A brief tour of remeshing: basic theory, applications and perspectives

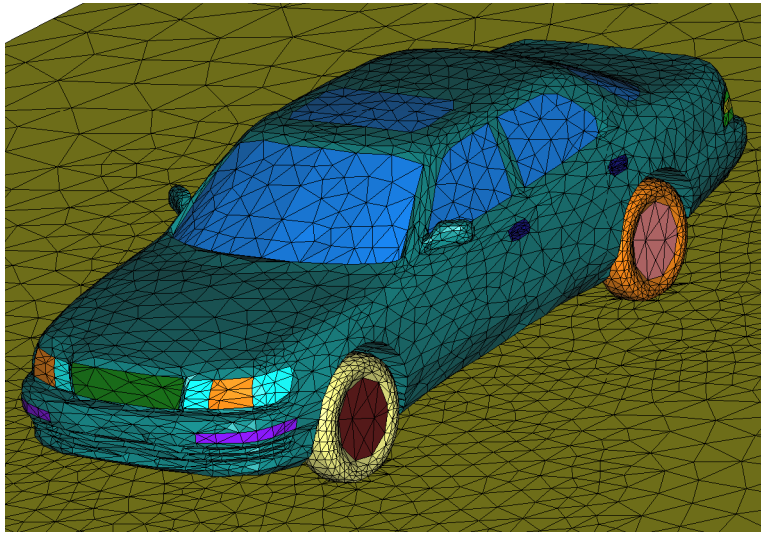
Charles Dapogny[1], Cécile Dobrzynski [2], Algiane Froehly[3] and Pascal Frey [4]

[1] CNRS & Laboratoire Jean Kuntzmann, Université Grenoble Alpes, Grenoble, France
[2] EPI Cardamom, Inria Bordeaux Sud-Ouest, France
[3] InriaSoft, Inria, Pau, France
[4] Laboratoire Jacques-Louis Lions, Sorbonne Universités, Paris, France

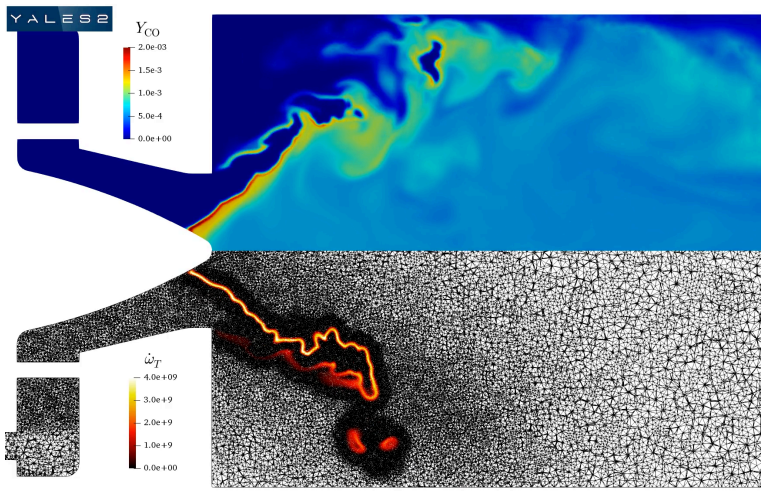
27th January, 2022

Foreword (I)

- Meshes are one of the prominent formats for representing a domain Ω in scientific computing.
- A substantial endeavour has long been devoted to mesh processing issues.
- Among these, remeshing aims at modifying an existing mesh so as to
  - Increase the quality of its elements;
  - Improve the approximation of the continuous domain.
- Remeshing has various applications:
  - Optimal adaptation of the local mesh size to the simulation of a physical phenomenon;
  - Robust description of the motion of a domain;
  - Mesh generation!



Mesh of a car.



Numerical simulation of a burner (Coria).

A word of advertisement

Mmg PLATFORM

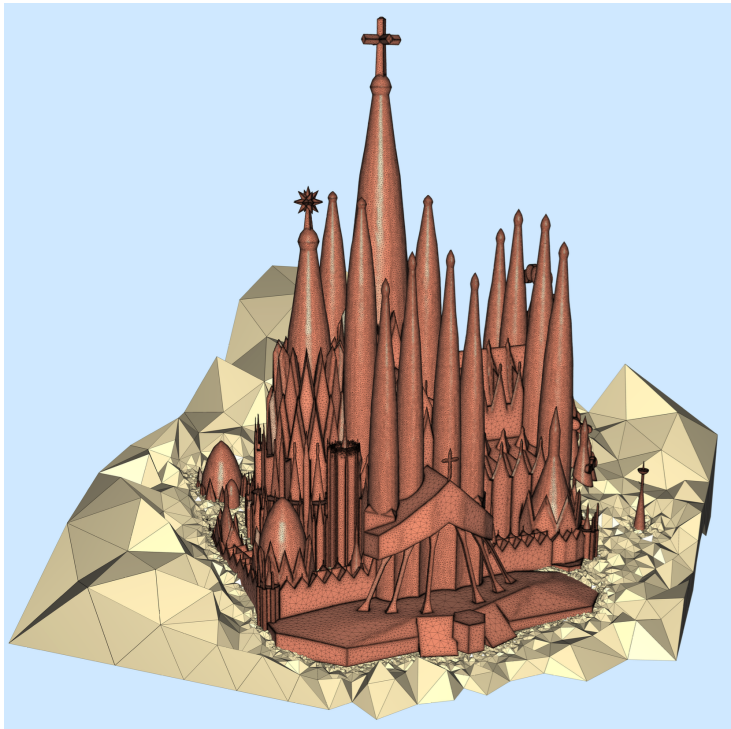
Robust, Open-source & Multidisciplinary Software for Remeshing

upgrade your meshes

Most of the features discussed in this presentation are integrated in the free, open-source environment Mmg.

https://www.mmgtools.org

https://github.com/MmgTools/mmg



1. Basic theory of remeshing
   - A few definitions and key concepts
   - Why remeshing?
   - Remeshing in practice
2. Applications of remeshing
   - Adaptation to a user-defined size prescription
   - Isosurface discretization and volume mesh generation
   - Lagrangian motion of a domain
   - Body-fitted interface tracking
3. A glimpse on a recent challenge: parallel remeshing

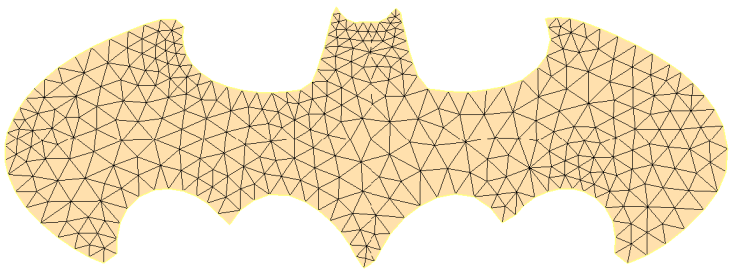
A few definitions about meshes (I)

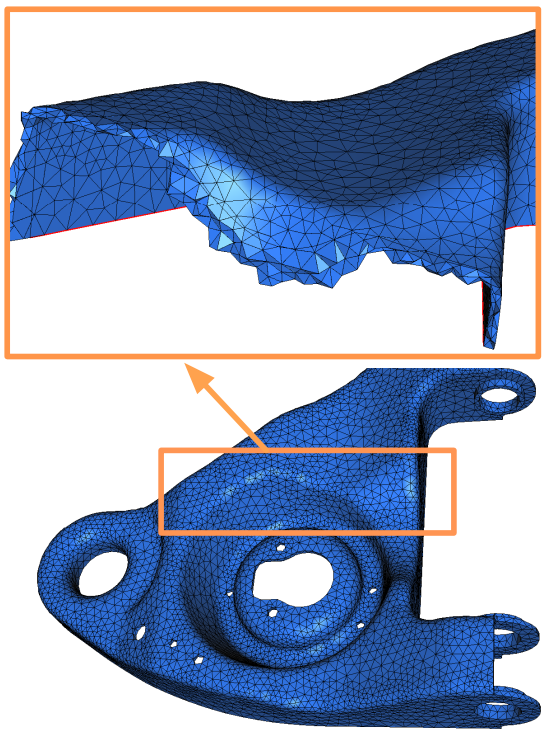
Let $\Omega \subset \mathbb{R}^d$ ($d = 2$ or 3) be a polyhedral domain.

Definition 1.

A simplicial mesh $\mathcal{T}$ of Ω is a collection $\{T_i\}_{i=1,\dots,N_T}$ of open simplices (triangles in 2d, tetrahedra in 3d) such that

$$\overline{\Omega} = \bigcup_{i=1}^{N_T} \overline{T_i}.$$

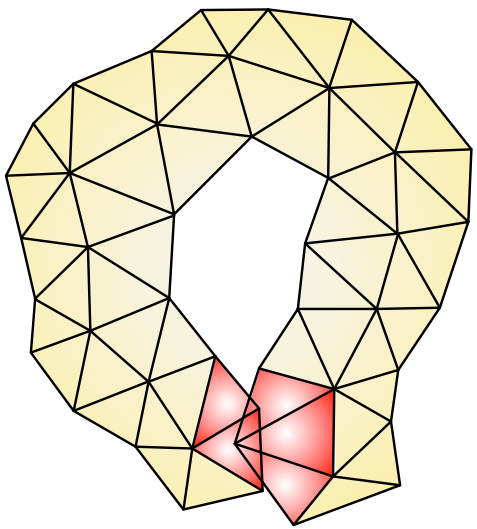




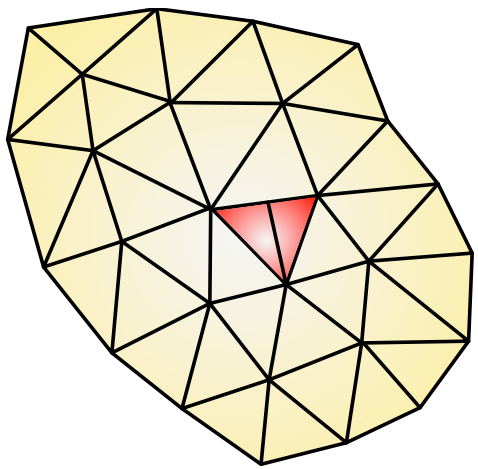
A few definitions about meshes (II)

Most often, the mesh $\mathcal{T}$ is often required to be

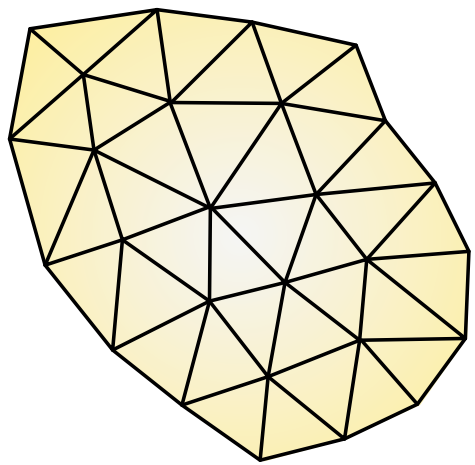
- Valid: the open simplices T_i are mutually disjoint: $T_i \cap T_j = \emptyset$ when $i \neq j$;
- Conforming: each intersection $\overline{T_i} \cap \overline{T_j}$, $i \neq j$ is reduced to either a vertex, an edge, or a face of the mesh.



Overlapping elements in an invalid mesh



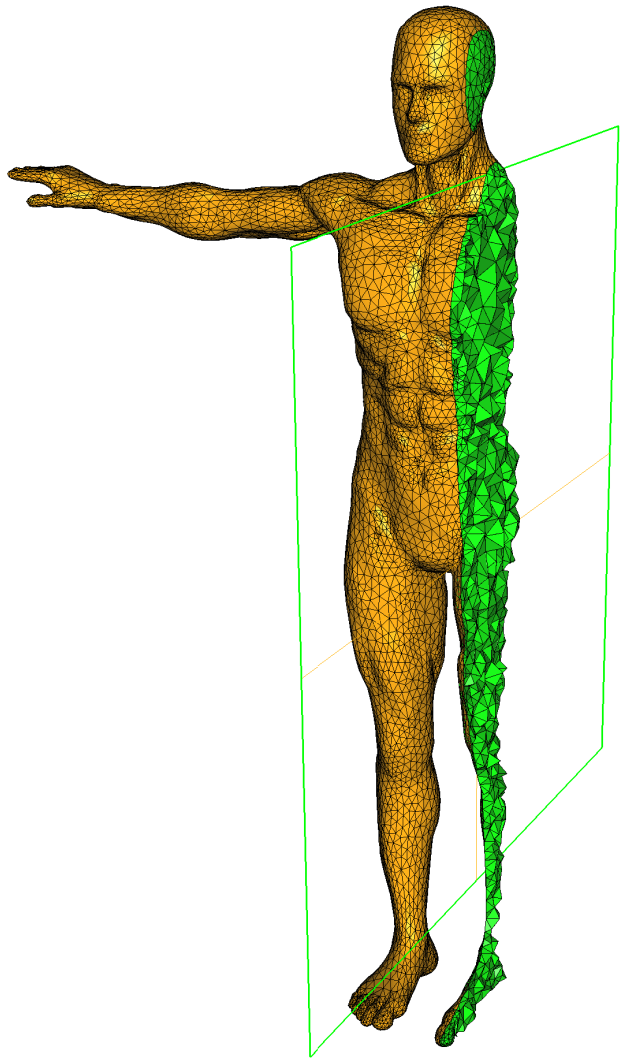
A non conforming mesh



A valid, conforming mesh

A few definitions about meshes (III)

- The mesh $\mathcal{T}$ naturally comprises a surface mesh $\mathcal{S}_\mathcal{T}$ associated to the boundary $\partial\Omega$:
  - In 2d, $\mathcal{S}_\mathcal{T}$ is a collection of segments;
  - In 3d, $\mathcal{S}_\mathcal{T}$ is a surface triangulation.
- In practice, the meshed domain Ω is *not* polyhedral and $\mathcal{S}_\mathcal{T}$ is meant to be an approximation of $\partial\Omega$.



Tetrahedral mesh $\mathcal{T}$ (in green) and associated surface triangulation $\mathcal{S}_\mathcal{T}$ (in yellow).

Quality of elements

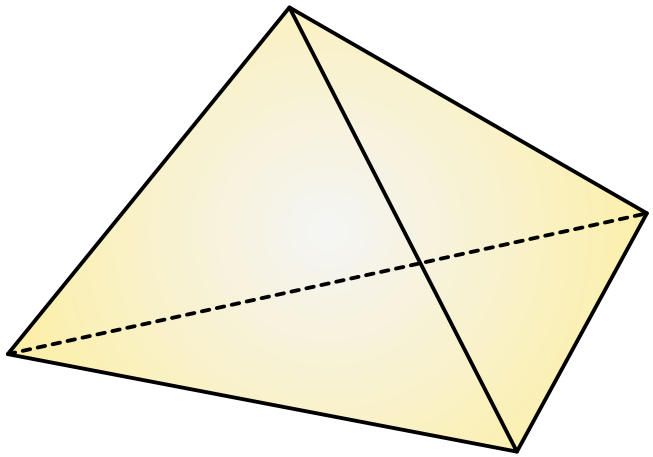
- The accuracy of most numerical methods using $\mathcal{T}$ as computational support (e.g. finite element computations) crucially depends on the quality of the simplices $T \in \mathcal{T}$.

- The latter is measured by a quality factor $\mathcal{Q}(T)$:
  - $\mathcal{Q}(T) \approx 1$ when T is close to regular;
  - $\mathcal{Q}(T) \approx 0$ when T is nearly flat.

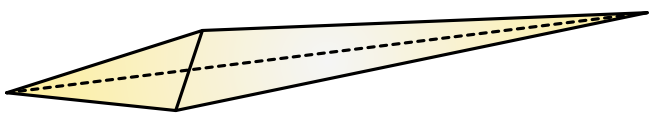
- In practice, $\mathcal{Q}(T)$ should "smoothly" discriminate "good", "bad" and "not so good" simplices T.

- A popular quality factor is for instance:

$$\mathcal{Q}(T) = \alpha \frac{\mathrm{Vol}(T)}{\left(\sum_{i=1}^{d(d+1)/2} |e_i|^2 \right)^{\frac{d}{2}}}.$$



A regular tetrahedron ($\mathcal{Q}(T) \approx 1$)



A nearly degenerate tetrahedron ($\mathcal{Q}(T) \approx 0$)

Quality of the geometric approximation (I)

The surface mesh $\mathcal{S}_\mathcal{T}$ should be a close approximation of $\partial\Omega$:

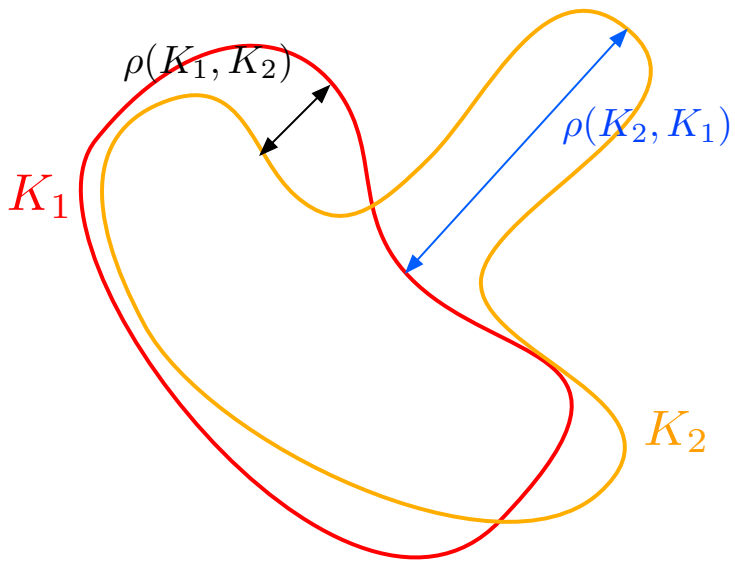
$$d^H(\mathcal{S}_\mathcal{T}, \partial\Omega) \leq \varepsilon,$$

where ε is a user-defined threshold and $d^H(\cdot,\cdot)$ stands for e.g. the Hausdorff distance:

Definition 2.

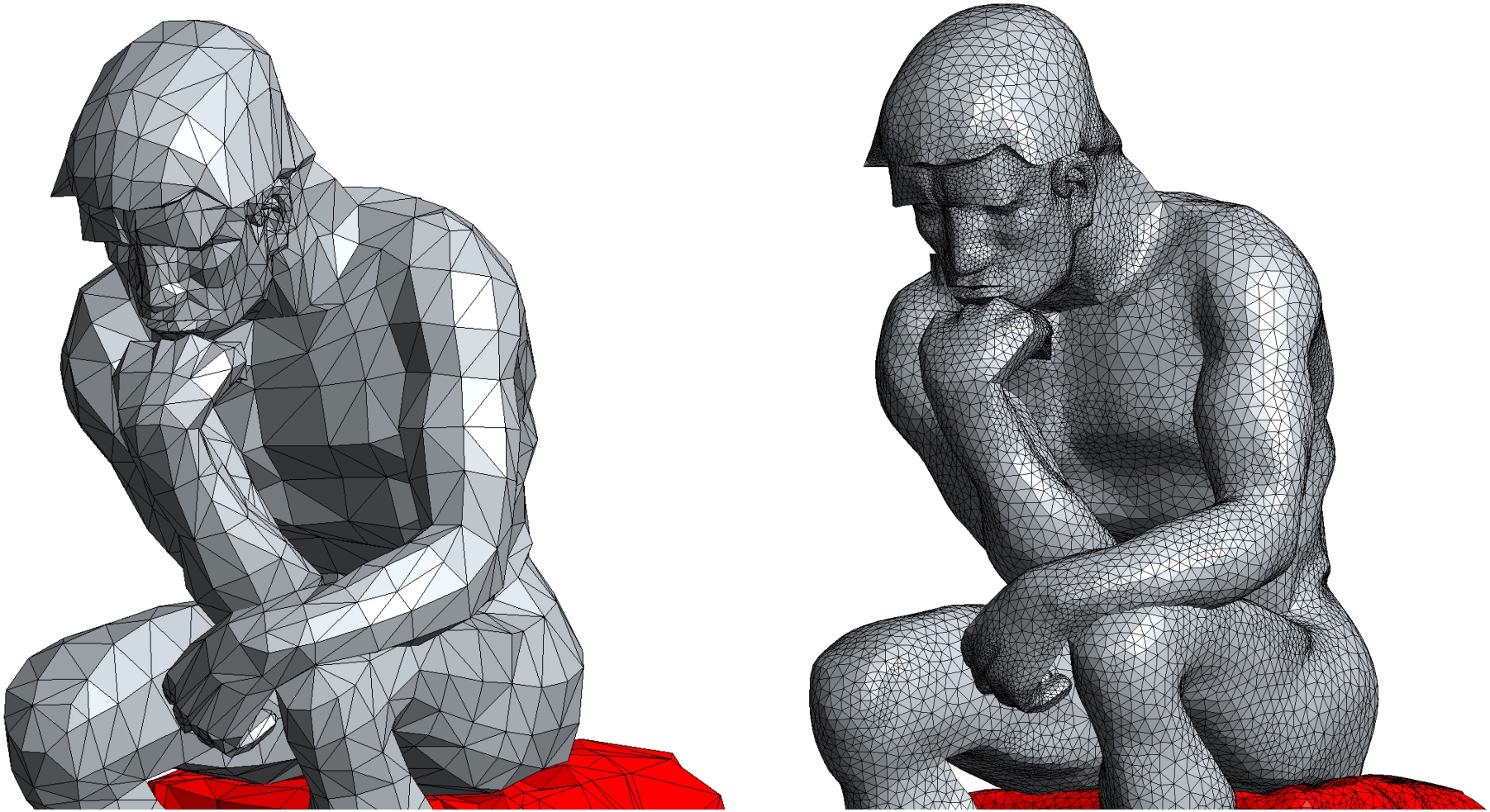
The Hausdorff distance $d^H(K_1, K_2)$ between two compact subsets $K_1, K_2 \subset \mathbb{R}^d$ is:

$$d^H(K_1, K_2) = \max(\rho(K_1, K_2), \rho(K_2, K_1)), \text{ where } \rho(K_1, K_2) := \max_{x \in K_1} d(x, K_2).$$



The Hausdorff distance between K_1 and K_2 measures the "maximum gap" between both sets.

