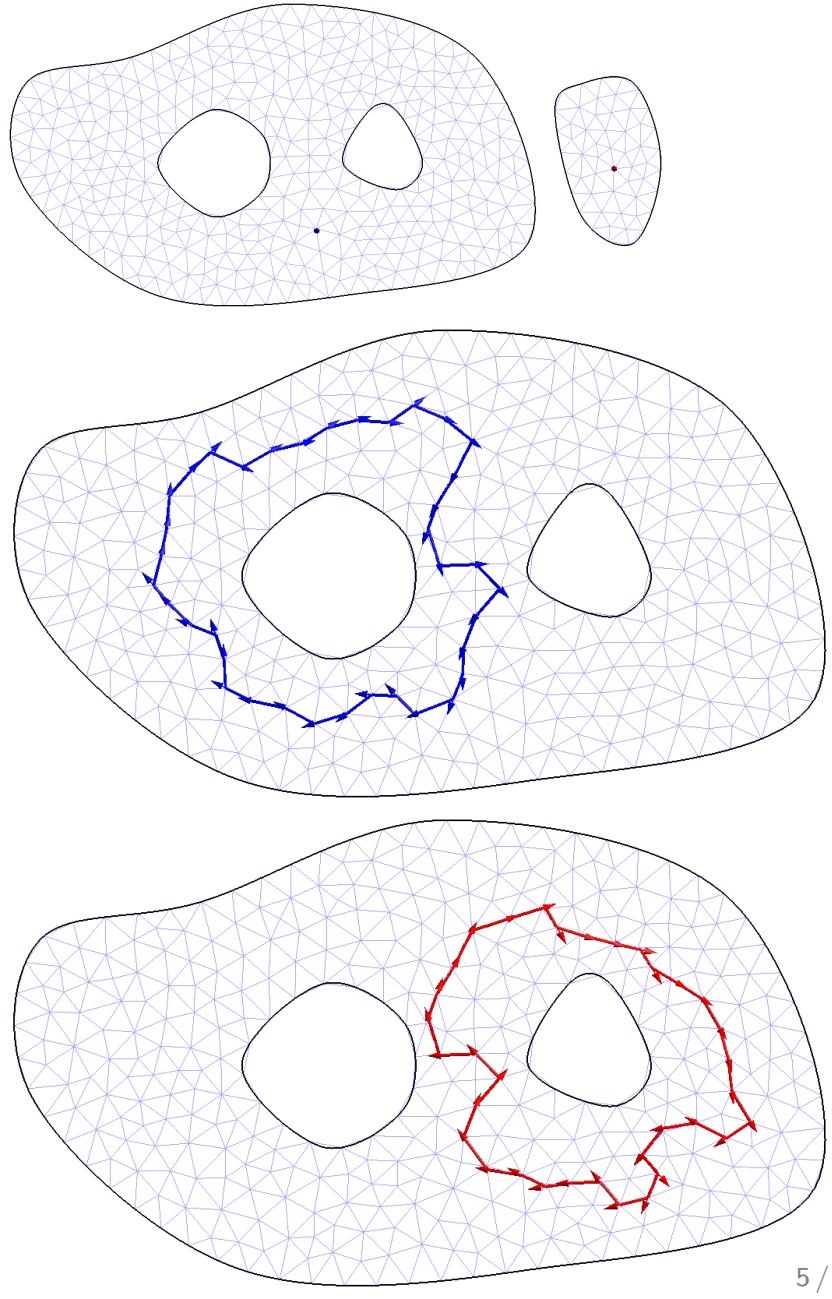
- A k-Homology group H_k contains classes of *non-bounding* k-cycles in a domain
- I.e. classes of k-cycles that are *not* k-boundaries
  - H_0: one class per a connected component
  - H_1: one class per a tunnel
  - H_2: one class per a void



What is homology

- A k-Homology group H_k contains classes of *non-bounding* k-cycles in a domain
- I.e. classes of k-cycles that are *not* k-boundaries
  - H_0: one class per a connected component
  - H_1: one class per a tunnel
  - H_2: one class per a void



