Shape optimization with a level-set based mesh deformation method

G. Allaire[1], Ch. Dapogny[1,2,3], and P. Frey[2]

[1] CMAP, UMR 7641 École Polytechnique, Palaiseau, France

[2] Laboratoire J.L. Lions, UPMC, Paris, France

[3] Technocentre Renault, Guyancourt

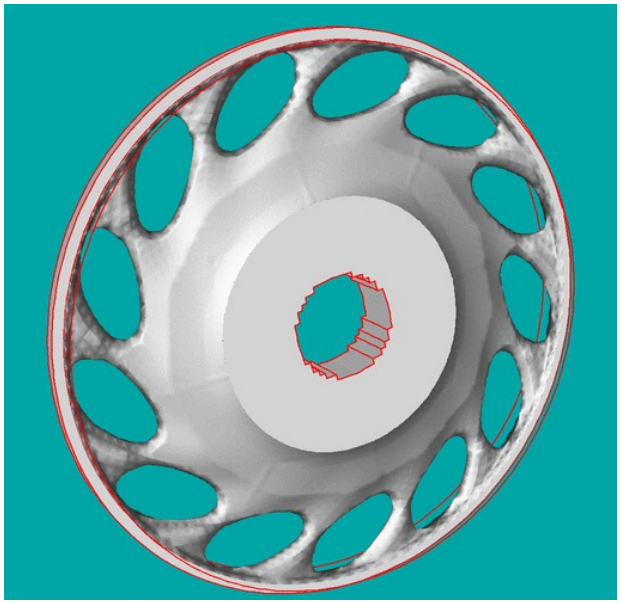
Shape optimization and industrial applications

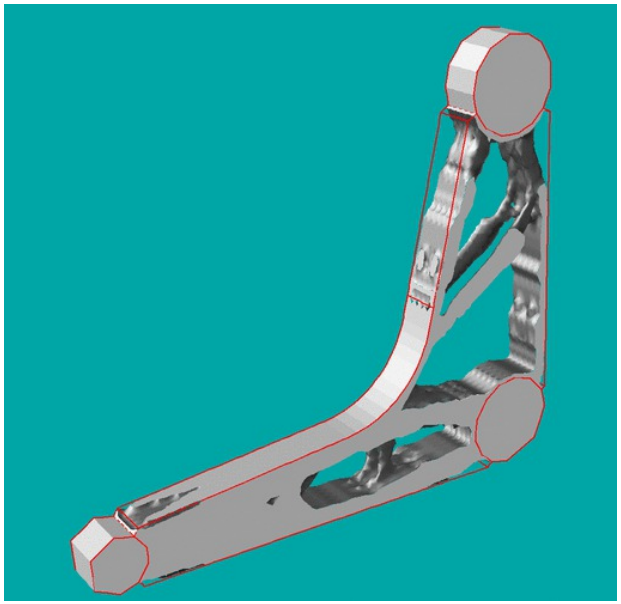
In industry, there is a growing need for optimizing mechanical parts from the early stages of design.

Such problems are difficult, because

- they are highly dependent on the mechanical(s) problem(s) at stake.
- they require an accurate description of the various shapes that could be obtained through the optimization process.

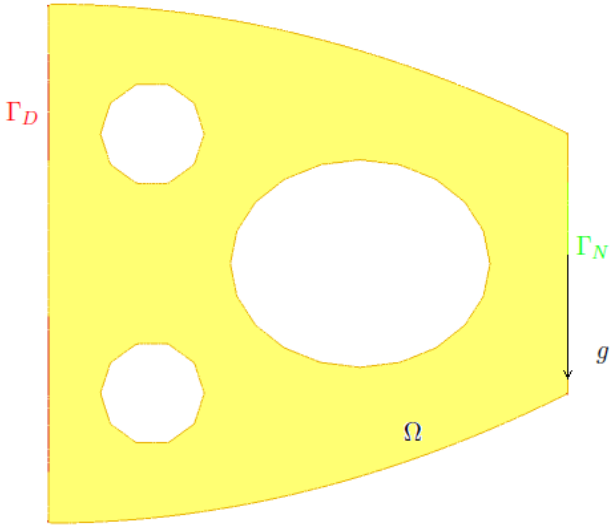
Basically, engineers work by trial and error, and highly rely on physical intuition, but automatic techniques that could lead to non-intuitive designs would prove much more efficient.





A model problem in linear elasticity

A structure is represented by a bounded open domain $\Omega \subset \mathbb{R}^d$, fixed on a part $\Gamma_D \subset \partial\Omega$ of its boundary, and submitted to a load case g (for the sake of simplicity, body forces are omitted), to be applied on $\Gamma_N \subset \partial\Omega$, $\Gamma_D \cap \Gamma_N = \emptyset$.