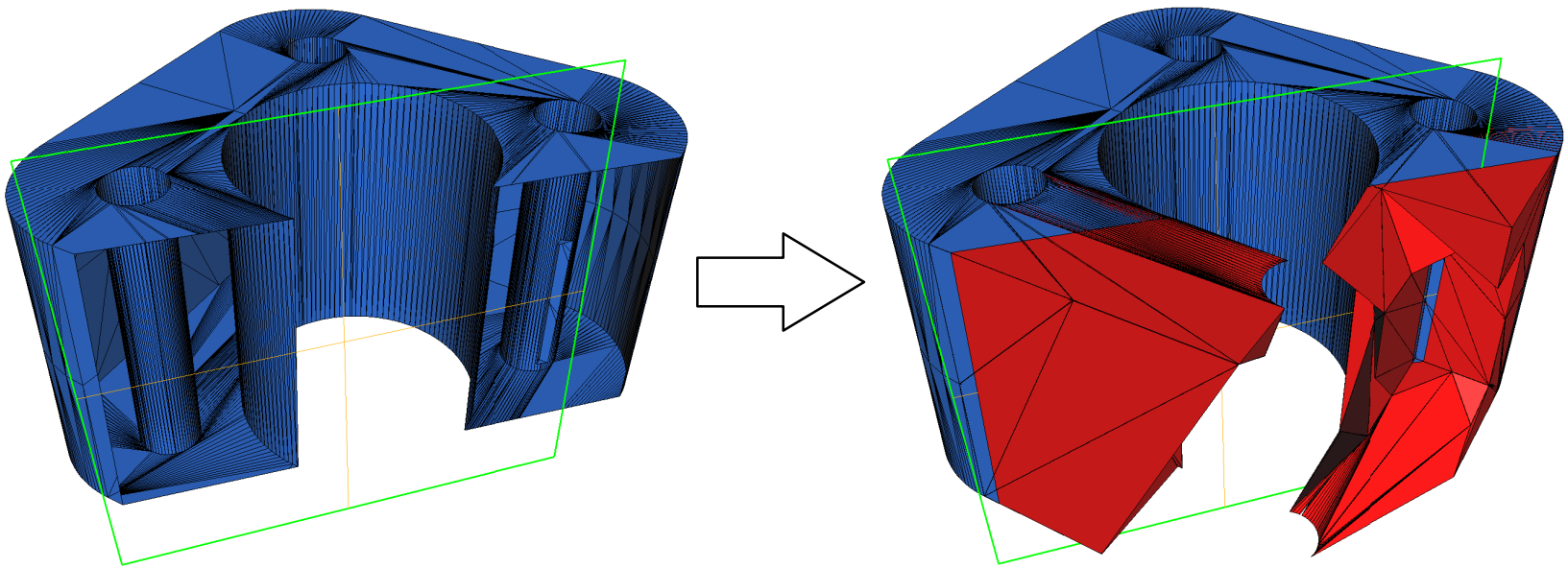
Quality of the geometric approximation (II)



(Left) Rough approximation of a domain $\Omega \subset \mathbb{R}^3$; (right) fine geometric approximation of Ω.

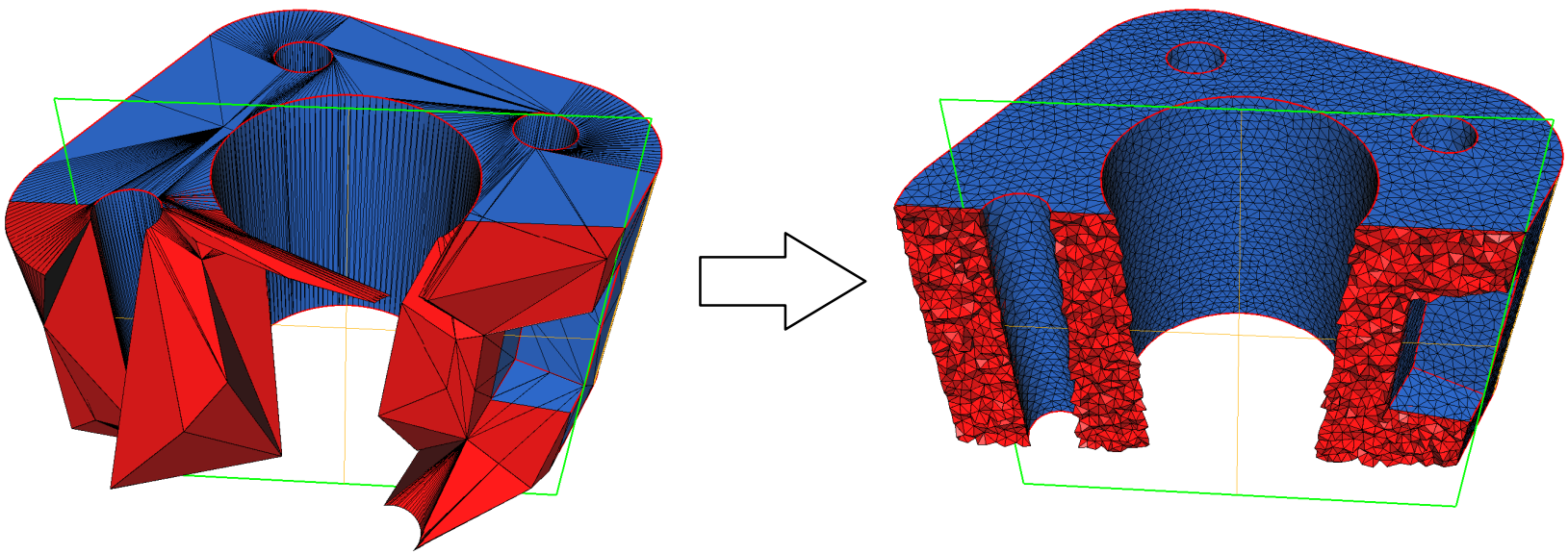
Meshing vs. remeshing (I)

- Meshing starts from the datum of a (line or surface) mesh $\mathcal{S}$ of the boundary $\partial\Omega$.
- It aims to fill the volume Ω with simplices, i.e. to construct a mesh $\mathcal{T}$ of Ω with surface part $\mathcal{S}_{\mathcal{T}} = \mathcal{S}$.



Meshing vs. remeshing (II)

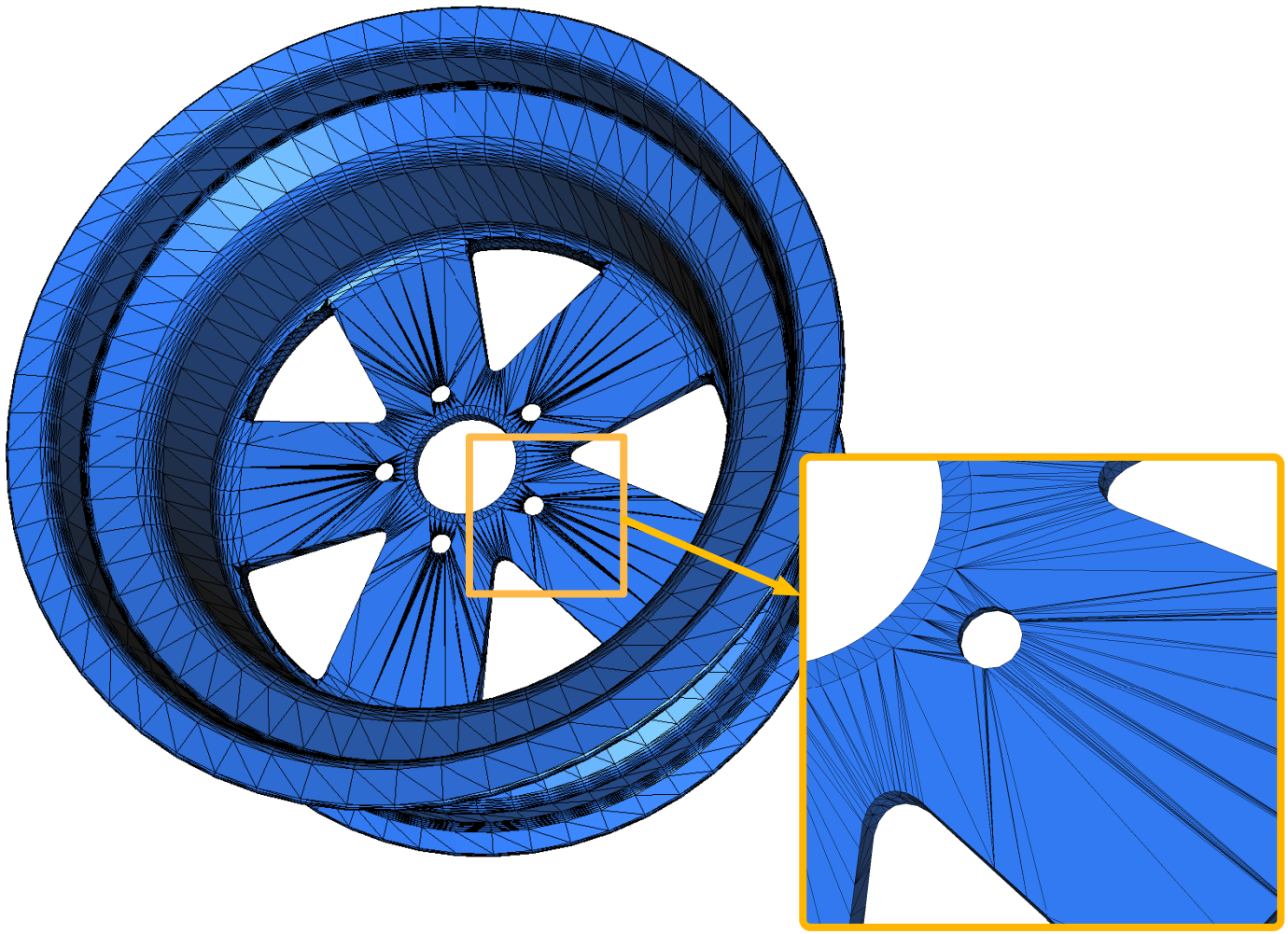
- Remeshing assumes the input of a valid, conforming mesh $\mathcal{T}$ of Ω.
- It aims to modify $\mathcal{T}$ into a "better" mesh $\widetilde{\mathcal{T}}$ of Ω.



1. Basic theory of remeshing
   - A few definitions and key concepts
   - Why remeshing?
   - Remeshing in practice
2. Applications of remeshing
   - Adaptation to a user-defined size prescription
   - Isosurface discretization and volume mesh generation
   - Lagrangian motion of a domain
   - Body-fitted interface tracking
3. A glimpse on a recent challenge: parallel remeshing

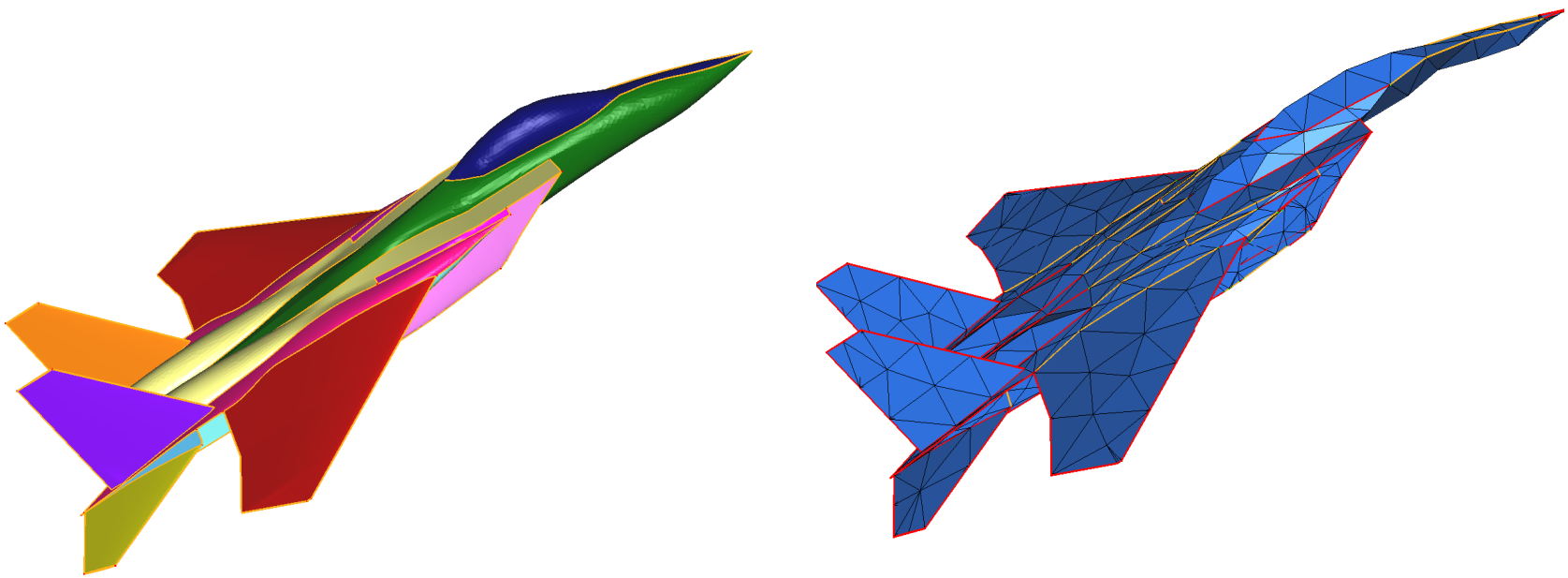
Why remeshing? (I)

1. The supplied mesh $\mathcal{T}$ may contain low-quality, nearly degenerate elements.



Why remeshing? (II)

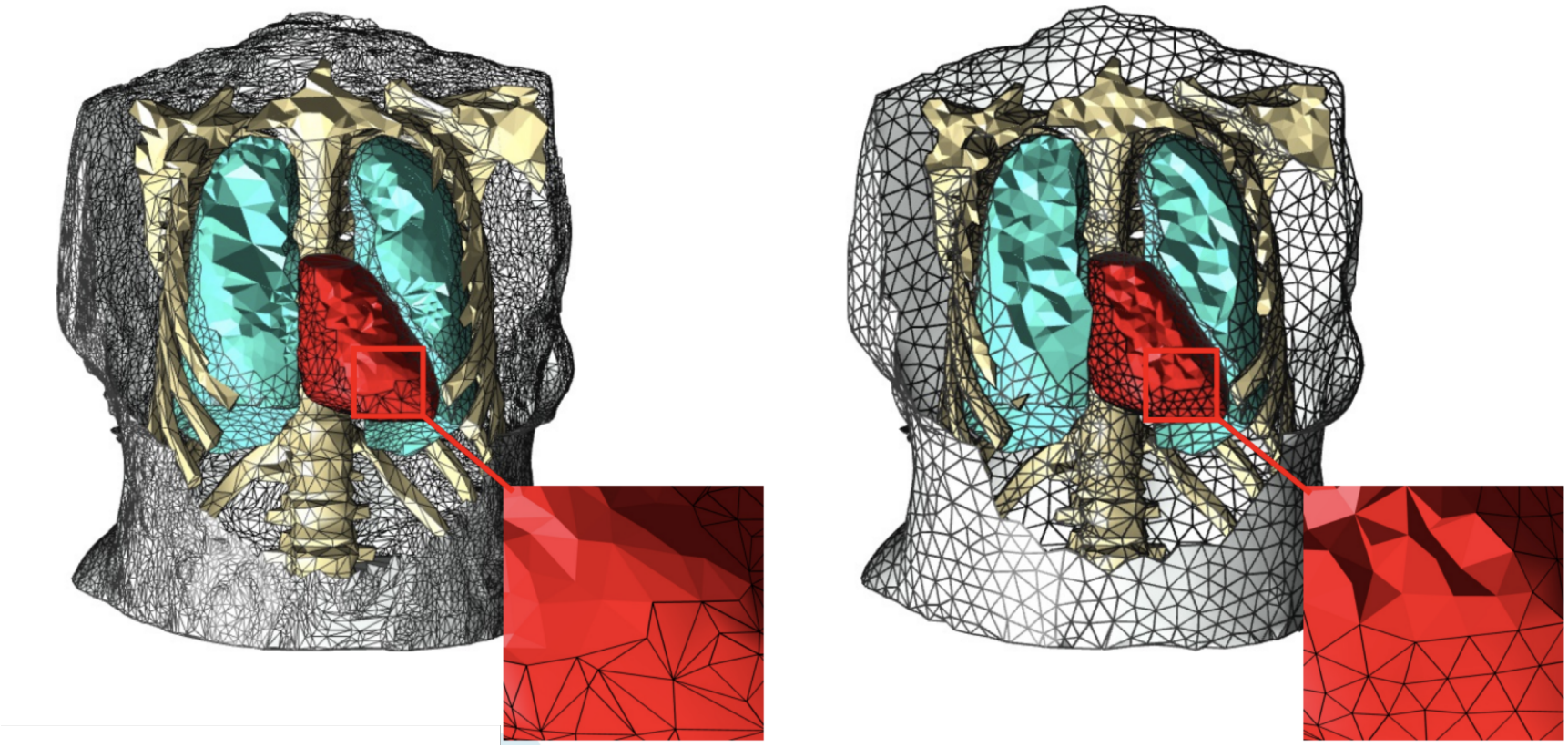
❷ The mesh $\mathcal{T}$ may be a rough geometric representation of the continuous domain Ω.



The mesh $\mathcal{T}$ on the right is a very coarse approximation of the continuous geometry Ω on the left.

Why remeshing? (III)

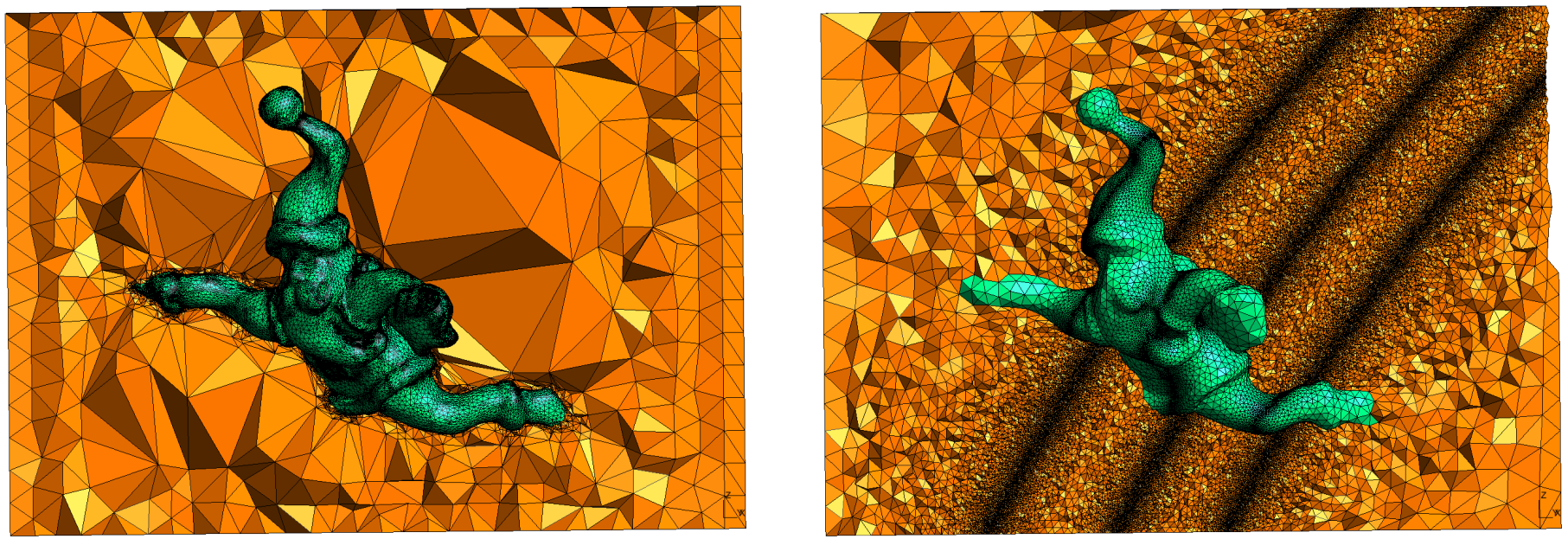
❸ The mesh $\mathcal{T}$ may be composed of an unnecessarily large number of elements, impeding the speed of mechanical computations.



Decimation of the oversampled mesh $\mathcal{T}$ (left) into a suitably sampled mesh $\widetilde{\mathcal{T}}$ (right) of the same domain Ω.

Why remeshing? (IV)

❹ The mesh $\mathcal{T}$ may have to be adapted to a user-defined size prescription resulting, e.g. from an a posteriori error analysis.



Adaptation of the mesh to a user-defined size prescription.

1. Basic theory of remeshing
   - A few definitions and key concepts
   - Why remeshing?
   - Remeshing in practice
2. Applications of remeshing
   - Adaptation to a user-defined size prescription
   - Isosurface discretization and volume mesh generation
   - Lagrangian motion of a domain
   - Body-fitted interface tracking
3. A glimpse on a recent challenge: parallel remeshing

The remeshing issue

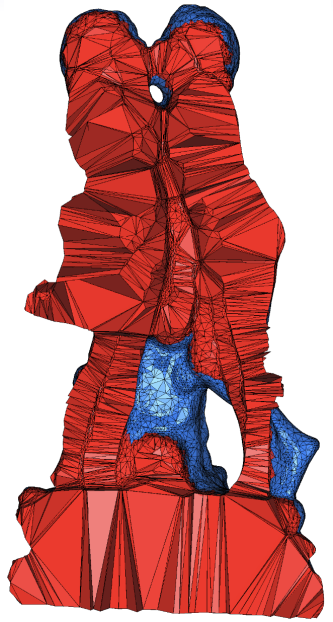
- Let $\Omega \subset \mathbb{R}^3$ be a bounded domain.