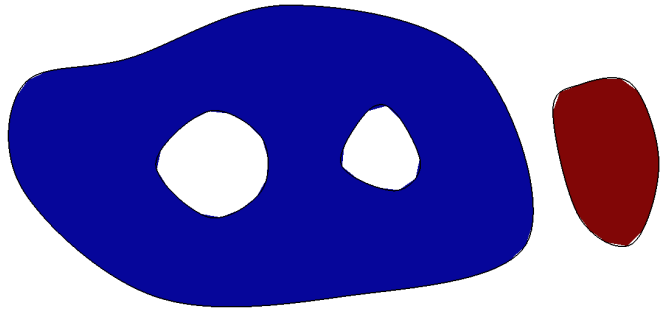


What is cohomology

- A k-Cohomology group H^k contains classes of fields whose integral depends only on the homology class of the integration domain
- I.e.
  - H^0: scalar fields that are constant in each connected component
  - H^1: vector fields whose line integrals only depend on the homology class of the integration curve
  - H^2: vector fields whose surface integrals only depend on the homology class of the integration surface

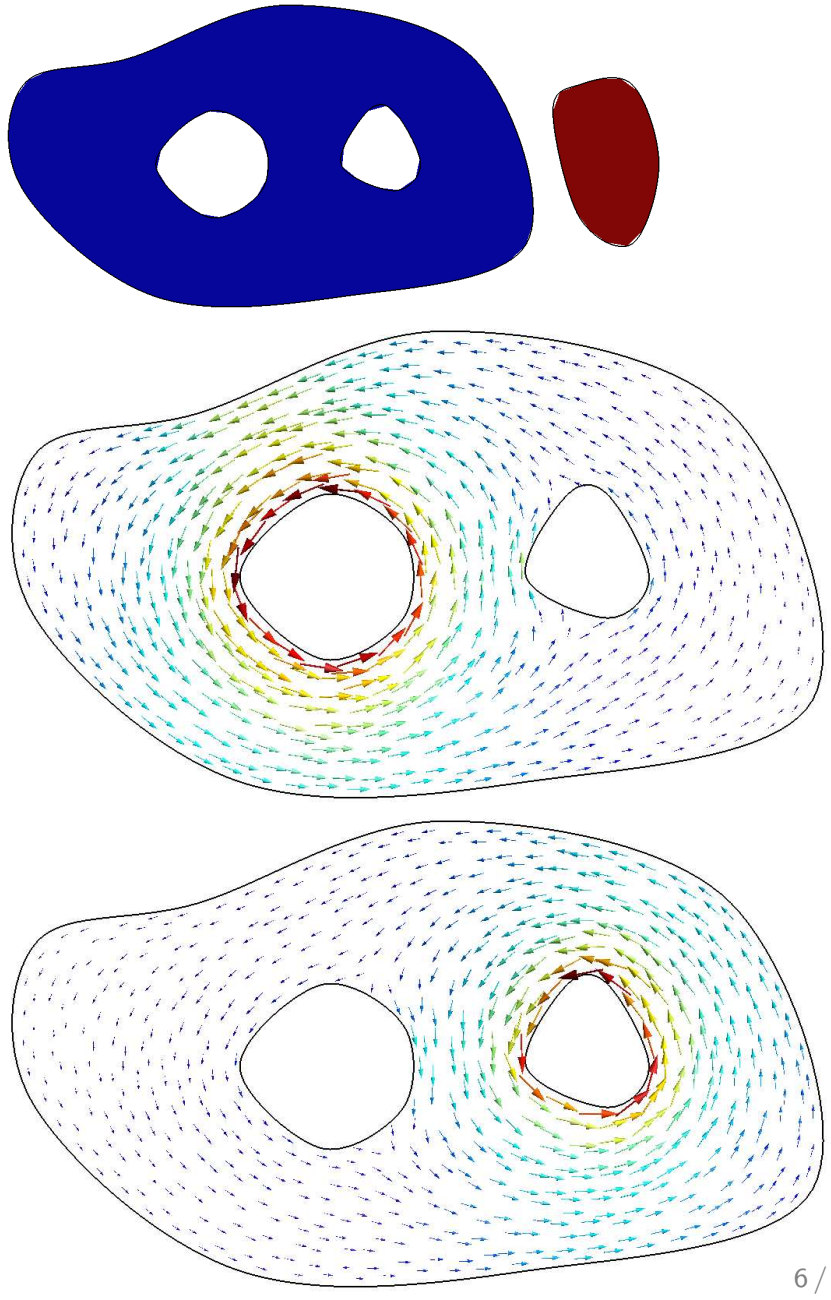
What is cohomology

- A k-Cohomology group H^k contains classes of fields whose integral depends only on the homology class of the integration domain
- I.e.
  - H^0: scalar fields that are constant in each connected component
  - H^1: vector fields whose line integrals only depend on the homology class of the integration curve
  - H^2: vector fields whose surface integrals only depend on the homology class of the integration surface