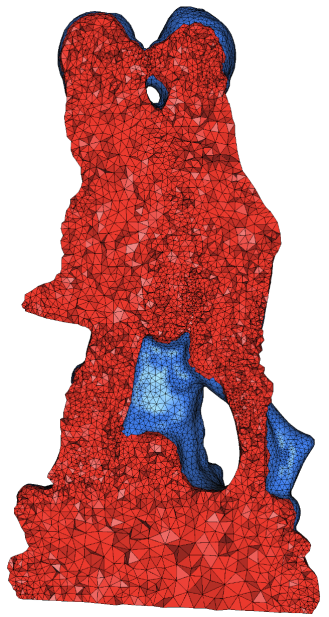
  (Similar considerations hold as regards remeshing of a 2d domain or a 3d surface)

- The continuous domain Ω is known solely via the datum of a discrete, simplicial mesh $\mathcal{T}$.

- The mesh $\mathcal{T}$ is unsuitable because
  - It is made of low-quality elements;
  - It is a poor geometric description of Ω.

- We wish to modify $\mathcal{T}$ into a new mesh $\widetilde{\mathcal{T}}$
  - With fine element quality;
  - Which is an accurate geometric approximation of Ω;
  - (Optionally) Whose elements comply with a user-defined size prescription.





The remeshing strategy (I)

- In practice, the domain Ω is unknown: the only information at hand is that of the input, discrete mesh $\mathcal{T}$.
- A key ingredient in the practice of remeshing is the reconstruction of a continuous, "ideal" domain Ω.
- This task may be conducted in two different manners:
  - A whole continuous representation Ω is calculated from the initial datum $\mathcal{T}$ at the beginning of the remeshing process and it is set once and for all.

  Or

  - Whenever needed, a piece of $\partial\Omega$, corresponding e.g. to a processed surface triangle $T \in \mathcal{S}_{\mathcal{T}}$, is calculated from the current state of the mesh $\mathcal{T}$.
- The continuous counterpart Ω of $\mathcal{T}$ serves for instance
  - To predict the local element size for an accurate geometric description of Ω

    (many small elements have to be placed in high curvature regions, while just a few large ones are needed in flat areas).
  - To detect when $\mathcal{T}$ deviates "too much" from the continuous geometry Ω.

The remeshing strategy (II)

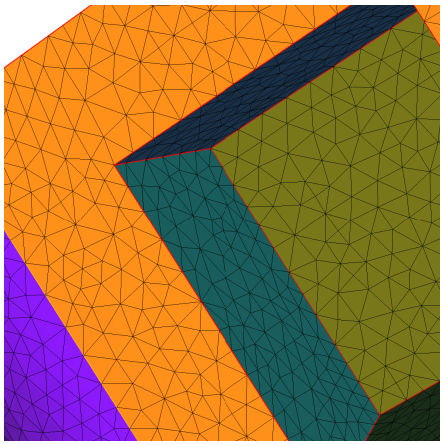
- A size map $h : \mathcal{T} \to \mathbb{R}$ is defined:

$$\text{For all vertex } x \in \mathcal{T}, \ h(x) = \text{ desired size for edges near } x.$$

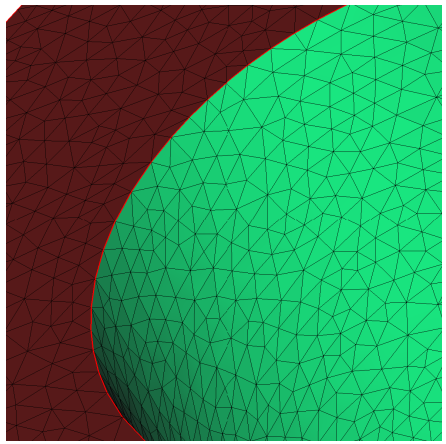
- Remeshing rests upon the combination of 4 local operators:
  - Edge split: a "long" edge in the mesh is split into two.
  - Edge collapse: the endpoints of a "short" edge are merged.
  - Edge swap: an edge is removed, and the leftover cavity is suitably reconnected.
  - Vertex relocation: one vertex is moved, while all connections are unchanged.
- Each of these operators is available in two versions, depending on whether it is applied to a surface or a volume configuration.
- Each operation has to be carefully controlled:
  - It may create invalid configurations (e.g. overlapping elements).
  - It may degrade too much the geometric approximation of Ω.

From the discrete to a continuous geometry (I)

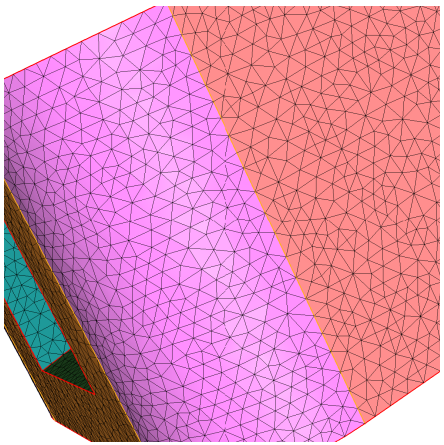
Step 1: Identify the geometric features of $\partial\Omega$ from the triangulation $\mathcal{T}$.



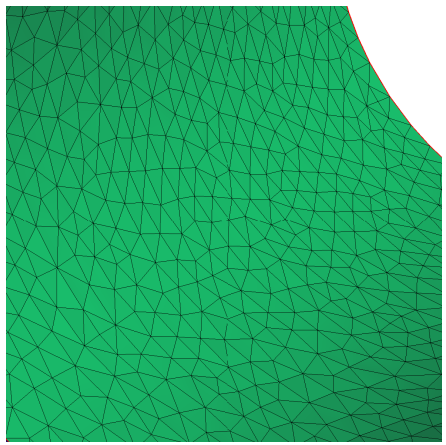
Corners should not be affected by remeshing



Two smooth portions of the surface meet with a sharp angle at a ridge



Reference edges are drawn on a smooth region of the surface, and delimit regions bearing different labels



The remaining vertices lie on "smooth" regions

From the discrete to a continuous geometry (II)

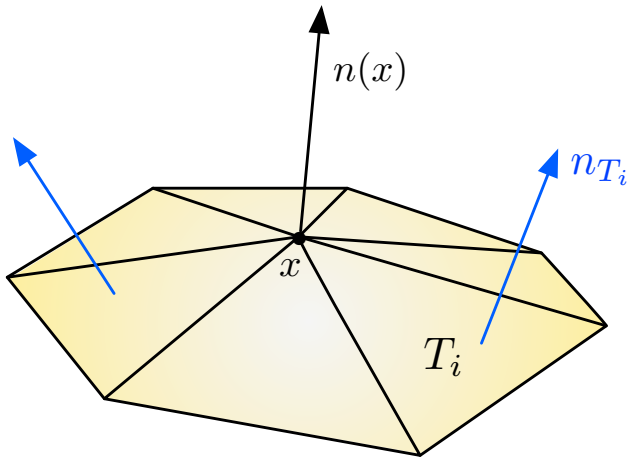
Step 2: Reconstruct first-order information about the ideal surface $\partial\Omega$:

- The normal vector at a smooth vertex $x \in \mathcal{T}$ is approximated from the normal vectors n_T to the neighboring triangles:

$$n(x) = \frac{1}{\sum_{T \in B(x)} \alpha_T} \sum_{T \in B(x)} \alpha_T n_T,$$

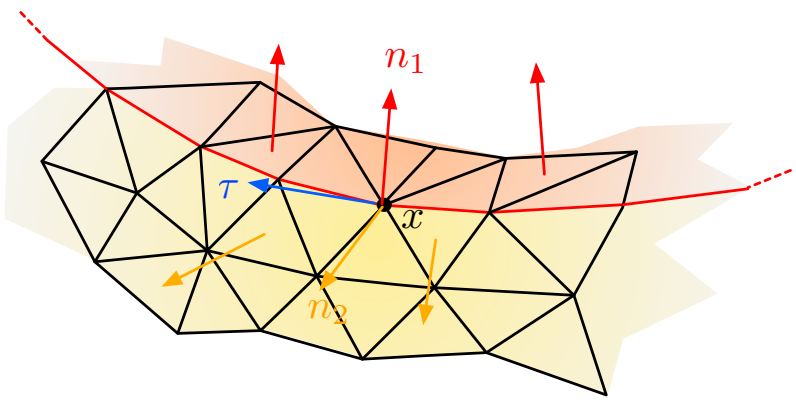
with the weights α_T given by, e.g.

$$\alpha_T = |T|.$$



- One vertex x on a ridge naturally bears
  - Two normal vectors n_1, n_2,
  - One tangent vector τ,

which are reconstructed analogously.

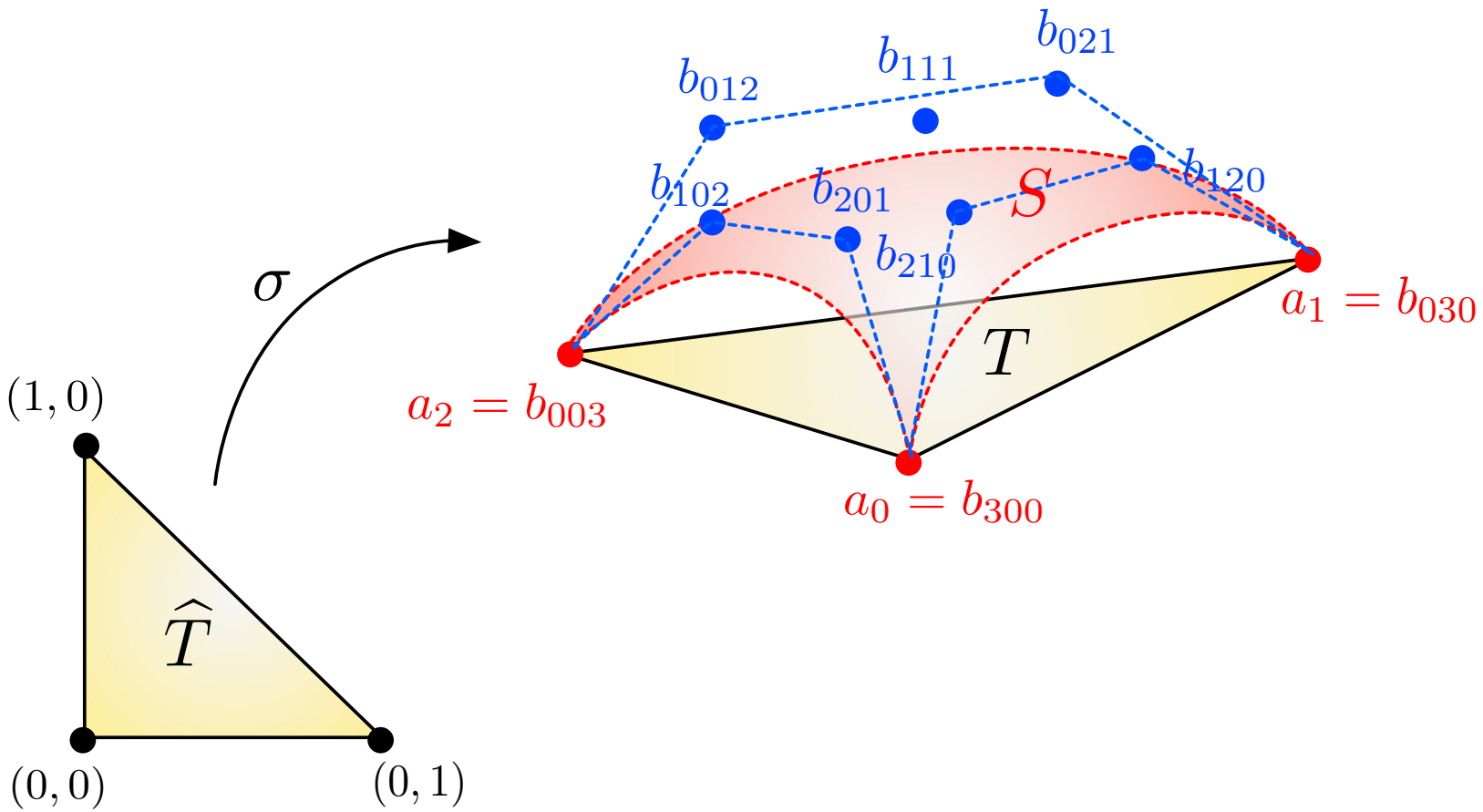


From the discrete to a continuous geometry (III)

Step 3: Generate a parametrization (e.g. a cubic Bézier patch) of a piece of "ideal" surface $S \subset \partial\Omega$ associated to T:

$$\forall (u,v) \in \widehat{T} \subset \mathbb{R}^2, \quad \sigma(u,v) = \sum_{\substack{i,j,k=0,\dots,3,\\ i+j+k=3}} \frac{3!}{i!j!k!} b_{ijk} (1-u-v)^i u^j v^k,$$

where the control points $b_{ijk} \in \mathbb{R}^3$ are inferred from the first-order entities of $\partial\Omega$.

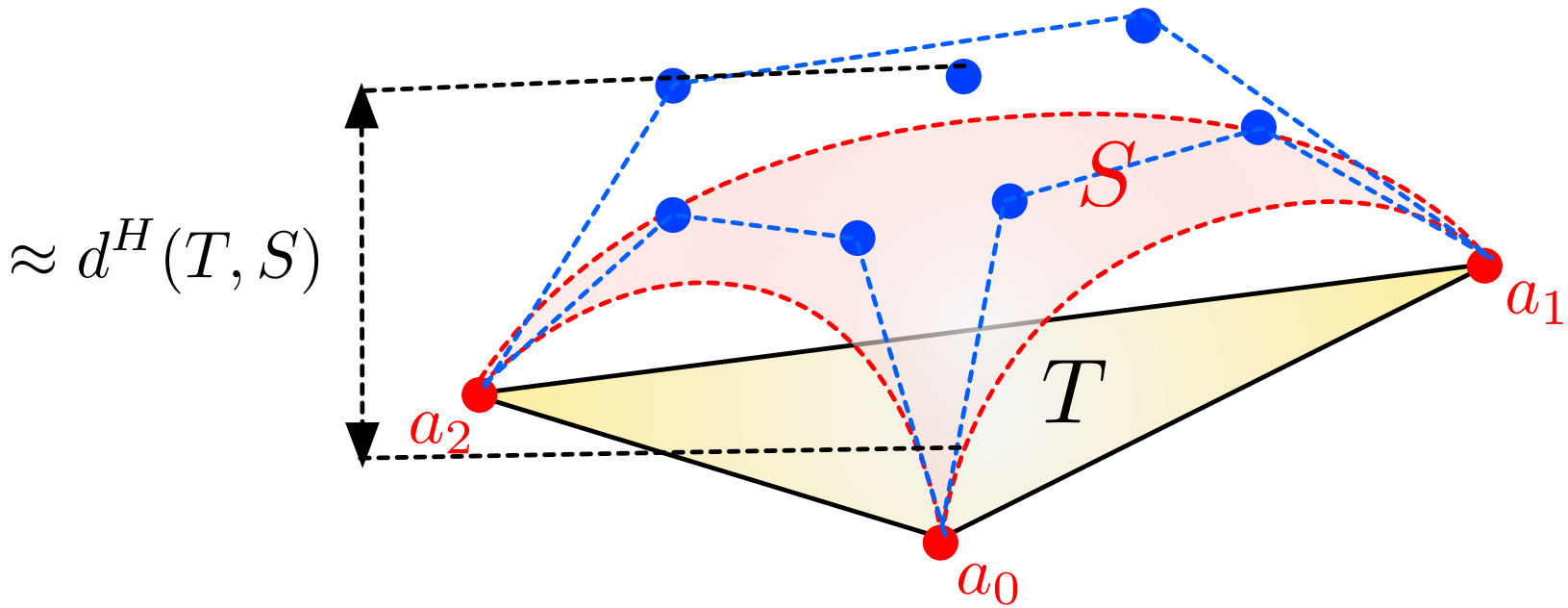


From the discrete to a continuous geometry (IV)

This local parametrization allows to:

1. Give a close estimate of the Hausdorff distance $d^H(T, S)$;
2. Predict which size is desirable for entities near a_0, a_1, a_2 so that

$$d^H(T, S) \leq \varepsilon.$$



The Hausdorff distance $d^H(T, S)$ can be estimated by looking at the control polygon of S.

Calculation of a size map (I)

A size map $h : \Omega \to \mathbb{R}$ is defined on (the mesh $\mathcal{T}$ of) Ω:

For each vertex $x \in \mathcal{T}$, $h(x)$ is the desired size for edges around x.

- When x is an internal vertex, $h(x)$ is calculated from the requirements of the user:
  - A minimum size for the edges in $\mathcal{T}$;
  - A maximum size for these edges;
  - (Optionally) A desired local size, stemming e.g. from a priori or a posteriori finite element analyses.
- When $x \in \mathcal{S}_\mathcal{T}$ is a surface vertex, the size $h(x)$ should additionally be adapted to the maximum imposed bound on the distance $d^H(\mathcal{S}_\mathcal{T}, \partial\Omega)$.
- The size map $h(x)$ should be "smooth": the ratio between the lengths of adjacent edges must be controlled to ensure a good element quality.

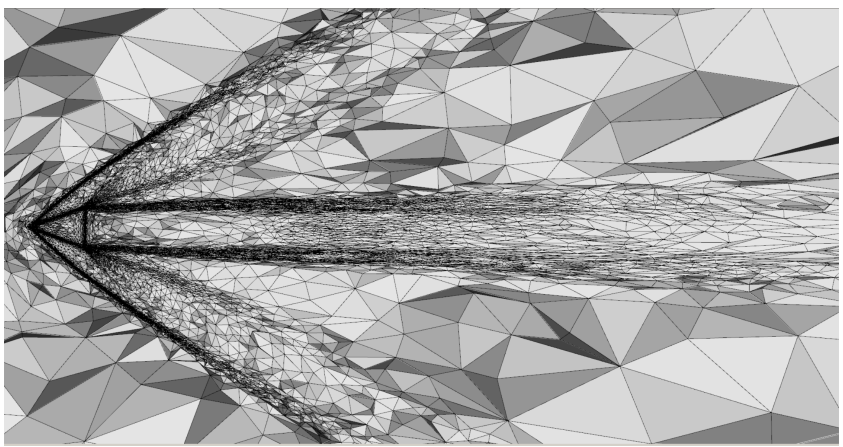
Calculation of a size map (II)

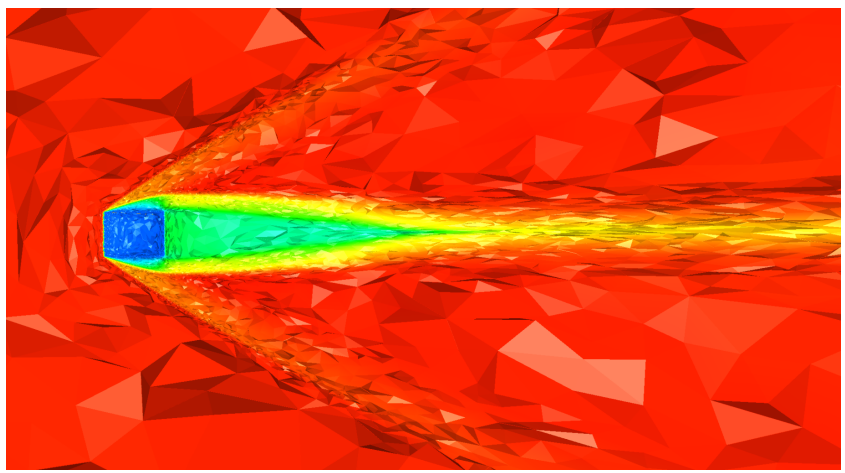
The size prescription may be anisotropic, encoded in a tensor field $\mathcal{M} : \mathcal{T} \to \mathbb{R}^{d \times d}$:

For all vertex $x \in \mathcal{T}$, for all direction $v \in \mathbb{S}^{d-1}$,

$\langle \mathcal{M}(x)v, v \rangle$ is (related to) the desired size for edges near x oriented along v;

see [VaHeMan].



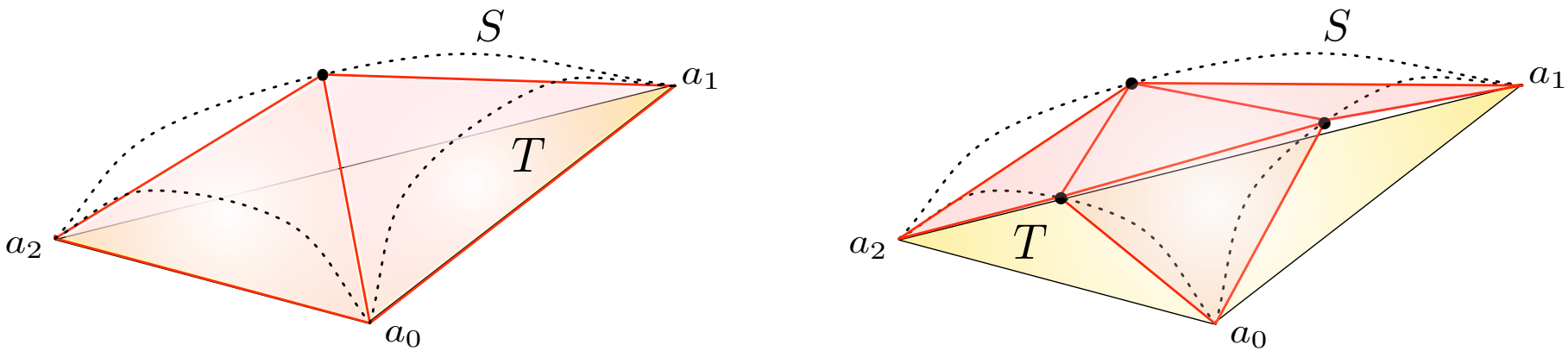


Anisotropic adaptation of the mesh for the simulation of a supersonic flow (from [AbAlBeDoNou]).

Local mesh operators: edge splitting

If an edge pq is "too long", insert its midpoint m, then split it into two.

- If pq belongs to a surface triangle $T \in \mathcal{S}_\mathcal{T}$, m lies on the piece of $\partial\Omega$ computed from T. Else, it is merely inserted as the midpoint of p and q.
- An edge may be deemed "too long" when compared to the prescribed size, or because it entails a bad geometric approximation of $\partial\Omega$.

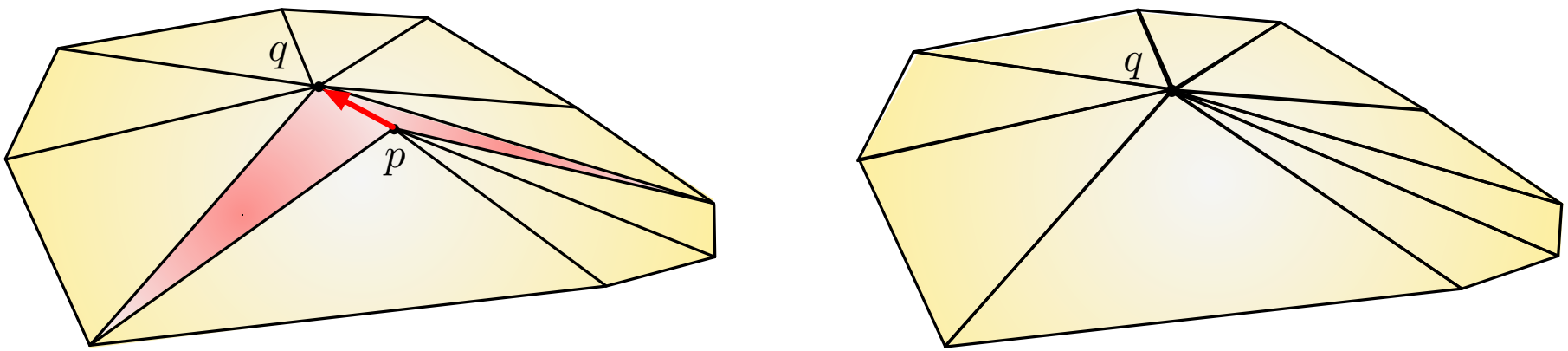


Splitting of one (left) or three (right) edges of triangle T, positioning the new points on the ideal surface $\mathcal{S}$ (dotted).

Local mesh operators: edge collapse

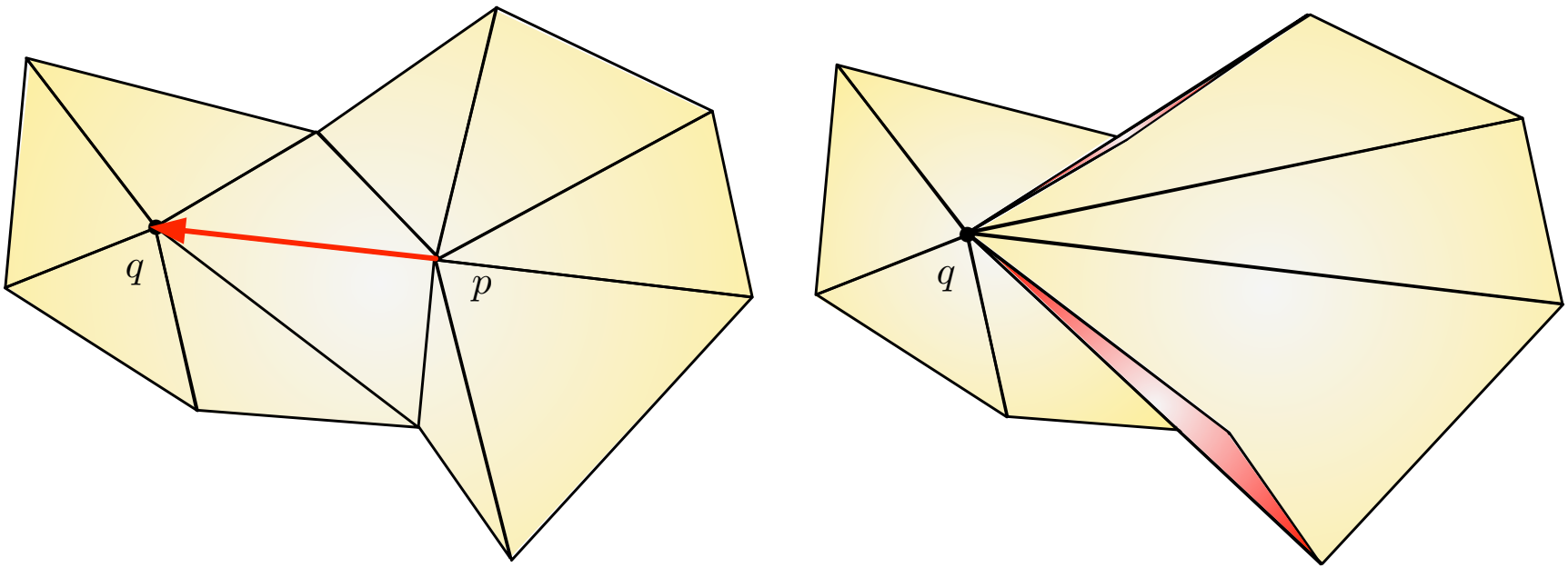
If an edge pq is "too short", merge its two endpoints.

- Careful checks are in order to ensure the validity of the resulting configuration:
  - This operation may invalidate some tetrahedra (i.e. create overlappings).
  - When it is applied to a surface configuration, it may deteriorate the geometric approximation of $\partial\Omega$;
- An edge may be "too short" when compared to the prescribed size, or because it is unnecessarily short for a fine geometric approximation of $\partial\Omega$.



Collapse of point p over q in a surface configuration.

Local mesh operators: edge collapse

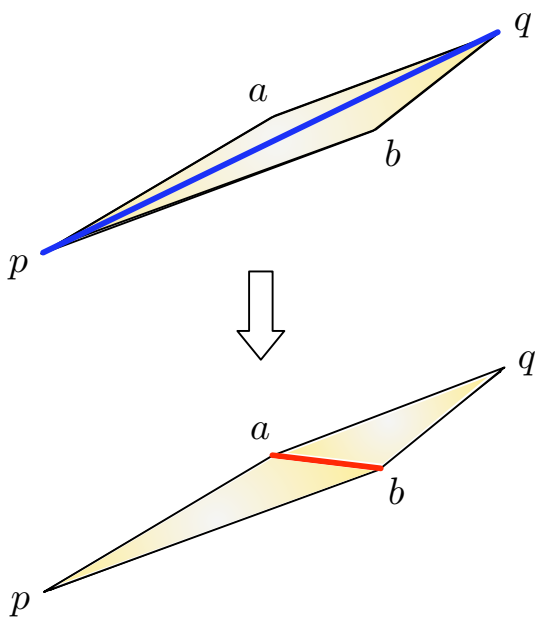


In 2d, collapsing p over q (left) invalidates the resulting mesh (right): both greyed triangles end up inverted.

Local mesh operators: edge swap (I)

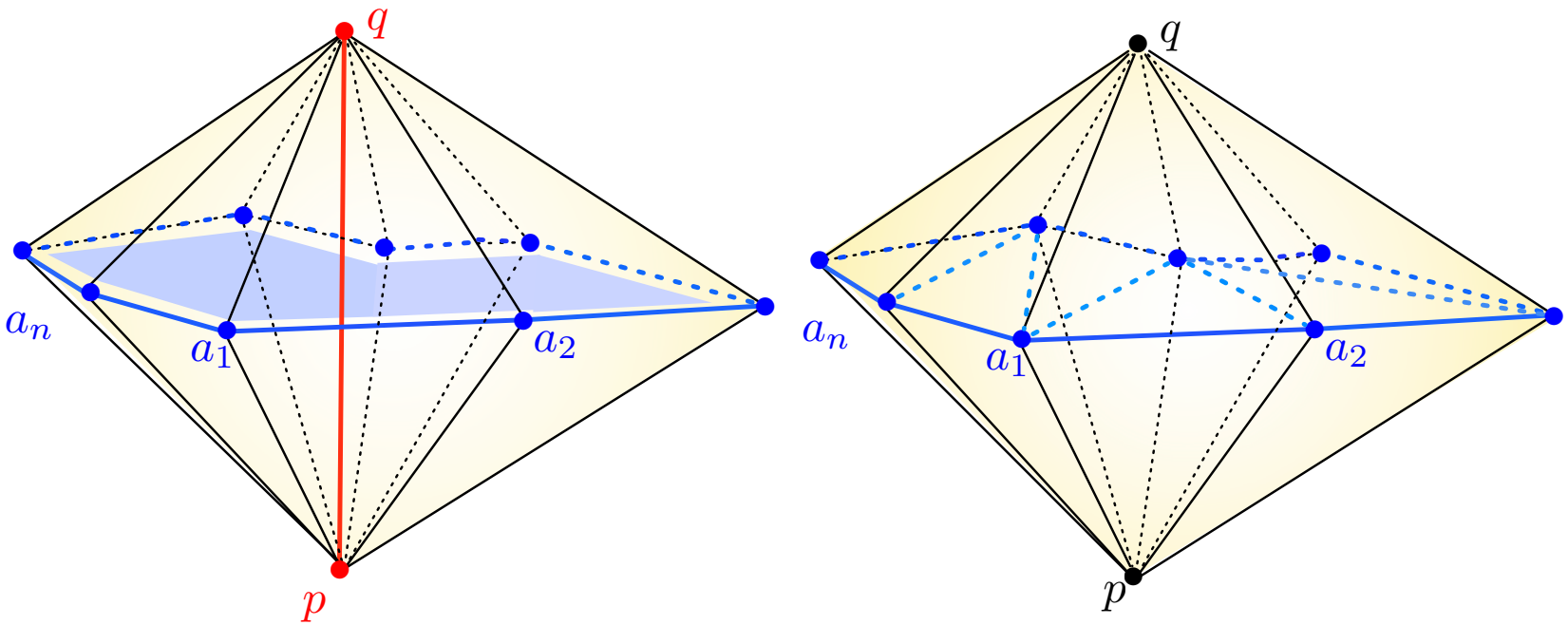
Suppress and edge pq from the mesh and reconnect the leftover cavity adequately.

This operator is key in improving the quality of the elements of the mesh.



In 2d the edge pq is removed from the mesh, and the edge ab corresponding to the alternate configuration is added.

Local mesh operators: edge swap (II)

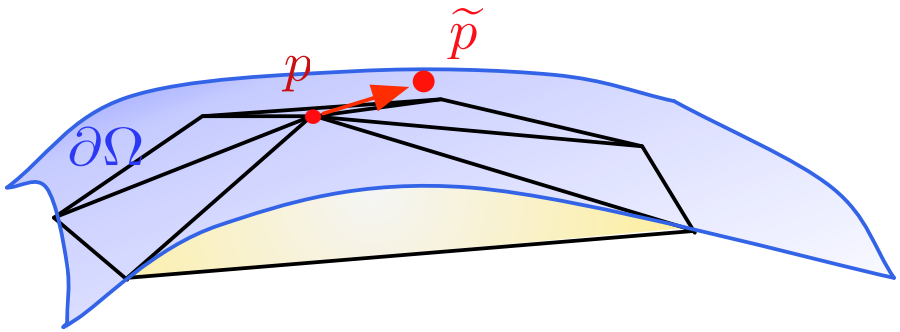


The 3d edge swap operator is much more involved than its 2d counterpart.

Local mesh operators: node relocation

Slightly move a point p in the mesh, while leaving all connectivities unchanged.

This operator is the main ingredient in the fine-quality tuning of the mesh



Relocation of node p to $\widetilde{p}$, along the surface.

Remeshing using mmg (I)

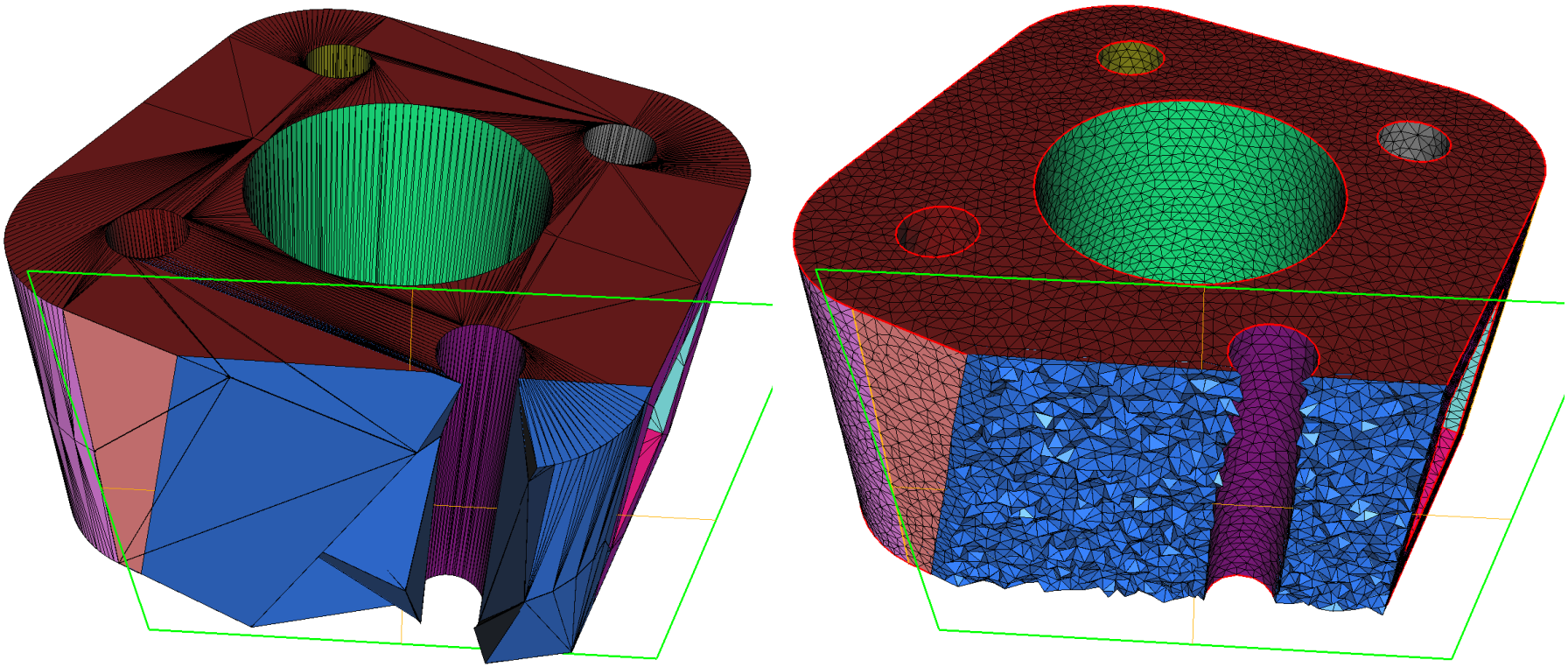
The mmg library relies on the input of a `.mesh` file for the mesh $\mathcal{T}$ (`input.mesh`), and a few parameters, passed on the command line:

- `-hmin`: Minimum value for the length of an edge in the mesh.
- `-hmax`: Maximum value for the length of an edge in the mesh.
- `-hausd`: Maximum gap between the mesh and the continuous geometry
- `-hgrad`: Maximum ratio between the lengths of two adjacent edges.
- `-nr`: Deactivates the detection of sharp features in the mesh.

Optionally, the user may provide a size map (or an anisotropic metric field) as a `.sol` file, with the same name as the mesh (`input.sol`).

Remeshing using mmg (II)

```
mmg3d input.mesh -hmin 0.001 -hmax 0.01 -hausd 0.005 -hgrad 1.3
```



(Left) initial, bad-quality mesh; (right) modified, good-quality mesh.

1. Basic theory of remeshing
   - A few definitions and key concepts
   - Why remeshing?
   - Remeshing in practice

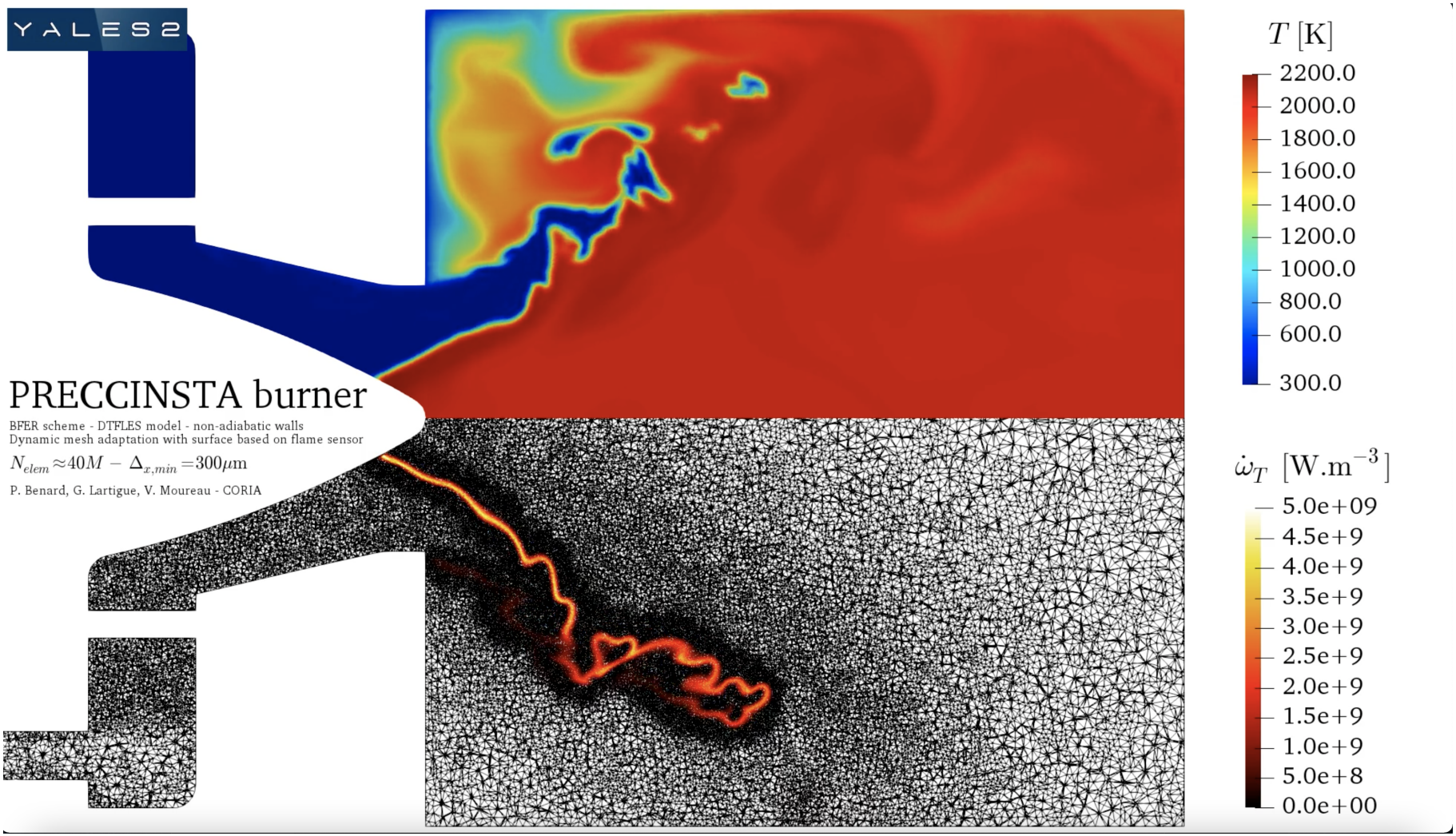
2. Applications of remeshing
   - Adaptation to a user-defined size prescription
   - Isosurface discretization and volume mesh generation
   - Lagrangian motion of a domain
   - Body-fitted interface tracking

3. A glimpse on a recent challenge: parallel remeshing

Adaptation to a size map

- One key application of remeshing consists in adapting the size of the mesh with respect to a user-defined size map.
- This allows to enforce "small" elements in regions of particular interest, and coarser elements elsewhere.
- Hence, the total size of the mesh is reasonable, while a particular focus is put on regions of interest.
- Depending on the purpose, this size prescription may be guided by:
  - The wish to enforce "small" elements in the vicinity of a moving front.
  - An a posteriori error estimate, attached to the resolution of a physical phenomenon by the finite element method;

Adaptation to a size map



Numerical simulation of an aeronautical burner using the `Yales2` *library [Yales2].*

1. Basic theory of remeshing
   - A few definitions and key concepts
   - Why remeshing?
   - Remeshing in practice

2. Applications of remeshing
   - Adaptation to a user-defined size prescription
   - Isosurface discretization and volume mesh generation
   - Lagrangian motion of a domain
   - Body-fitted interface tracking

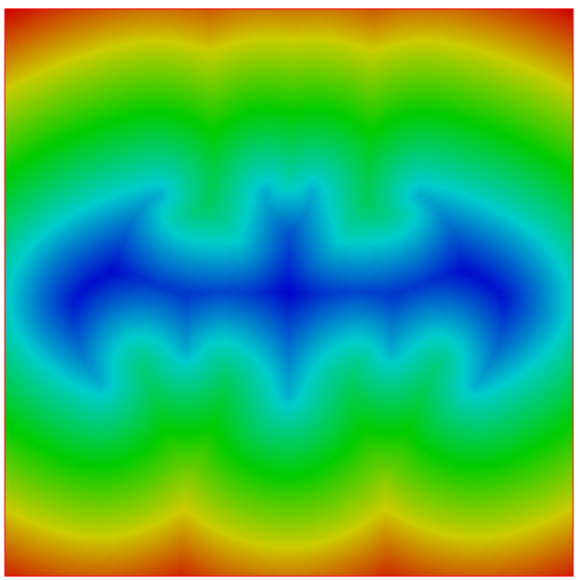
3. A glimpse on a recent challenge: parallel remeshing

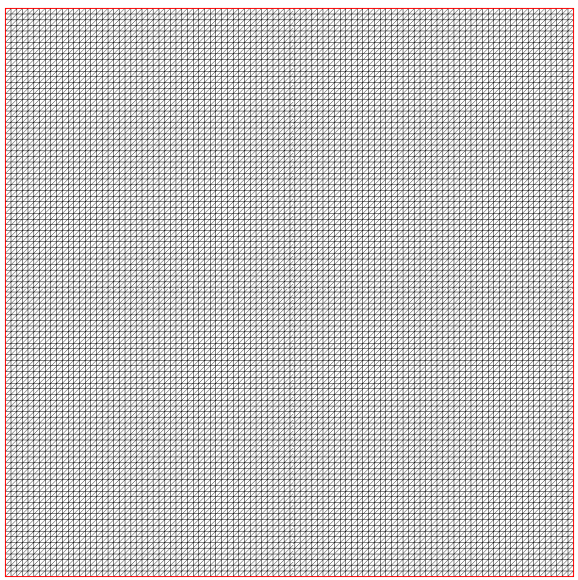
Isosurface discretization (I)

- In many applications of interest, a hold-all domain D is equipped with a computational, simplicial mesh $\mathcal{T}$.
- A scalar "level set" function $\phi : D \to \mathbb{R}$ is defined at the vertices of $\mathcal{T}$.
- Of particular interest is an isosurface Γ of ϕ (say, that associated to the value 0), or the corresponding negative subdomain Ω,

$$\Gamma := \{x \in D,\ \phi(x) = 0\}, \quad \Omega := \{x \in D,\ \phi(x) < 0\}.$$

- We wish to construct a mesh of Ω (or a surface triangulation of Γ).