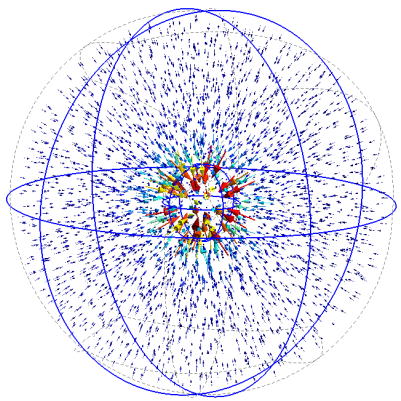
What is cohomology

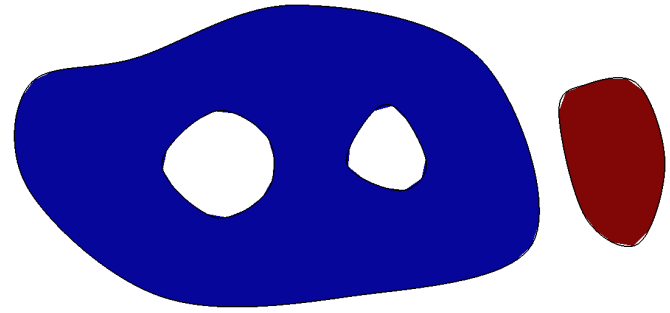
- A k-Cohomology group H^k contains classes of fields whose integral depends only on the homology class of the integration domain
- I.e.
  - H^0: scalar fields that are constant in each connected component
  - H^1: vector fields whose line integrals only depend on the homology class of the integration curve
  - H^2: vector fields whose surface integrals only depend on the homology class of the integration surface

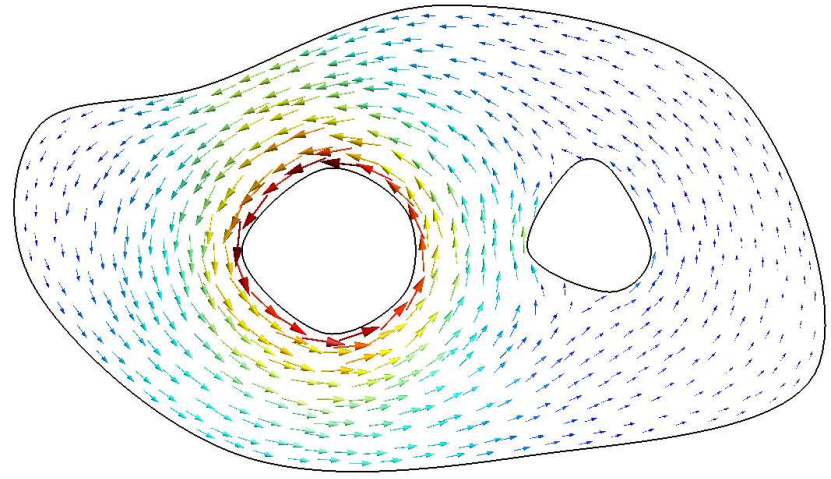


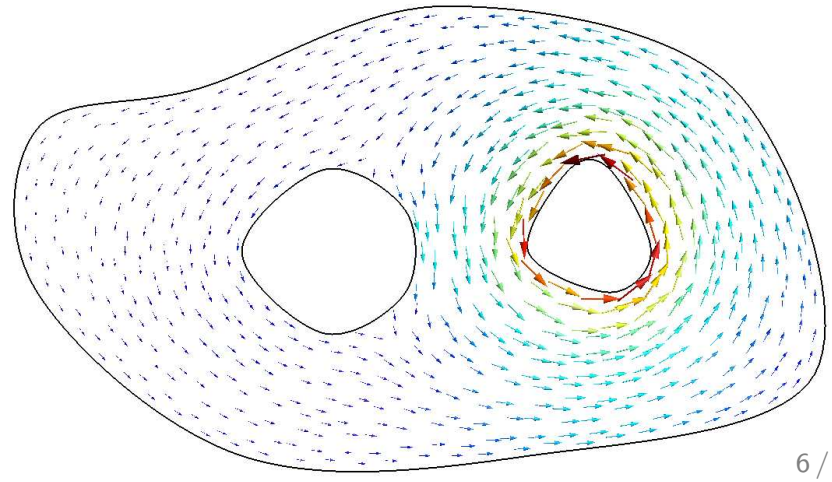
What is cohomology

- A k-Cohomology group H^k contains classes of fields whose integral depends only on the homology class of the integration domain
- I.e.
  - H^0: scalar fields that are constant in each connected component
  - H^1: vector fields whose line integrals only depend on the homology class of the integration curve
  - H^2: vector fields whose surface integrals only depend on the homology class of the integration surface