(Left) Isolines of a level set function ϕ defined at the vertices of a mesh $\mathcal{T}$ of a computational box D (right).

Isosurface discretization (II)

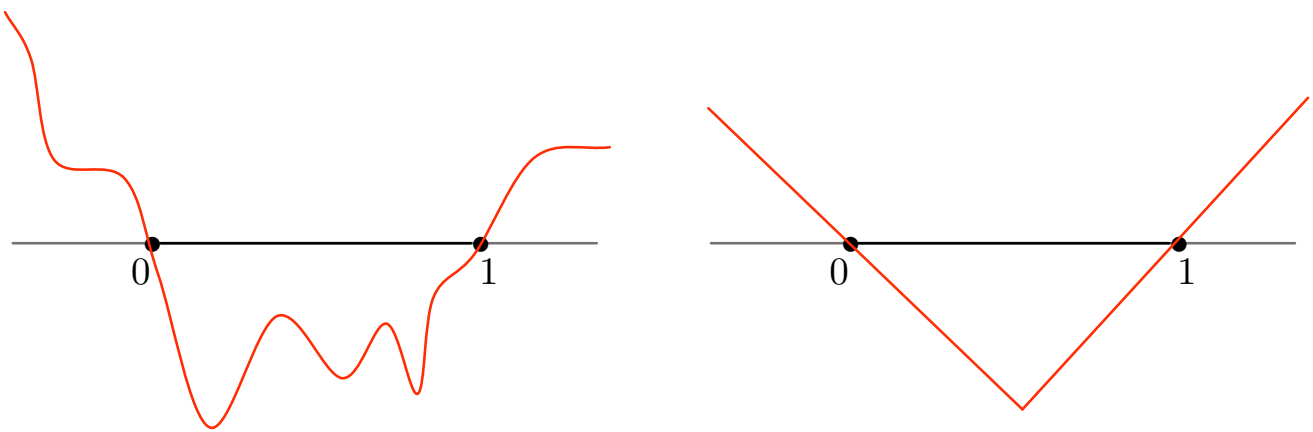
The level set function ϕ for $\Omega \subset D$ is often chosen as the signed distance function.

Definition 3.

The signed distance function $d_\Omega : \mathbb{R}^d \to \mathbb{R}$ *to a bounded domain* $\Omega \subset \mathbb{R}^d$ *is given by:*

$$d_\Omega(x) = \begin{cases} -d(x, \partial\Omega) & \text{if } x \in \Omega, \\ 0 & \text{if } x \in \partial\Omega, \\ d(x, \partial\Omega) & \text{otherwise,} \end{cases}$$

where $d(x, \partial\Omega) := \min_{p \in \partial\Omega} |x - p|$ *is the usual Euclidean distance from* x *to* $\partial\Omega$.

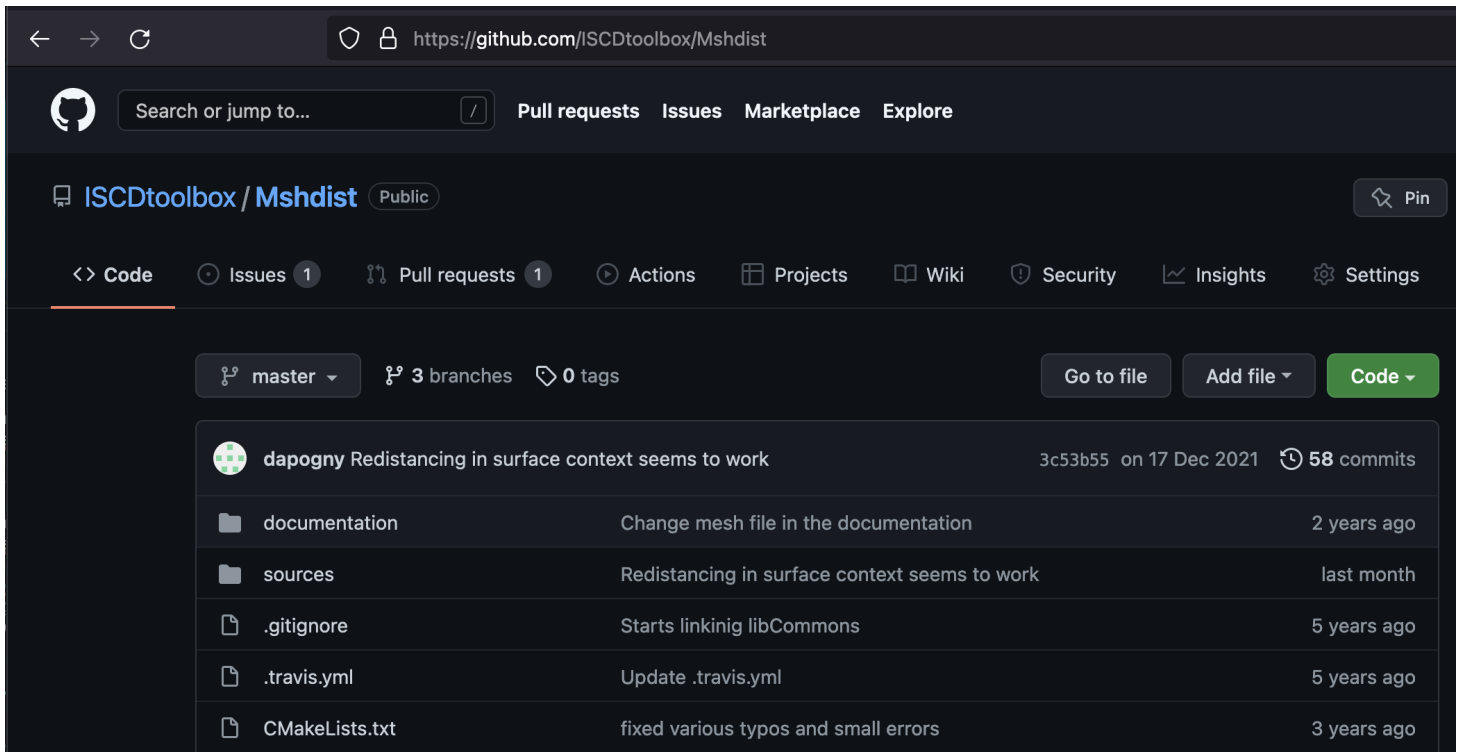


Two level set functions for the domain $\Omega = (0, 1) \subset \mathbb{R}$.

Isosurface discretization (III)

- Efficient algorithms exist to calculate d_Ω, such as the Fast Marching algorithm [SethianFMM], the Fast Sweeping algorithm [Zhao], etc.
- A free, open-source implementation: `mshdist` [DaFre].

`https://github.com/ISCDtoolbox/Mshdist`



Isosurface discretization (IV)

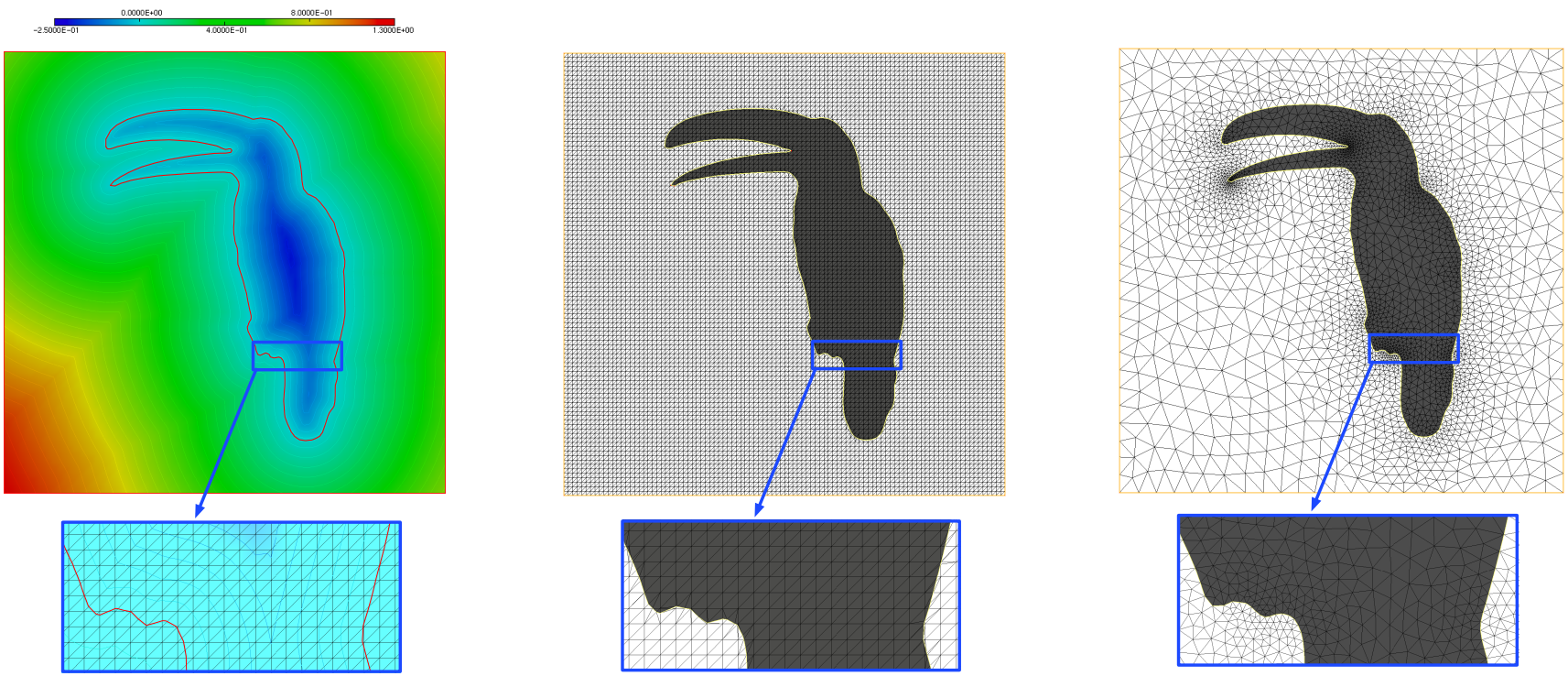
A two-step solution:

1. Discretize explicitly the 0 level set of ϕ into $\mathcal{T}$ by using patterns.

   $\Rightarrow$ a new valid, conforming mesh $\mathcal{T}_{\text{temp}}$ is obtained, which is of very low quality.

2. Improve the quality of $\mathcal{T}_{\text{temp}}$ by remeshing, to obtain $\widetilde{\mathcal{T}}$.

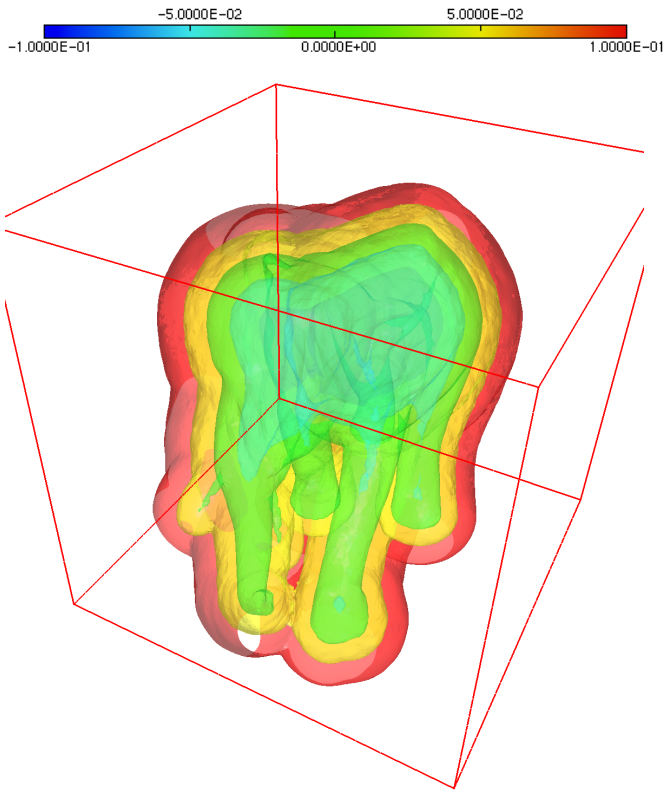
$\Rightarrow$ A high-quality mesh $\widetilde{\mathcal{T}}$ is obtained, where Ω and $D \setminus \overline{\Omega}$ are explicitly discretized.

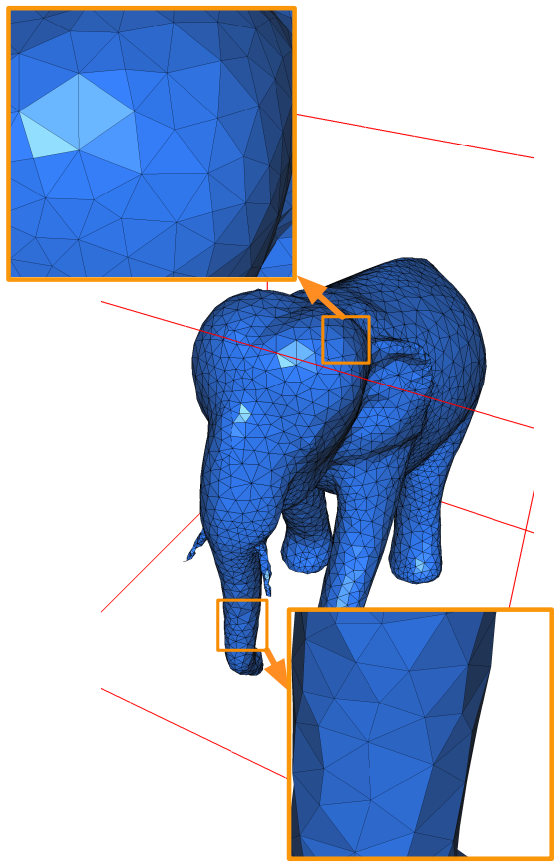


(Left) One level set function ϕ defined at the vertices of $\mathcal{T}$; (middle) low-quality mesh $\mathcal{T}_{temp}$ obtained from the discretization of the 0 level set of ϕ into $\mathcal{T}$; (right) high-quality mesh $\widetilde{\mathcal{T}}$ obtained after remeshing $\mathcal{T}_{temp}$.

Isosurface discretization using mmg

```
mmg3d input.mesh -ls -sol lsfunc.sol -hmin 0.001 -hmax 0.01
                 -hausd 0.001 -hgrad 1.3
```

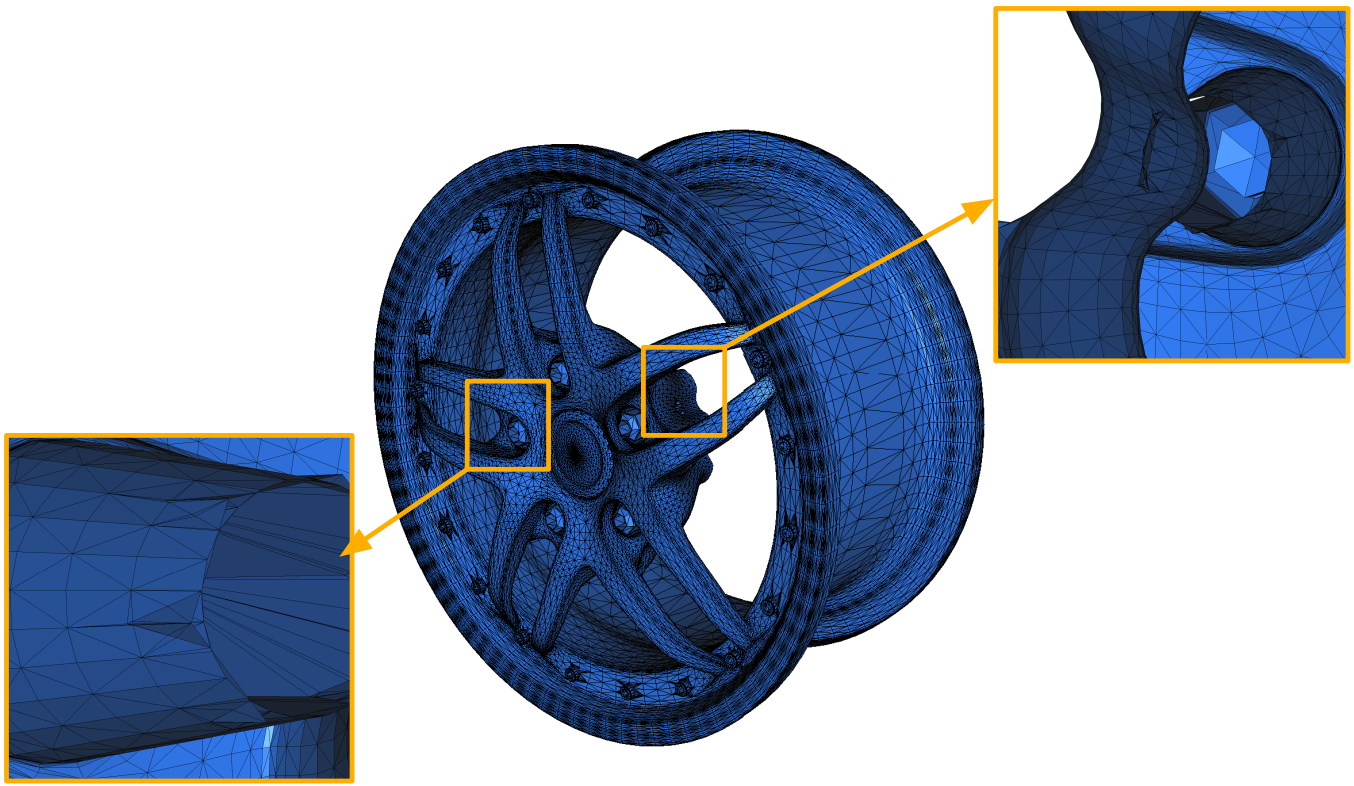




(Left) some isosurfaces of an implicit function defined in a cube, (right) result after discretization in the ambient mesh and local remeshing.

Volume mesh generation from an invalid surface triangulation (I)

- Let $\Omega \subset \mathbb{R}^d$ be a domain, supplied only via a surface mesh $\mathcal{S}$ of its boundary $\partial\Omega$.
- The mesh $\mathcal{S}$ may be invalid (i.e. show intersecting elements, small gaps, etc.).
- We wish to construct a mesh of Ω from this datum.

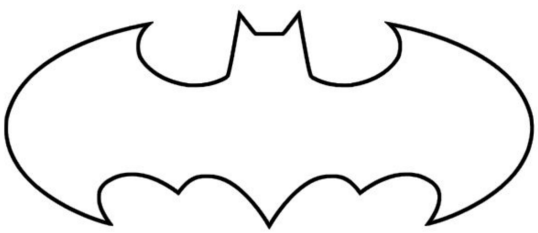


An invalid surface mesh of a domain Ω.

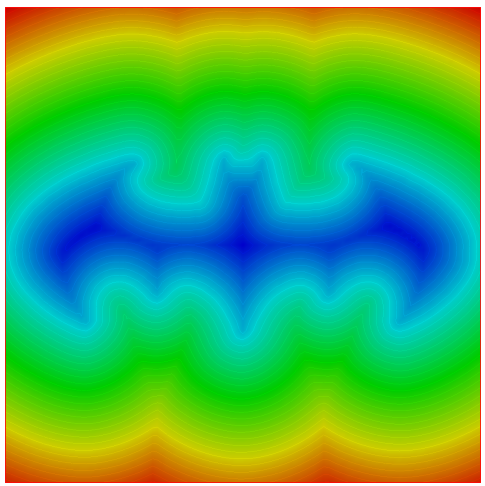
Volume mesh generation from an invalid surface triangulation (II)

One possible solution:

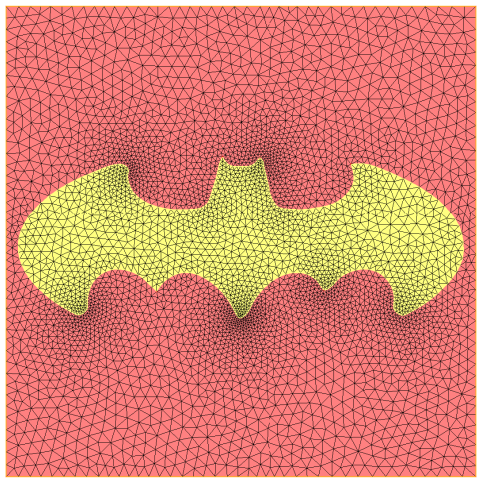
1. Calculate the signed distance function d_Ω to Ω, at the vertices of a mesh $\mathcal{T}$ of a larger, computational box D.
   - This calculation is possible even if the surface mesh $\mathcal{S}$ is invalid.
   - The mesh $\mathcal{T}$ may be adapted so that this calculation is accurate.
2. Apply the isosurface discretization operation to obtain a new mesh $\widetilde{\mathcal{T}}$ of D in which Ω is explicitly discretized.



Mesh $\mathcal{S}$ of the contour $\partial\Omega$.

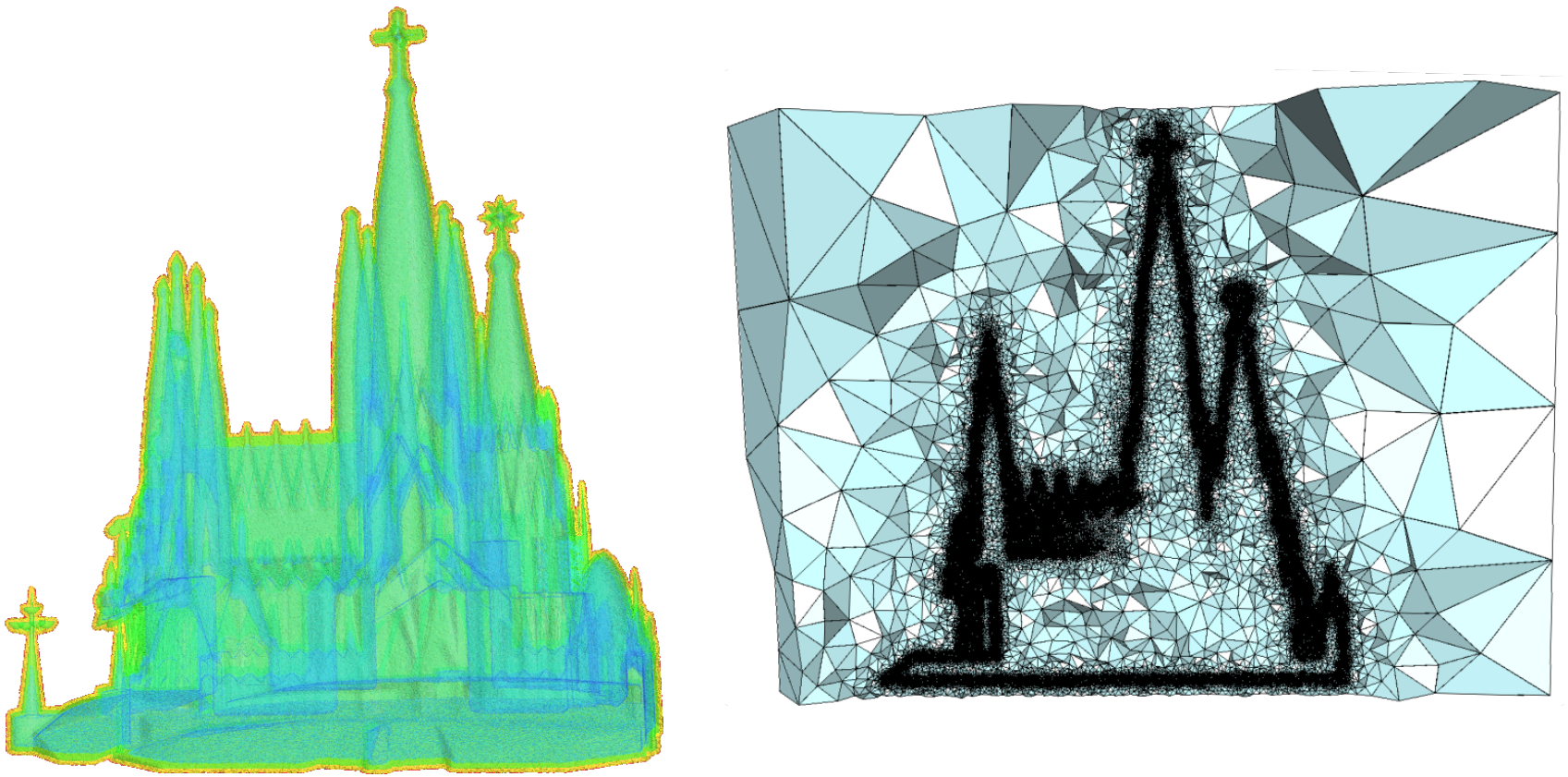


Isolines of d_Ω at the vertices of a mesh $\mathcal{T}$ of a bounding box D.



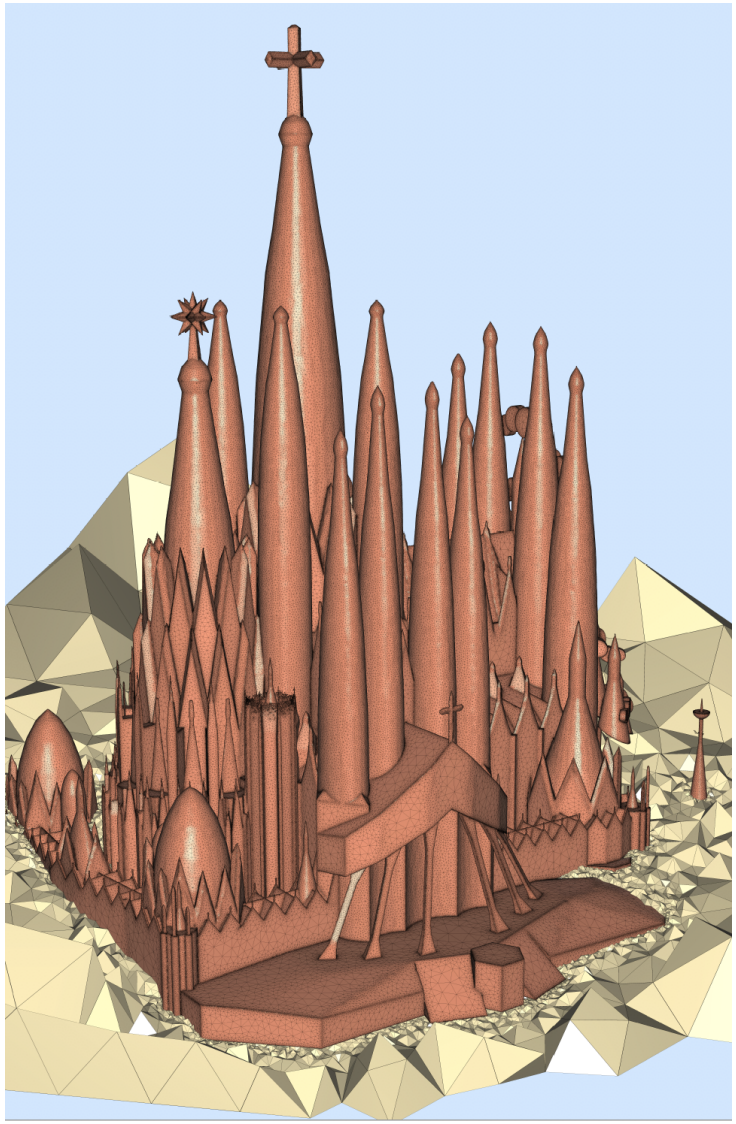
New mesh $\widetilde{\mathcal{T}}$ of D, enclosing Ω as a submesh.

Volume mesh generation from an invalid surface triangulation (III)



(Left) isosurfaces of the signed distance function to the "Sagrada Familia", calculated at the vertices of an adapted mesh (right).

Volume mesh generation from an invalid surface triangulation (IV)



Reconstructed mesh by using the isosurface discretization operation from the signed distance function to Ω.

1. Basic theory of remeshing
   - A few definitions and key concepts
   - Why remeshing?
   - Remeshing in practice

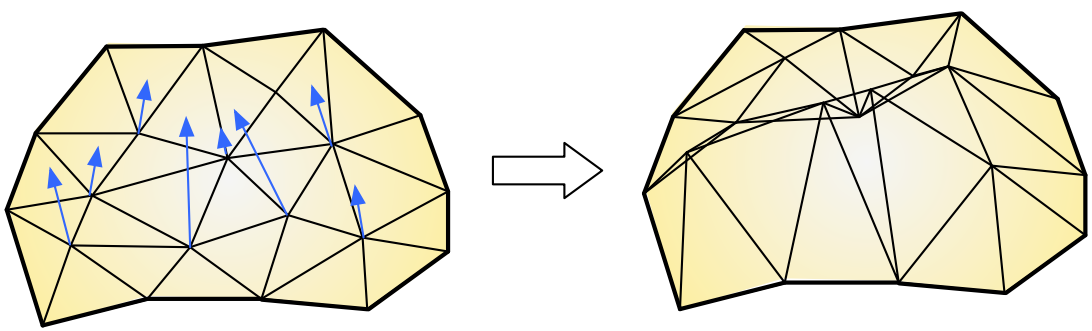
2. Applications of remeshing
   - Adaptation to a user-defined size prescription
   - Isosurface discretization and volume mesh generation
   - Lagrangian motion of a domain
   - Body-fitted interface tracking

3. A glimpse on a recent challenge: parallel remeshing

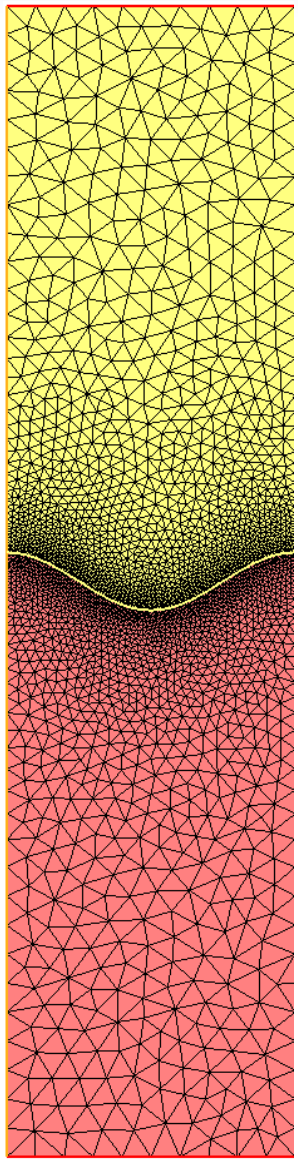
Tracking domain evolution (I)

- Many physical phenomena occur on a moving domain $\Omega(t)$.
- Accurate simulations rely on an exact discretization of $\Omega(t)$ at all times.
- Simple "Lagrangian strategies" involve the deformation of a mesh $\mathcal{T}$ according to a velocity field $V(x)$...

... which is a notoriously difficult operation!



Moving the vertices of a mesh according to a velocity field $V(x)$ is prone to produce overlapping elements.

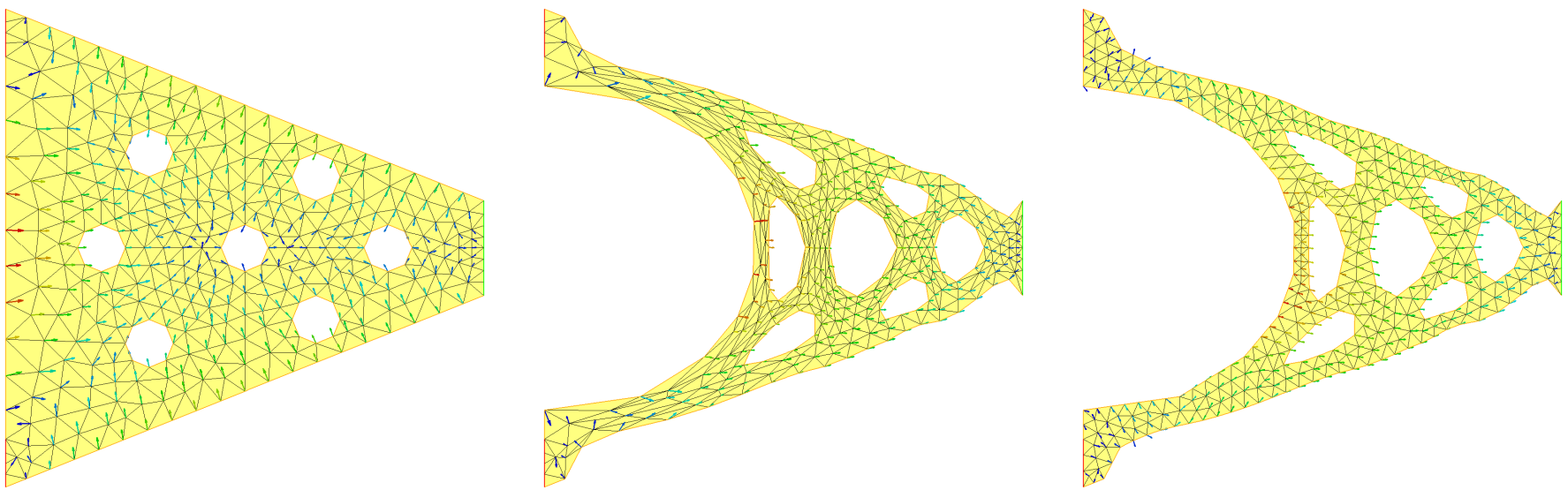


A Rayleigh-Taylor instability in computational fluid dynamics

Tracking domain evolution (II)

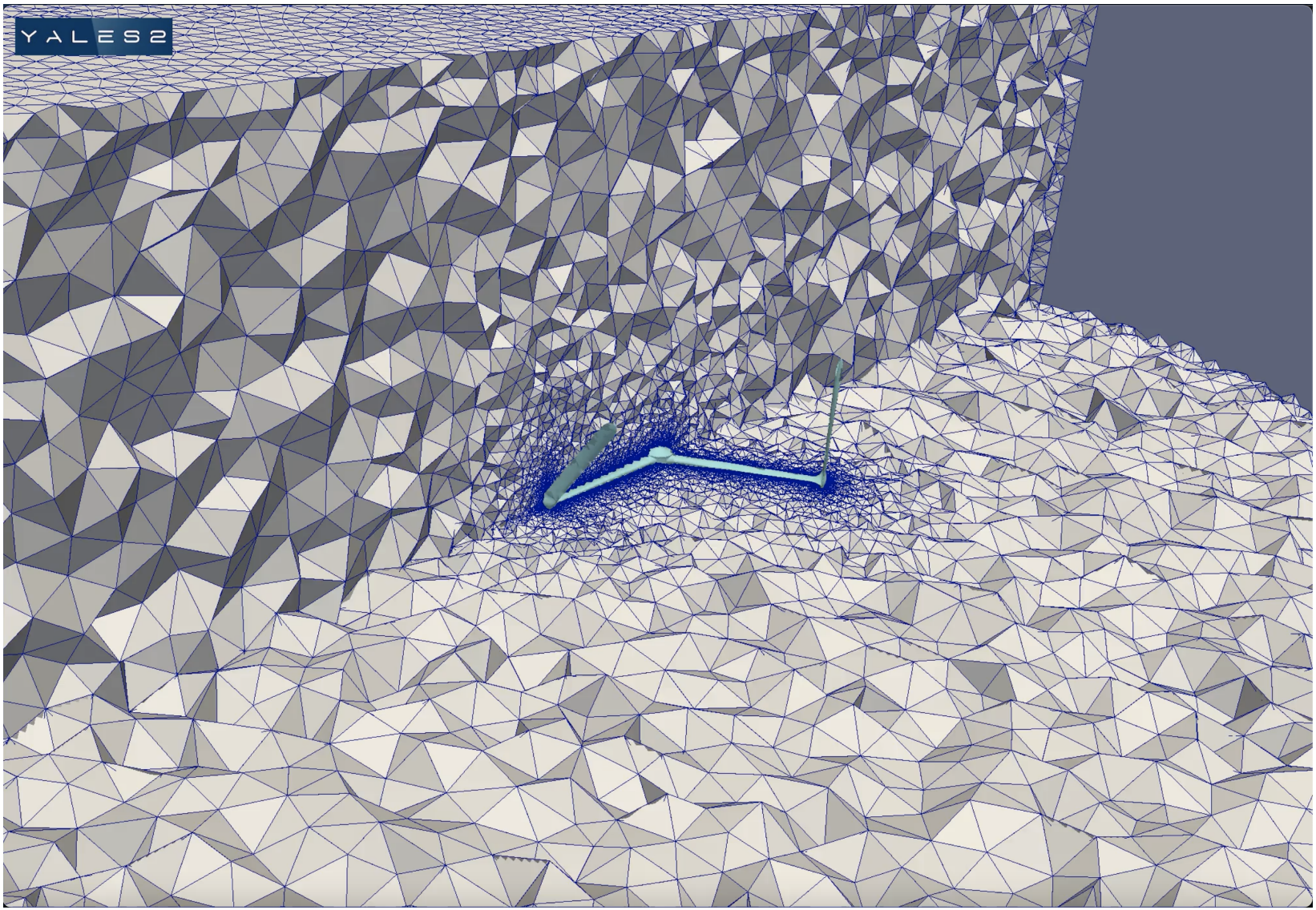
One deformation strategy of a mesh $\mathcal{T}$ along a vector field $V(x)$ over $(0, T)$ reads:

1. Find the largest time τ such that moving each vertex $x \in \mathcal{T}$ to $x + \tau V(x)$ results in a valid mesh;
2. Apply remeshing to improve the quality of the resulting mesh;
3. If $\tau = T$, the motion is complete. Else, return to 1 for the remaining time.



(Left) initial shape and associated deformation field; (middle) the mesh has become very stretched, nearly invalid; (right) after quality-oriented remeshing, the deformation process can resume.

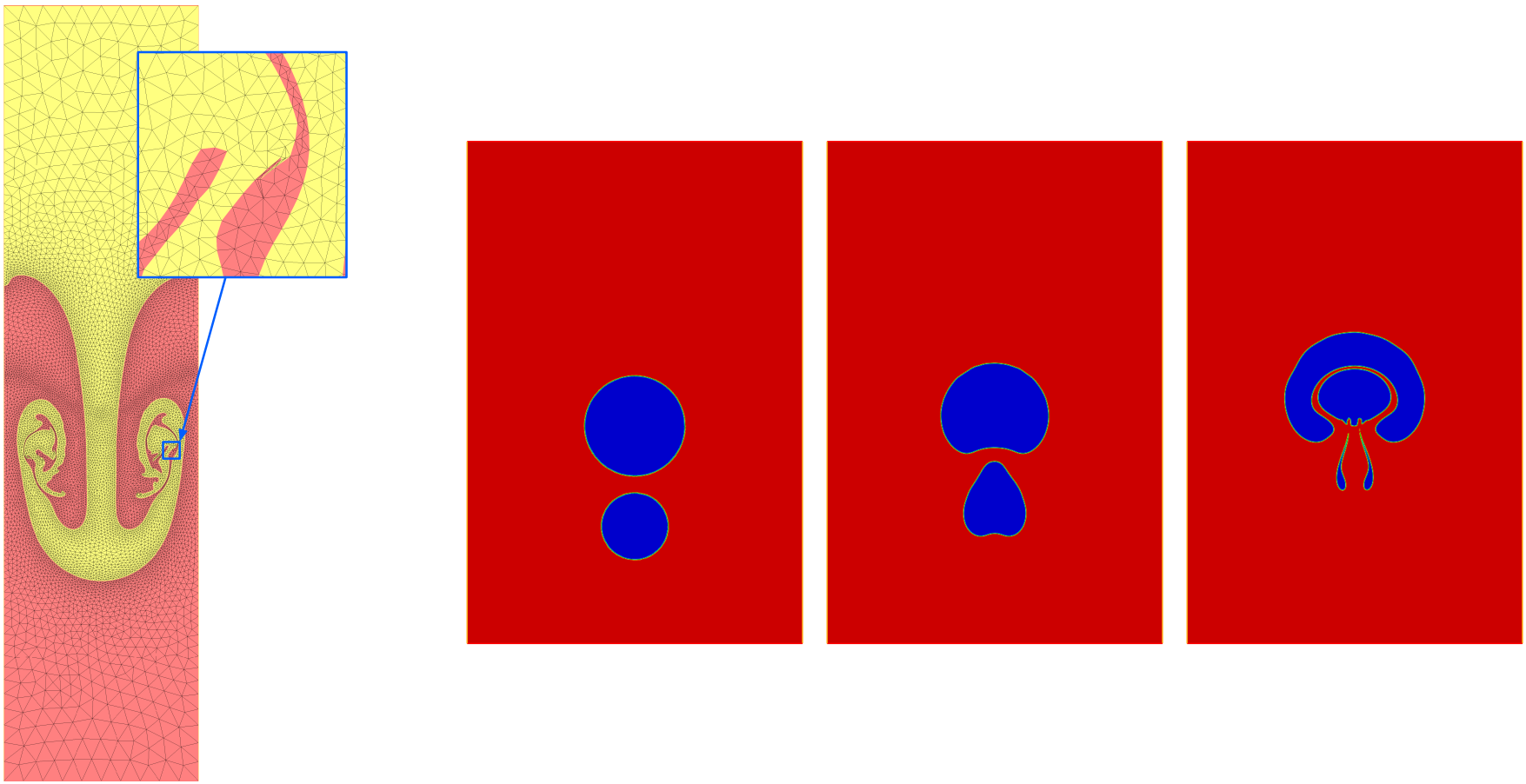
Tracking domain evolution (III)



Rotation of a blade by Lagrangian motion using the library `Yales2` *[Yales2].*

Tracking domain evolution (IV)

Even though "clever" heuristics may improve their efficiency, such Lagrangian procedures are usually reserved to relatively mild deformations.



(Left) As the evolving shape gets very stretched, the quality of the mesh tends to degenerate; (right) example of a motion involving topological changes, which is typically difficult to realize by mesh deformation.