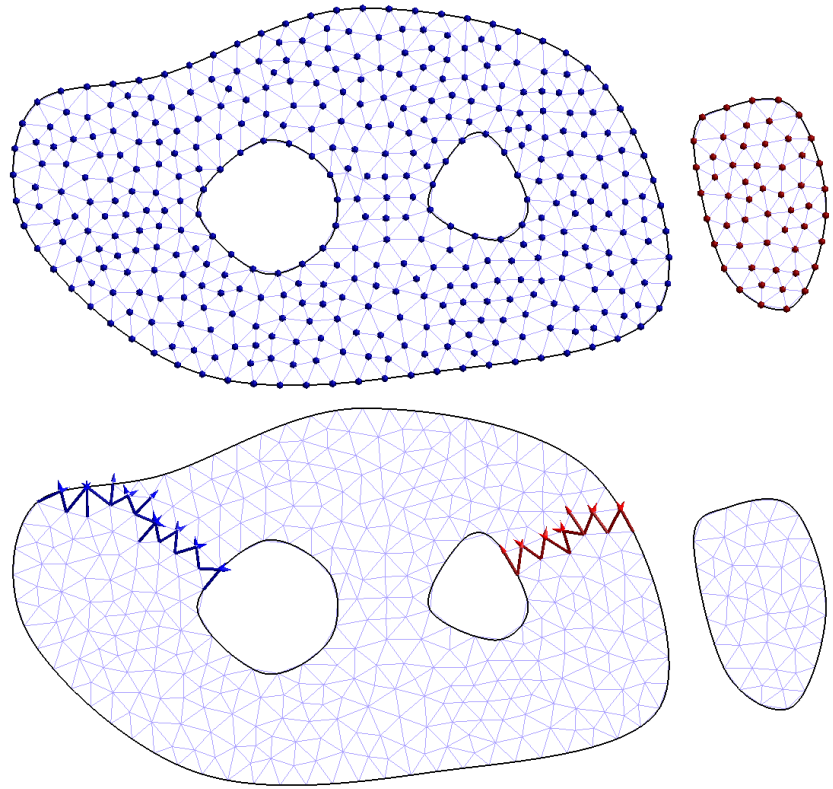
Homology and cohomology solver in Gmsh

- Input for the solver is the finite element mesh
  - Physical group tags designate the computation domain and possibly the relative subdomain
- Homology solver finds a representative k-cycle from each class
- Cohomology solver finds a set of vertices, edges, or faces whose shape functions, edge elements, and facet elements represent a cohomology field