1. Basic theory of remeshing
   - A few definitions and key concepts
   - Why remeshing?
   - Remeshing in practice

2. Applications of remeshing
   - Adaptation to a user-defined size prescription
   - Isosurface discretization and volume mesh generation
   - Lagrangian motion of a domain
   - Body-fitted interface tracking

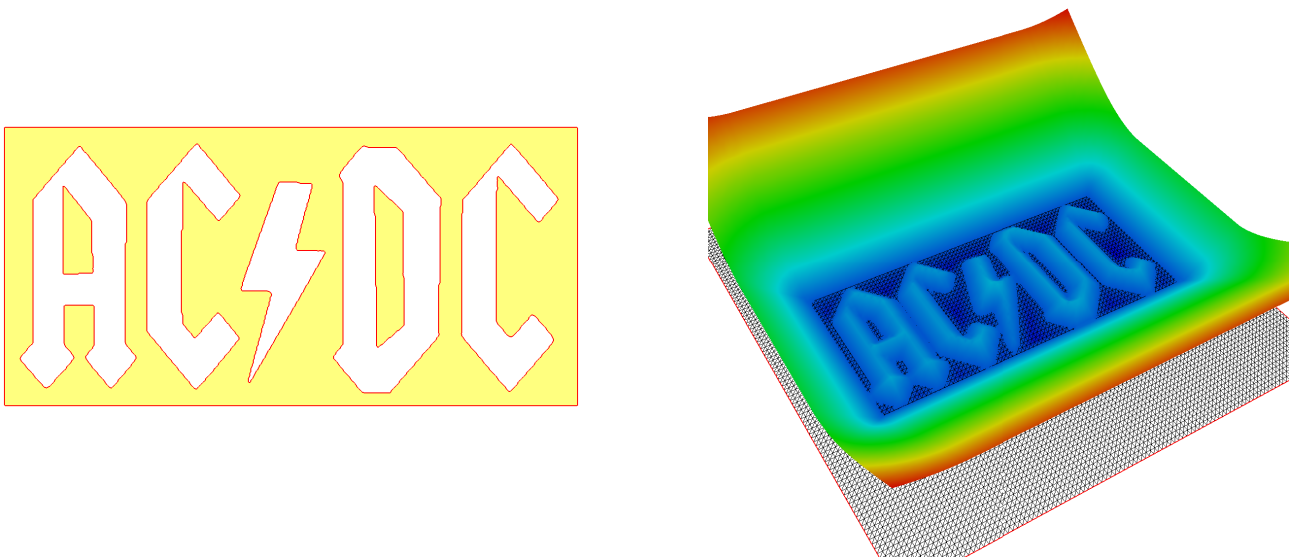
3. A glimpse on a recent challenge: parallel remeshing

A short digression: the level set method (I)

A paradigm: *the motion of an evolving domain is conveniently described in an implicit way.*

A domain $\Omega \subset \mathbb{R}^d$ is equivalently defined by a function $\phi : \mathbb{R}^d \to \mathbb{R}$ such that:

$$\phi(x) < 0 \quad \text{if } x \in \Omega \quad ; \quad \phi(x) = 0 \quad \text{if } x \in \partial\Omega \quad ; \quad \phi(x) > 0 \quad \text{if } x \in {}^c\overline{\Omega}$$



One domain $\Omega \subset \mathbb{R}^2$ (left), graph of an associated level set function (right).

A short digression: the level set method (II)

- Let $\Omega(t) \subset \mathbb{R}^d$ be a domain moving according to a velocity field $v(t,x) \in \mathbb{R}^d$.
- Let $\phi(t,x)$ be a level set function for $\Omega(t)$.
- The motion of $\Omega(t)$ translates in terms of ϕ as the level set advection equation:

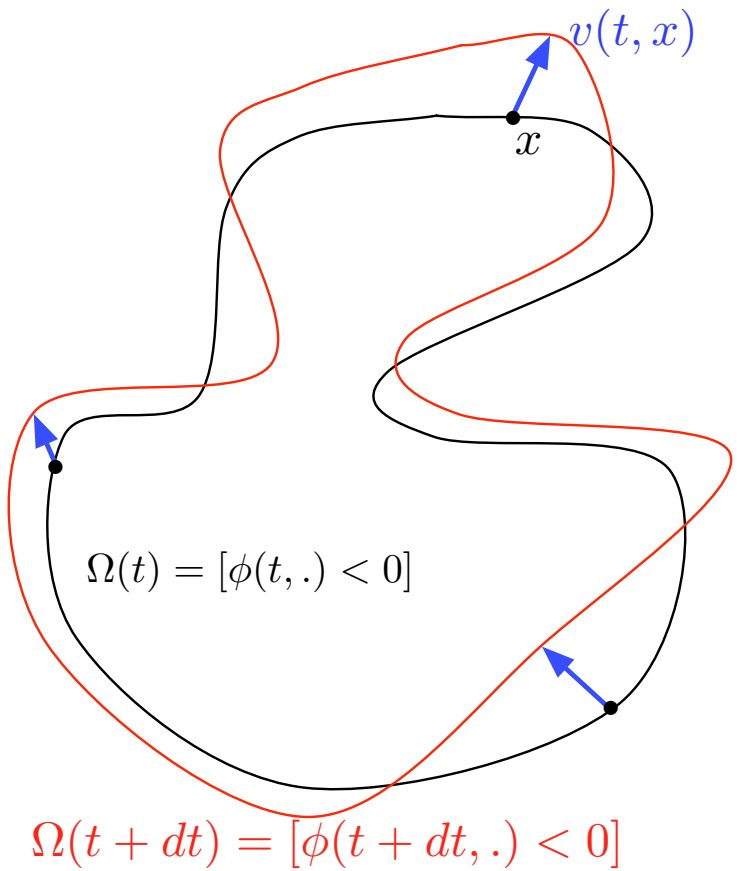
$$\frac{\partial \phi}{\partial t}(t,x) + v(t,x).\nabla\phi(t,x) = 0$$

- If $v(t,x)$ is normal to the boundary $\partial\Omega(t)$, i.e.:

$$v(t,x) := V(t,x)\frac{\nabla\phi(t,x)}{|\nabla\phi(t,x)|},$$

this rewrites as a Hamilton-Jacobi equation:

$$\frac{\partial \phi}{\partial t}(t,x) + V(t,x)|\nabla\phi(t,x)| = 0$$



The "classical" practice of the level set method

In its "classical" implementation, the level set method is conducted on a fixed mesh $\mathcal{T}$ of the computational domain D:

- The time interval $(0, T)$ is discretized as $0 = t_0 < t_1 < \ldots < t_N = T$.
- For all $n = 0, \ldots, N$, the domain $\Omega(t^n)$ is solely known as the negative subdomain

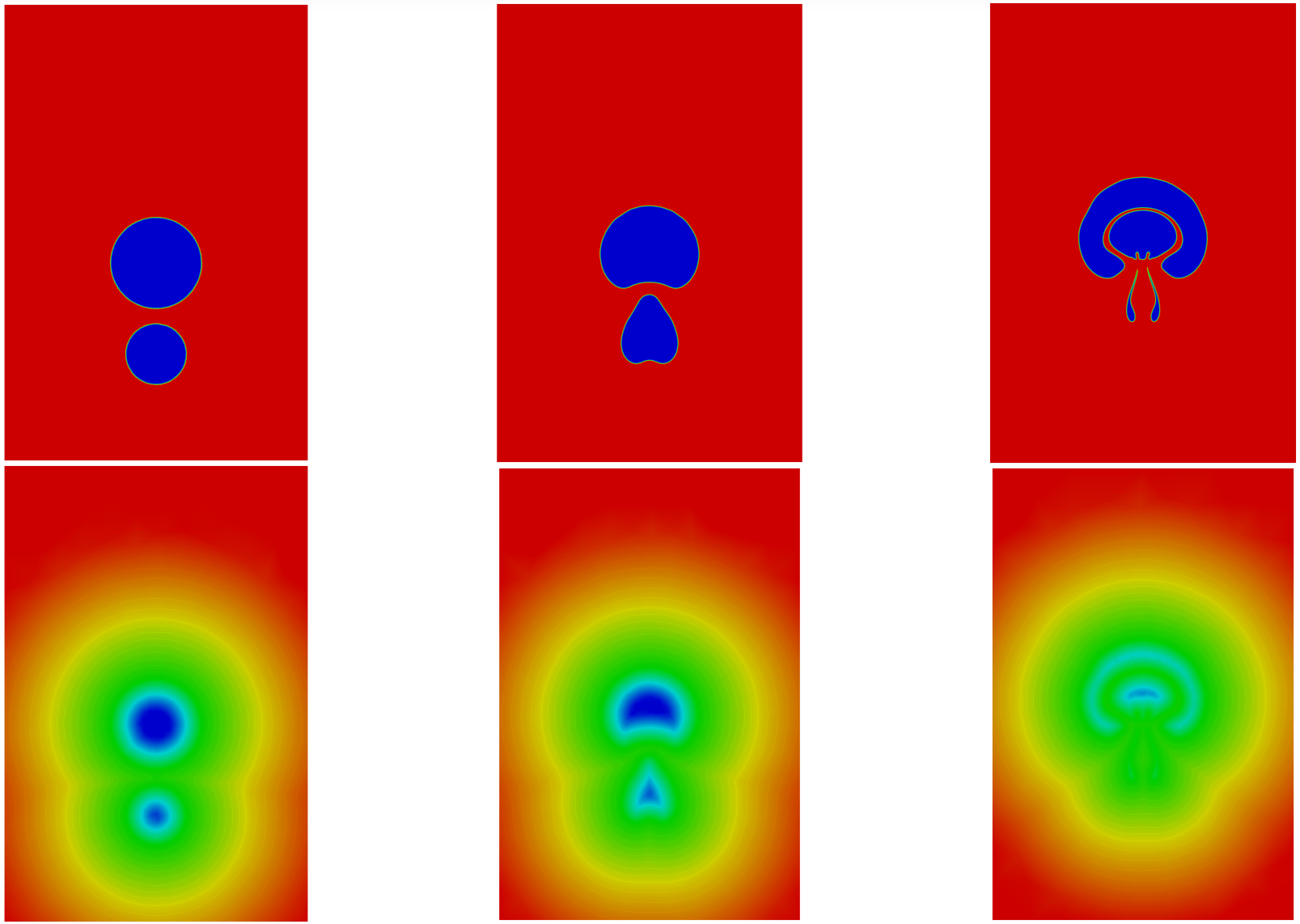
$$\Omega(t^n) = \Big\{ x \in D, \;\; \phi(t^n, x) < 0 \Big\},$$

  where the level set function $\phi(t^n, \cdot)$ is discretized at the vertices of $\mathcal{T}$.
- The motion $\Omega(t^n) \to \Omega(t^{n+1})$ is realized by solving the level set equation

$$\begin{cases} \frac{\partial \phi}{\partial t}(t,x) + v(t,x) \cdot \nabla \phi(t,x) = 0 & \text{for } (t,x) \in (t^n, t^{n+1}) \times D, \\ \phi(t^n, x) & \text{is given for } x \in D \end{cases}$$

  on the fixed mesh $\mathcal{T}$.
- **Drawback:** Since $\Omega(t)$ is never discretized, the physical equation of the motion, producing the velocity field $v(t,x)$ has to be approximated by an equation on D.

The "classical" practice of the level set method



(Upper row) Evolution of a rising bubble of fluid immersed in another, more dense fluid; (lower row) values of associated level set functions.

Isosurface discretization (III)

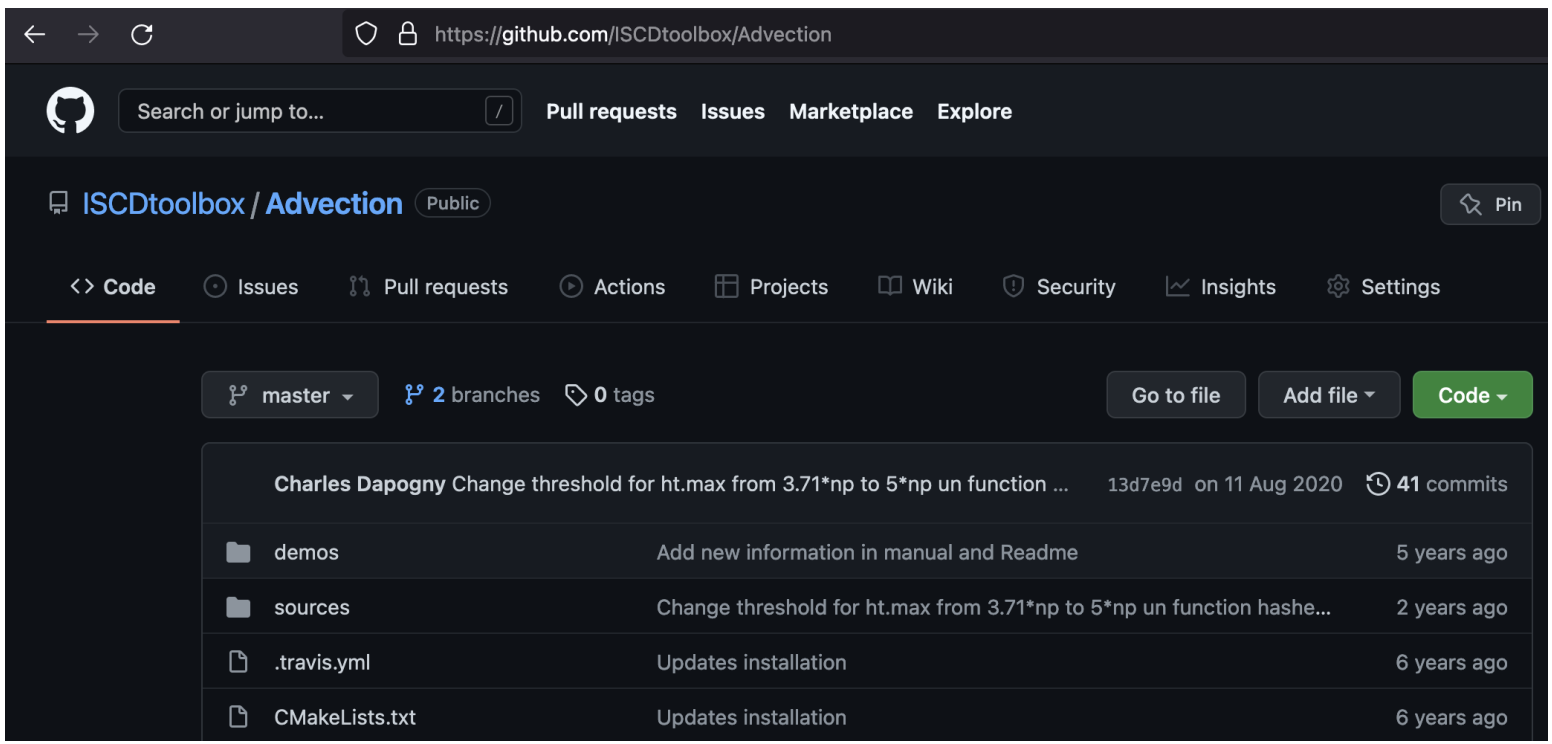
Open-source implementations dedicated to the level set method:

- Distancing / redistancing operations: `mshdist` [DaFre]

  `https://github.com/ISCDtoolbox/Mshdist`

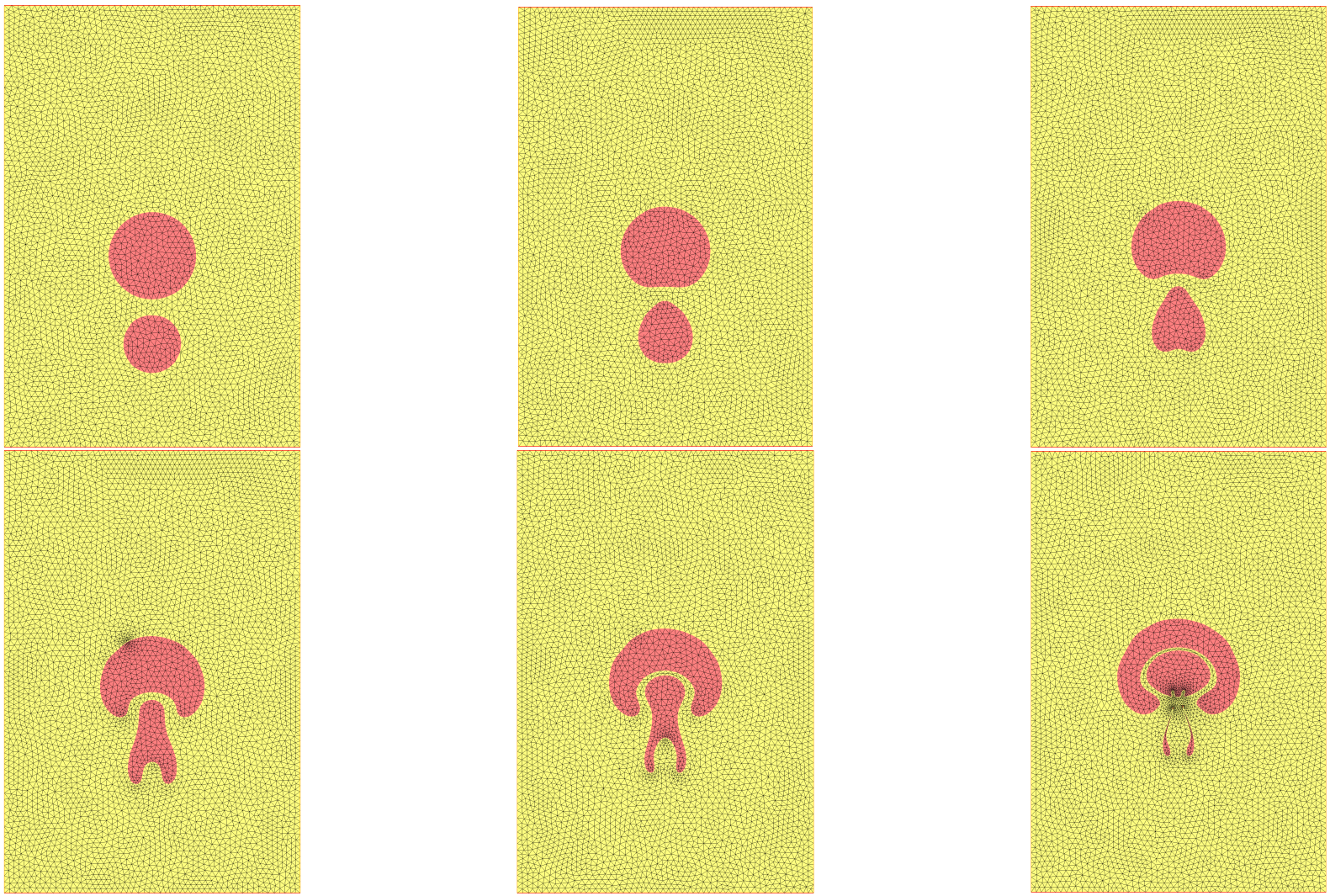
- Resolution of the level set advection equation: `advect` [BuDaFre]

  `https://github.com/ISCDtoolbox/Advection`



Body-fitted interface tracking

Using the isosurface discretization operation, this program can be carried out while modifying the computational mesh at each iteration.



Evolution of the rising bubble by using the combination of the level set method with isosurface discretization.

Example: shape optimization (I)

- Shape optimization aims at improving the performance of the initial design Ω^0 of a mechanical structure (e.g. a beam, a mechanical actuator,...) or a fluid duct, with respect to a physical criterion.
- The problem arises under the form:

$$\min_{\Omega \in \mathcal{U}_{\mathrm{ad}}} J(\Omega),$$

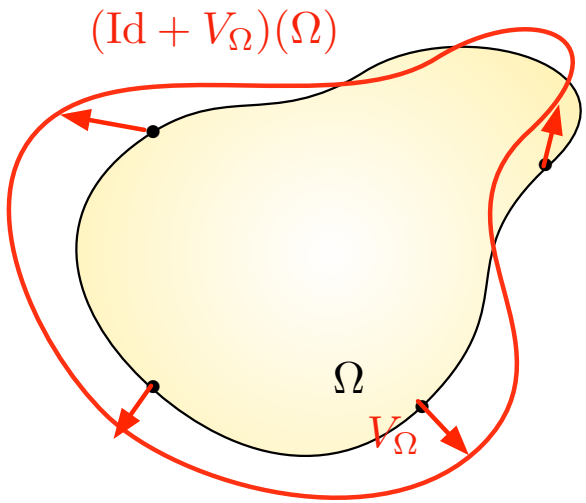
where

  - $J(\Omega)$ is a cost functional, depending on Ω in a possibly very complicated way (via the solution to a PDE posed on Ω). For instance,
    - When Ω is a structure, $J(\Omega)$ may be the work of external forces on Ω, a vibration frequency, etc.
    - When Ω is a fluid duct, $J(\Omega)$ may account for the work of viscous forces inside Ω.
  - $\mathcal{U}_{\mathrm{ad}}$ is a set of admissible designs, which encompasses, e.g. volume, or manufacturability constraints.

Example: shape optimization (II)

- Techniques from shape optimization make it is possible to calculate a shape gradient at a shape Ω, i.e. a vector field $V_\Omega : \mathbb{R}^d \to \mathbb{R}^d$ such that:

$$J((\mathrm{Id} + \tau V_\Omega)(\Omega)) < J(\Omega), \quad \text{for } \tau > 0 \text{ small enough.}$$



- Starting from an initial design Ω^0, the sequence of shapes

$$\Omega^{n+1} := (\mathrm{Id} + \tau^n V_{\Omega^n})(\Omega^n), \quad \text{where } \tau^n \text{ is a pseudo-time step,}$$

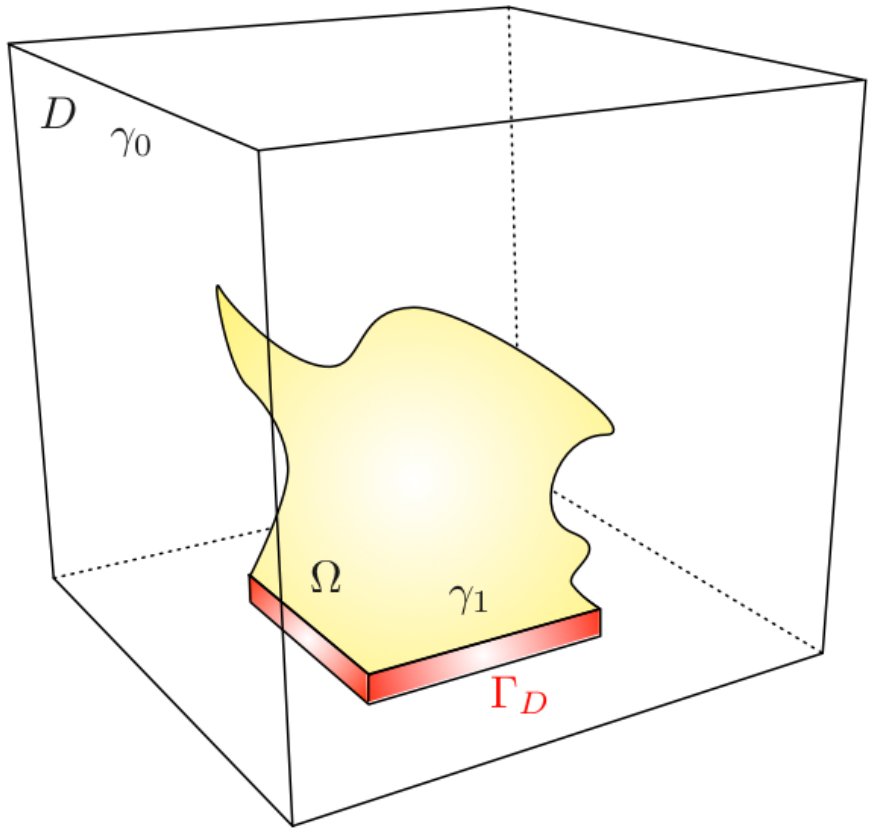
evolves by decreasing the criterion $J(\Omega)$.

Optimization of the shape of a heat diffuser (I)

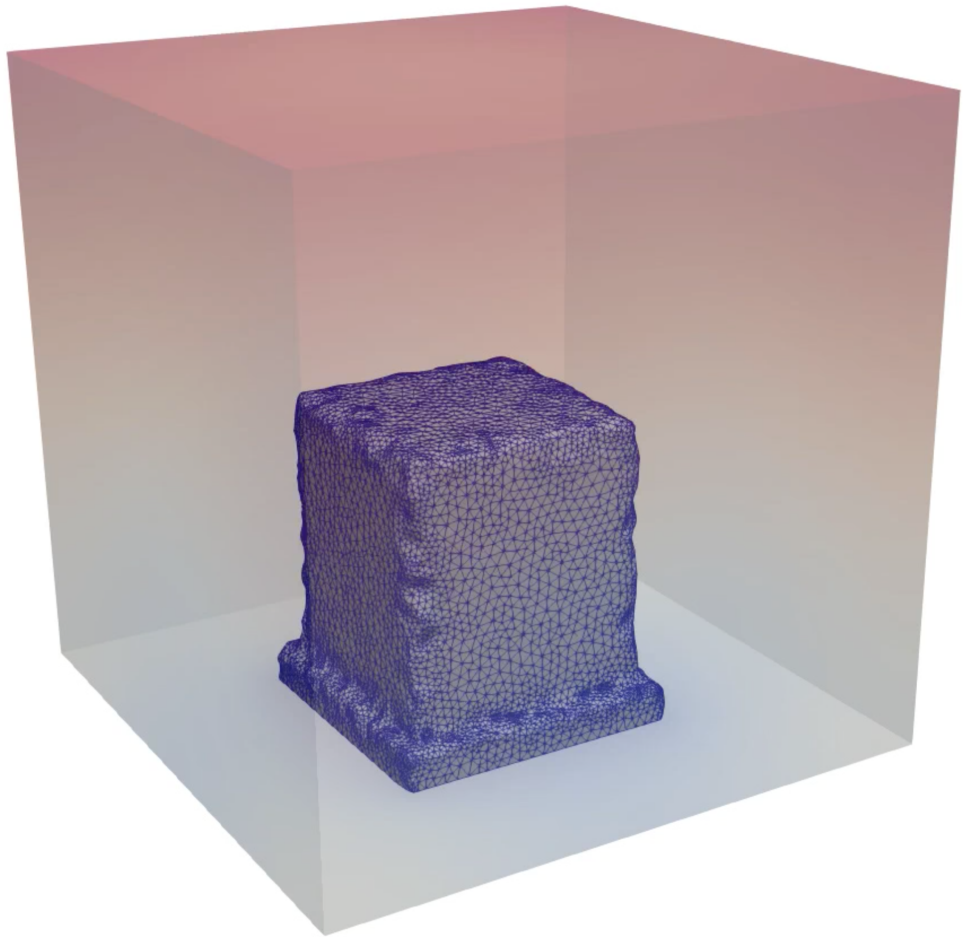
- A thermal chamber D is divided into
  - A phase Ω with high conductivity γ_1
  - A phase $D \setminus \overline{\Omega}$ with low conductivity γ_0.
- A temperature $T_0 = 0$ is imposed on Γ_D and the remaining boundary $\partial D \setminus \overline{\Gamma_D}$ is insulated from the outside.
- A heat source is acting inside D.
- The temperature u_Ω inside D is solution to the two-phase Laplace equation.
- The average temperature inside D,

$$J(\Omega) = \frac{1}{|D|} \int_D u_\Omega \, \mathrm{d}x$$

is minimized under a volume constraint.



Optimization of the shape of a heat diffuser (II)



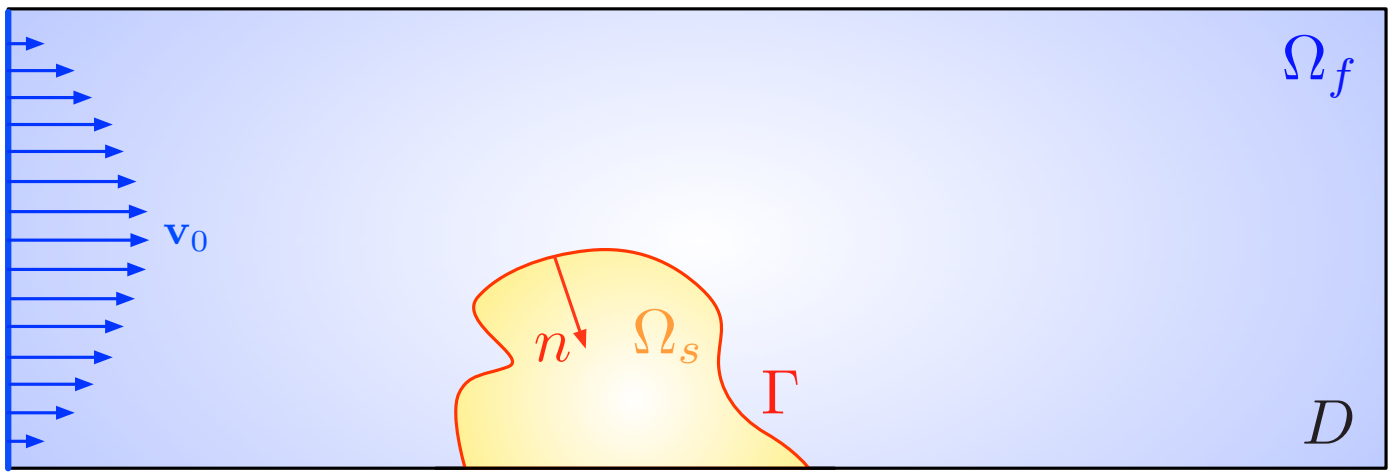
Optimization of the shape of a heat diffuser, from [FeAlDaJo].

An advanced example in fluid-structure interaction (I)

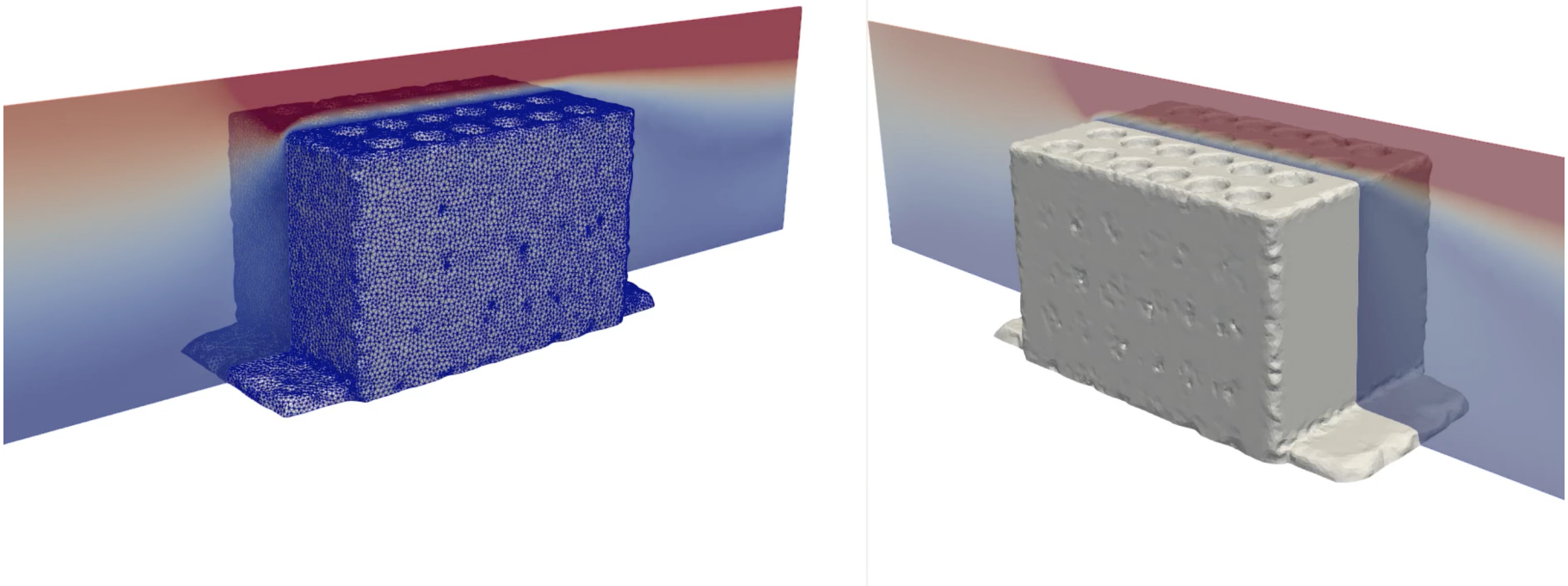
- A solid obstacle $\Omega_s := \Omega$ is placed inside a fixed cavity D where a fluid is flowing, occupying the phase $\Omega_f := D \setminus \Omega_s$.
- The fluid obeys the Navier-Stokes equations ($Re = 60$), and the solid is governed by the linearized elasticity system.
- Weak coupling between Ω_f and Ω_s: the fluid exerts a traction on the interface Γ.
- We optimize the shape of Ω_s with respect to the solid compliance

$$J(\Omega) = \int_{\Omega_s} Ae(u_{\Omega_s}) : e(u_{\Omega_s}) \, \mathrm{d}x,$$

under a volume constraint.



An advanced example in fluid-structure interaction (II)



Optimization of the shape of a mast withstanding an incoming flow in 3d, from [FeAlDaJo].

1. Basic theory of remeshing
   - A few definitions and key concepts
   - Why remeshing?
   - Remeshing in practice
2. Applications of remeshing
   - Adaptation to a user-defined size prescription
   - Isosurface discretization and volume mesh generation
   - Lagrangian motion of a domain
   - Body-fitted interface tracking
3. A glimpse on a recent challenge: parallel remeshing

Parallel remeshing

- The dramatic increase in computational resources paves the way to more and more realistic numerical simulations. based on large-scale meshes.

- This raises the need to process (remesh, adapt) very large meshes, that can barely be stored in memory.

- A burning challenge is then to perform remeshing in parallel, on distributed memory architectures: each process should receive and treat a portion of the mesh independently of what happens on the other processes.

Parallel remeshing

- The mesh $\mathcal{T}$ is partitioned into disjoint submeshes $\mathcal{T}_k$, $k = 1, \ldots, K$.
- The edges and triangles lying at the interface between different submeshes form the overlap mesh $\mathcal{O}$.
- Each part $\mathcal{T}_k$ is modified on its corresponding processor, independently of the treatment of the other submeshes.

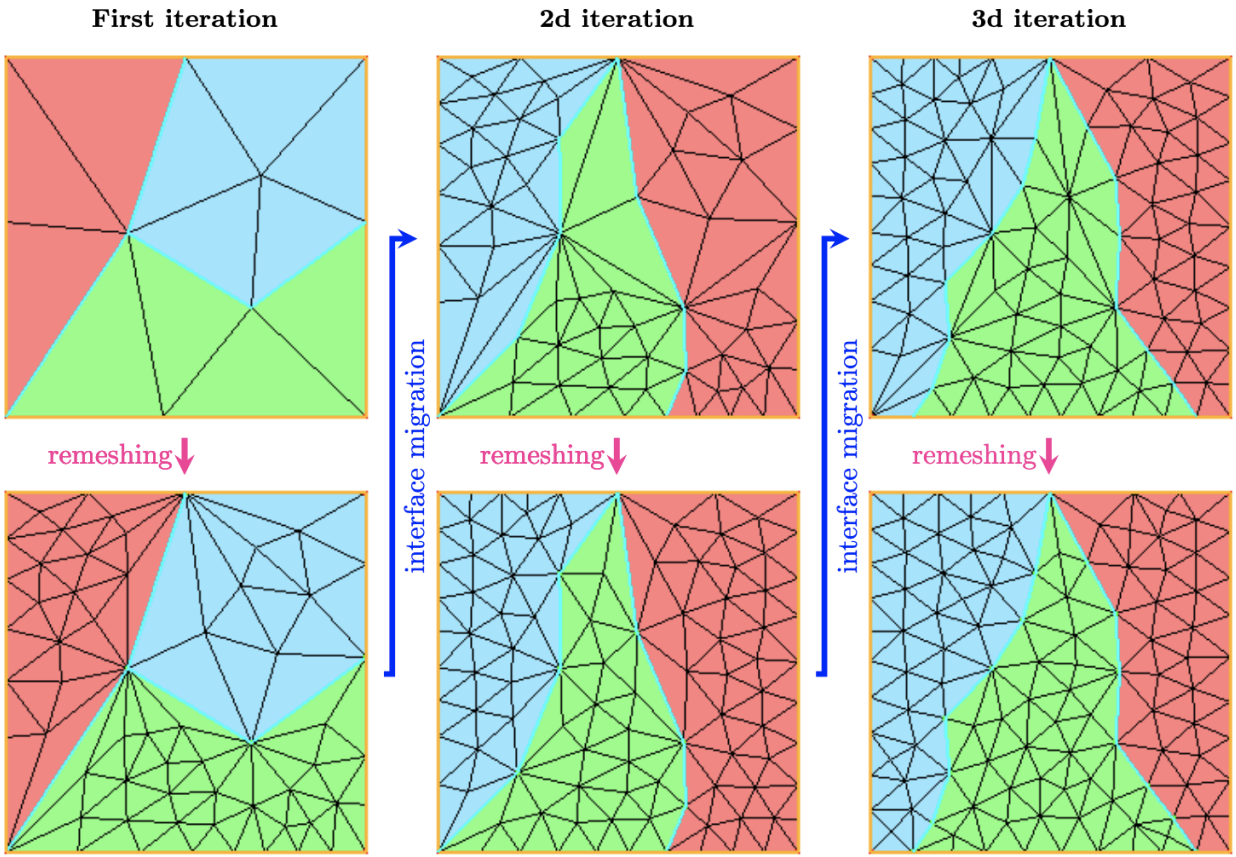
Difficulties:

- To achieve full efficiency, it is desirable that the different processes bear comparable work loads.... a feature which is difficult to predict.
- Specific data structures are needed for elements $T \in \mathcal{T}$ sharing an interface entity in $\mathcal{O}$ (update of the neighboring elements treated on other processes, etc), which have to be maintained.

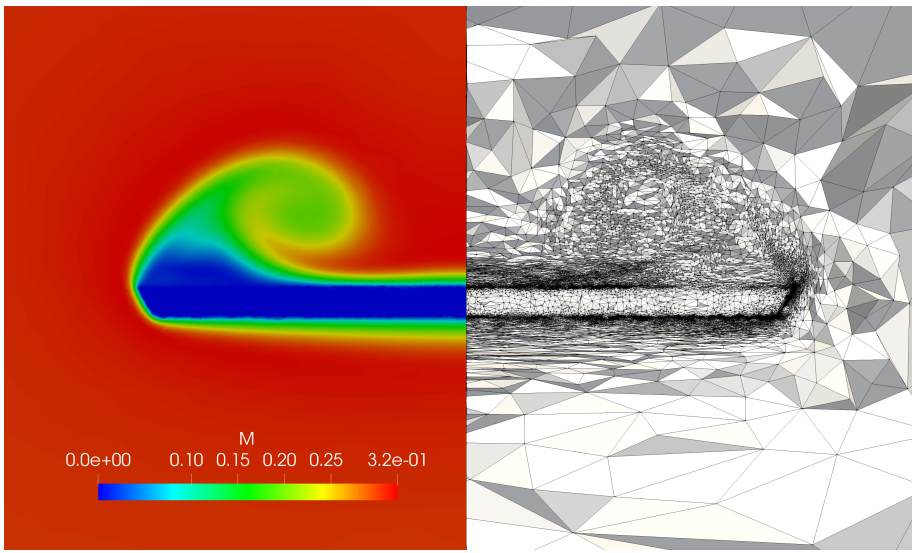
Parallel remeshing

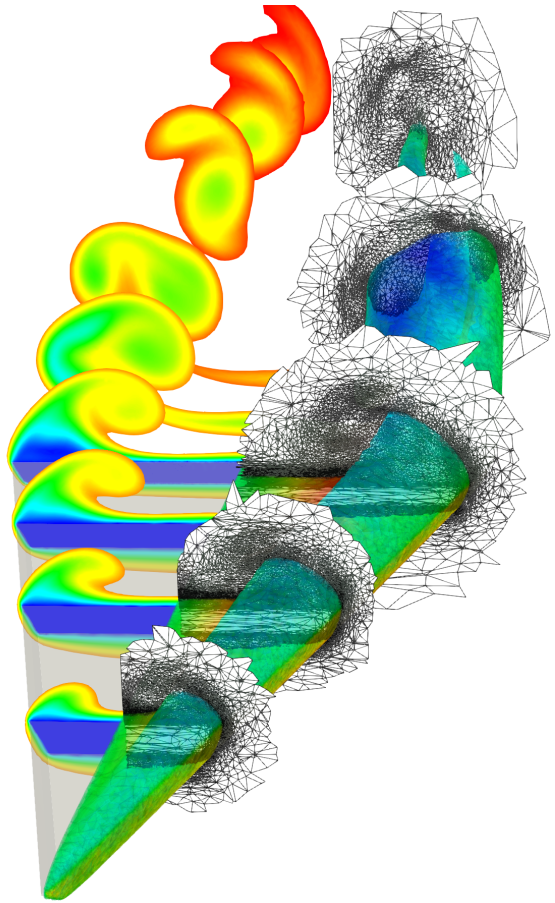
One solution: iterative remeshing-repartitioning procedure: For $n = 0, \ldots,$

1. Create a new partition of $\mathcal{T}$ into K disjoint submeshes $\mathcal{T}_1, \ldots \mathcal{T}_k$, and create or update the structure for the overlap mesh $\mathcal{O}$.
2. Apply the sequential remesher to all the $\mathcal{T}_k$ while leaving the entities in $\mathcal{O}$ unmodified;



Parallel remeshing





Simulation of a turbulent flow around a delta wing on a mesh with 28M tetrahedra; (left) slices of the Mach number and adapted mesh; (right) flow and mesh at the trailing edge (from [ArBeCi]).

Thank you for your attention !

References I

[AbAlBeDoNou] R. Abgrall, H. Alcin, H. Beaugendre, C. Dobrzynski and L. Nouveau, *Residual schemes applied to an embedded method expressed on unstructured adapted grids*, In Eighth International Conference on Computational Fluid Dynamics (ICCFD8) (2014).

[AlDaFre] G. Allaire, C. Dapogny, and P. Frey, *Shape optimization with a level set based mesh evolution method*, Comput Methods Appl Mech Eng, 282 (2014), pp. 22–53.

[ArBeCi] L. Arpaia, H. Beaugendre, L. Cirrottola, A. Froehly, M. Lorini , L. Nouveau and M. Ricchiuto, *h− and r− adaptation on simplicial meshes using MMG tools*, (2021), arXiv preprint arXiv:2109.08451.

[BuDaFre] C. Bui, C. Dapogny and P. Frey, *An accurate anisotropic adaptation method for solving the level set advection equation*, International Journal for Numerical Methods in Fluids, 70(7), (2012), pp. 899–922.

[DaFre] C. Dapogny and P. Frey, *Computation of the signed distance function to a discrete contour on adapted triangulation*, Calcolo, 49(3), (2012), pp. 193–219.

References II

[DaDobFre] C. Dapogny, C. Dobrzynski and P. Frey, *Three-dimensional adaptive domain remeshing, implicit domain meshing, and applications to free and moving boundary problems*, J. Comput. Phys., 262, (2014), pp. 358–378.

[FeAlDaJo] F. Feppon, G. Allaire, C. Dapogny and P. Jolivet, *Topology optimization of thermal fluid–structure systems using body-fitted meshes and parallel computing*, Journal of Computational Physics, 417, (2020), 109574.

[FreGeo] P.J. Frey and P.L. George, *Mesh Generation : Application to Finite Elements*, Wiley, 2nd Edition, (2008).

[OFed] S.J. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Springer Verlag, (2003).

[OSe] S. J. Osher and J.A. Sethian, *Front propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations,* J. Comp. Phys. **78** (1988) pp. 12-49

[SethianFMM] J.A. Sethian, *A fast marching level set method for monotonically advancing fronts*, Proc. Natl. Acad. Sci. USA Vol. 93, (1996), pp. 1591–1595.

References III

[Sethian] J.A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, Cambridge University Press, (1999).

[VaHeMan] M. Vallet, F. Hecht, and B. Mantel, *Anisotropic control of mesh generation based upon a Voronoi type method,* Numerical grid generation in computational fluid dynamics and related fields, (1991), pp. 93–103.

[Zhao] H. Zhao, *A fast sweeping method for eikonal equations*, Mathematics of computation, 74(250), (2005), pp. 603–627.

[Yales2] `Yales2` *library*, `https://www.coria-cfd.fr/index.php/YALES2`.