$$f = \sum_i z_i \mathsf{n}^i,$$

$$\mathbf{F} = \sum_i z_i \bar{\mathsf{e}}^i,$$

$$\mathbf{F} = \sum_i z_i \bar{\mathsf{f}}^i$$



- Results are stored as physical groups in the mesh

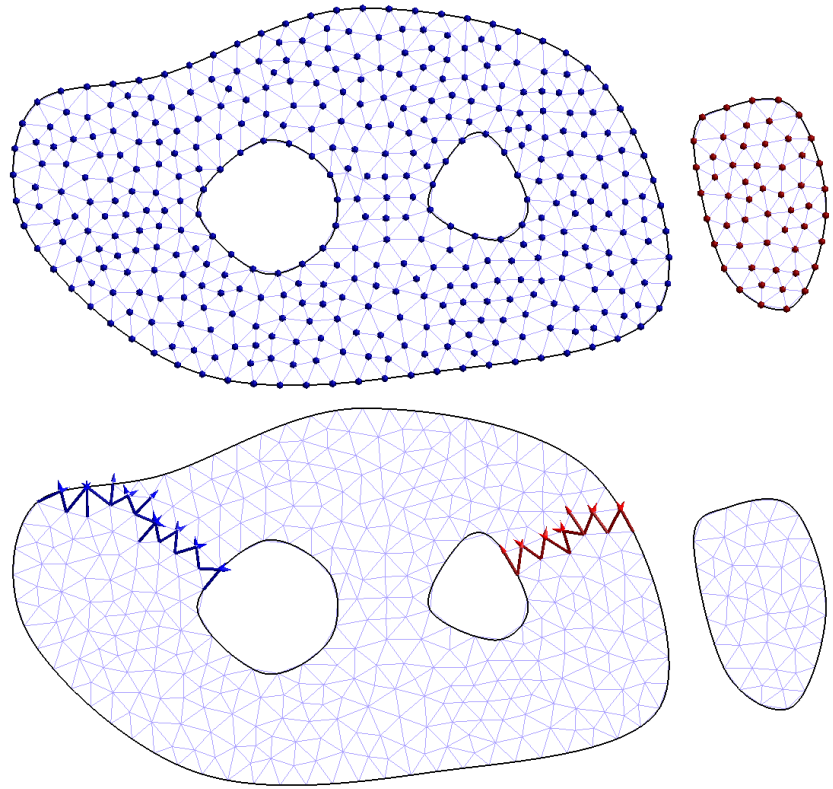
Homology and cohomology solver in Gmsh

- Input for the solver is the finite element mesh
  - Physical group tags designate the computation domain and possibly the relative subdomain
- Homology solver finds a representative k-cycle from each class
- Cohomology solver finds a set of vertices, edges, or faces whose shape functions, edge elements, and facet elements represent a cohomology field

$$f = \sum_i z_i \mathsf{n}^i,$$

$$\mathbf{F} = \sum_i z_i \bar{\mathsf{e}}^i,$$

$$\mathbf{F} = \sum_i z_i \bar{\mathsf{f}}^i$$



- Results are stored as physical groups in the mesh

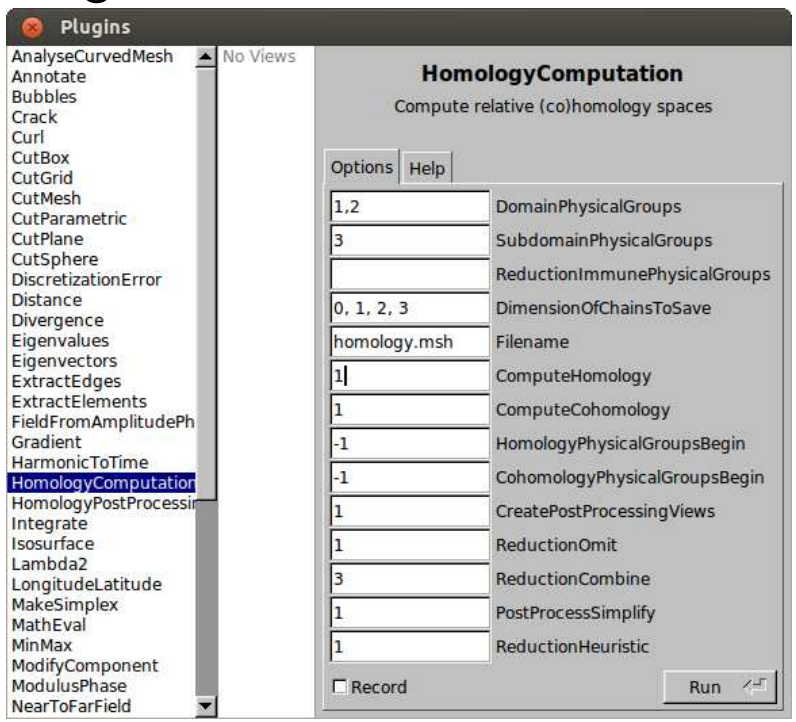
Usage of the solver

1. In .geo-files:

```
// ... geometry definitions

Homology {{1,2},{3}};
Cohomology {{4},{}};
```

2. Plugin:



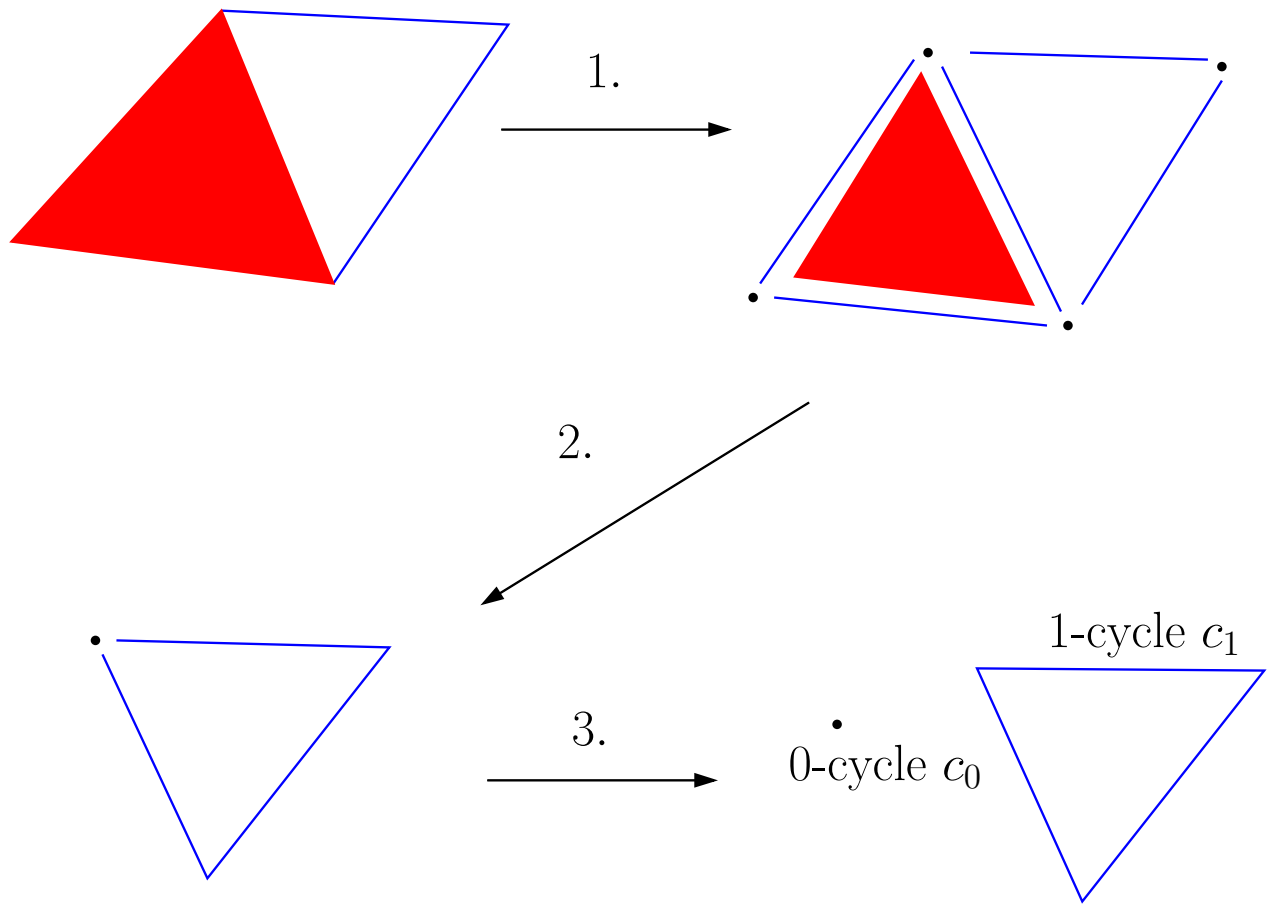
3. API:

```
GModel *m = new GModel();
m->readMSH(``model.msh'');

Homology myHomology(m, domain, subdomain);
myHomology.findHomologyBasis();
myHomology.addChainsToModel(1);
m->writeMSH(``model_hom.msh'');
```

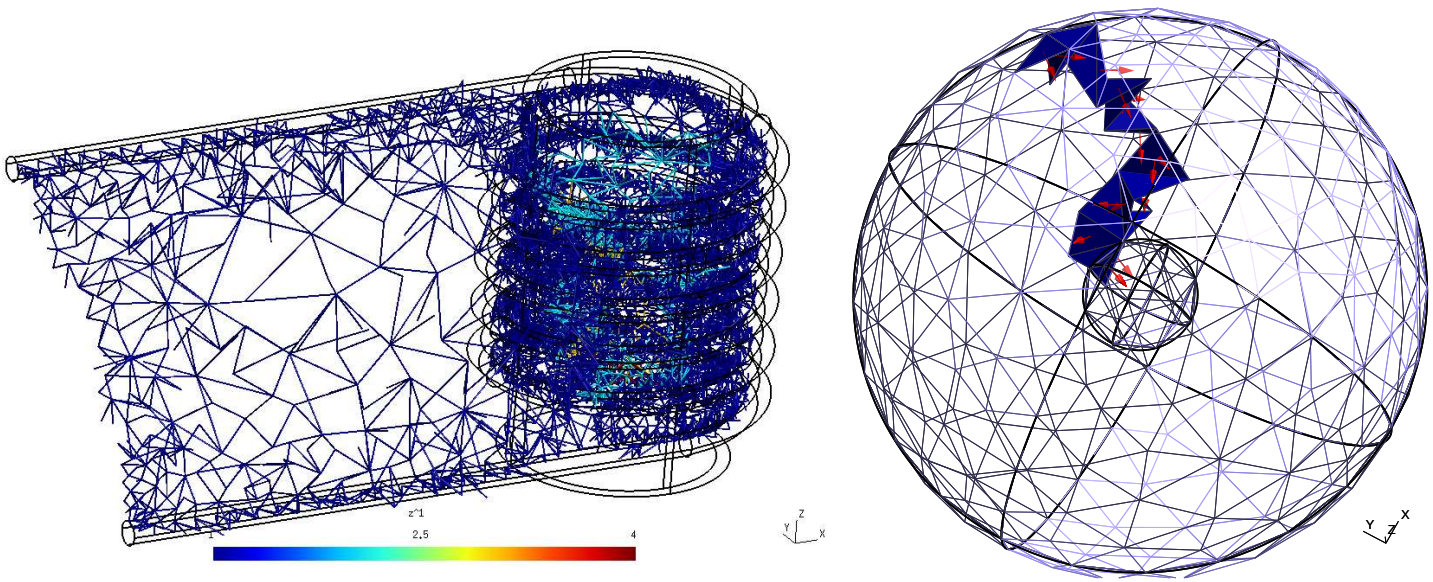
Some details about the solver

- Homology and cohomology computation has three stages
  1. Construction of a cell complex, $O(n \log n)$
  2. Reduction of a cell complex, $\sim O(n \log n)$
  3. Solution of an algebraic problem using integer arithmetic, $O(n^3)$, but usually $n < 1000$ at this stage



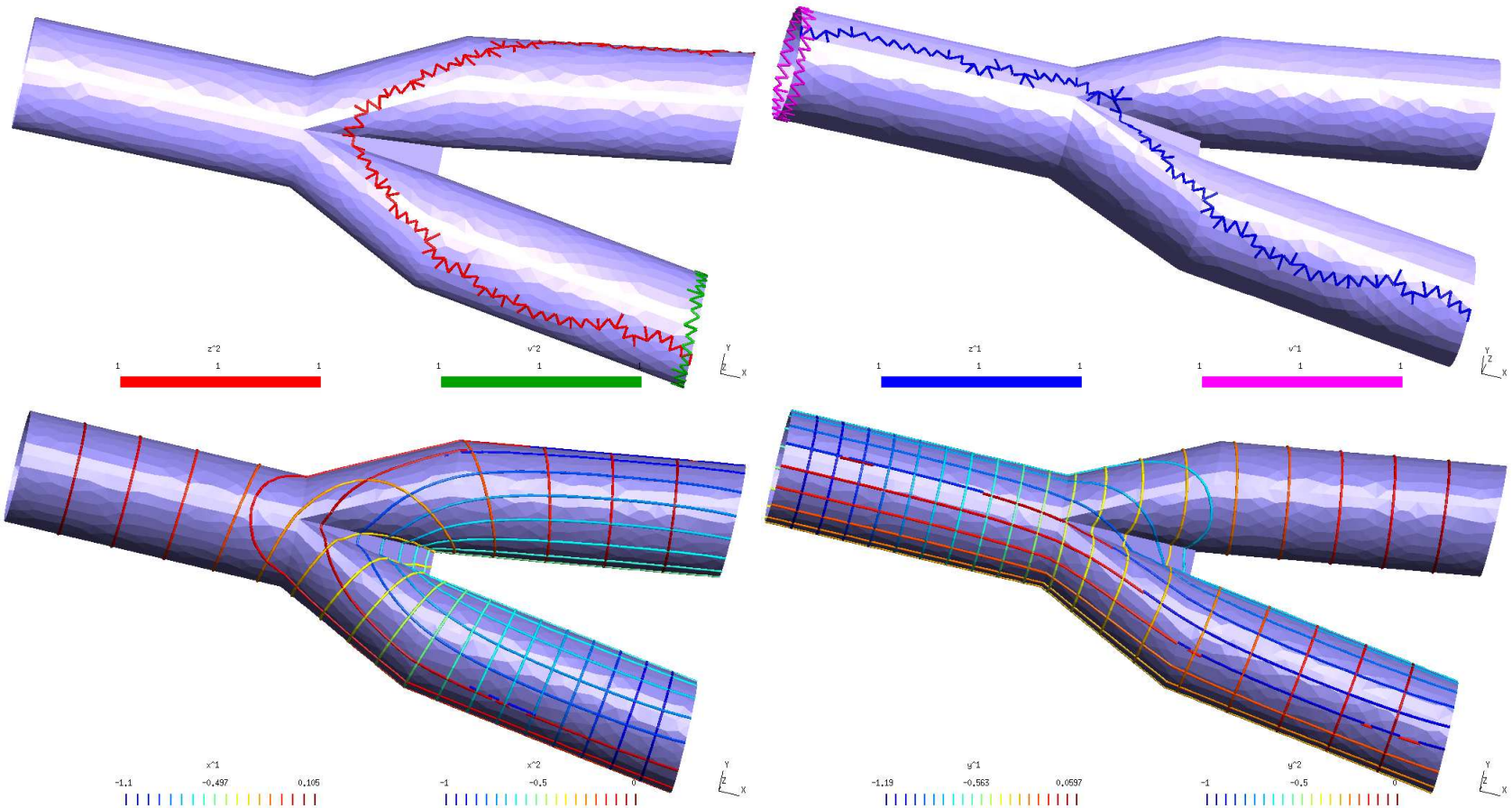
Applications in computational physics

- Many physical phenomena are modeled by vector fields that are both irrotational and solenoidal
  - Their line or surface integrals depend only on the homology class of the integration domain: they belong to a cohomology class
- Some related boundary value problems are difficult to pose, for example:
  - Magnetodynamics using magnetic scalar potential
  - Electrostatics using vector potential
- The results of the cohomology solver can be used to produce the source vector fields:



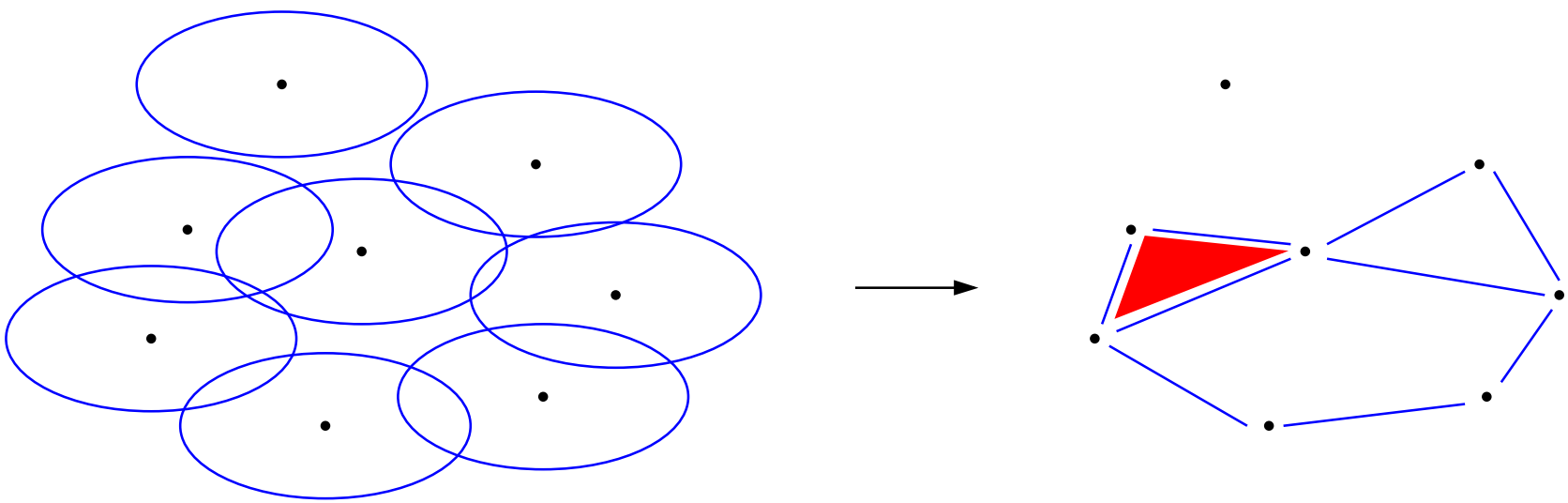
Applications in surface parametrization

- Many surfaces cannot be parametrized with a single 2-dimensional coordinate patch
- Cohomology solver can be used to produce the needed "zippers":



Applications in analysis of cell complexes

- Cell complexes arise also from other sources than finite element meshes
- Number of connected components, number of loops, and number of voids of a cell complex can be a measure of quality in some applications
- Example: Cell complex from a point cloud:



Conclusions

- Homology is detects holes of various dimensions in a domain
- Cohomology assigns quantities to these holes
- Quite efficient homology and cohomology solver available in Gmsh
- Implementation motivated by applications in the computational electromagnetics
- Waiting to be exploited by other applications

Thank you!