A 'Cantilever'

The displacement vector field $u_\Omega : \Omega \to \mathbb{R}^d$ is governed by the linear elasticity system :

$$\begin{cases} -div(Ae(u_\Omega)) = 0 & \text{in } \Omega \\ u_\Omega = 0 & \text{on } \Gamma_D \\ Ae(u_\Omega).n = g & \text{on } \Gamma_N \\ Ae(u_\Omega).n = 0 & \text{on } \Gamma := \partial\Omega \setminus (\Gamma_D \cup \Gamma_N) \end{cases},$$

where $e(u) = \frac{1}{2}({}^t\nabla u + \nabla u)$ is the strain tensor field, $Ae(u) = 2\mu e(u) + \lambda tr(e(u))I$ is the stress tensor, and λ, μ are the Lamé coefficients of the material.

The displaced cantilever

A model problem in linear elasticity

goal : Given an initial structure Ω_0, find a new domain Ω that minimizes a certain functional of the domain $J(\Omega)$, under a volume constraint.

Example : The work of the external loads g or compliance $c(\Omega)$ of domain Ω :

$$c(\Omega) = \int_\Omega Ae(u_\Omega) : e(u_\Omega)dx = \int_{\Gamma_N} g.u_\Omega \, ds$$

The volume constraint is enforced with a fixed penalty parameter ℓ :

$$\Rightarrow \text{minimize } J(\Omega) := c(\Omega) + \ell \, Vol(\Omega).$$

Outline

I. Mathematical modeling of shape optimization problems
   1. Differentiation with respect to the domain : Hadamard's method
   2. The generic numerical algorithm

II. From meshed domains to a level set description,... and conversely
   1. A few words about the level set Method
   2. Initializing level set functions with the signed distance function
   3. Meshing the negative subdomain of a level set function : local remeshing

III. Application to shape optimization
   1. Numerical Implementation
   2. The algorithm in motion
   3. Some numerical results

Outline

I. Mathematical modeling of shape optimization problems
   1. Differentiation with respect to the domain : Hadamard's method
   2. The generic numerical algorithm

II. From meshed domains to a level set description,... and conversely
   1. A few words about the level set Method
   2. Initializing level-set functions with the signed distance function
   3. Meshing the negative subdomain of a level set function : local remeshing

III. Application to shape optimization
   1. Numerical Implementation
   2. The algorithm in motion
   3. Some numerical results

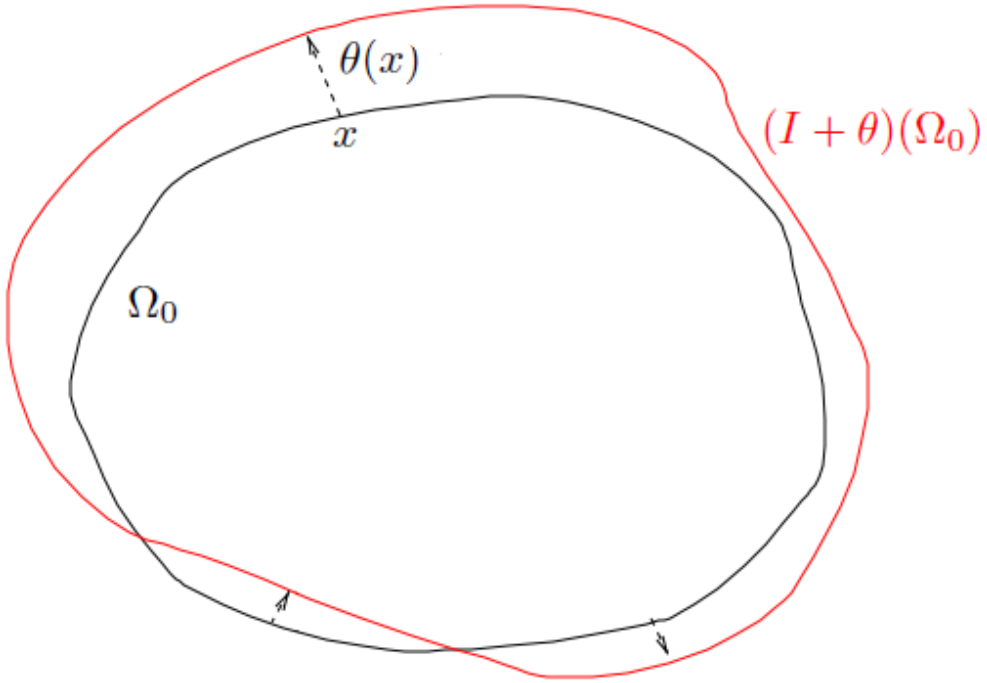
Differentiation with respect to the domain : Hadamard's method

LEMMA 1 *For all* $\theta \in W^{1,\infty}\left(\mathbb{R}^d, \mathbb{R}^d\right)$ *with norm* $||\theta||_{W^{1,\infty}(\mathbb{R}^d,\mathbb{R}^d)} < 1$, $(I+\theta)$ *is a Lipschitz diffeomorphism of* $\mathbb{R}^d$, *with Lipschitz inverse.*

Given a reference, smooth domain Ω_0, we consider variations of Ω_0 of the form

$$\Omega_0 \to (I+\theta)(\Omega_0)$$

for $\theta \in W^{1,\infty}\left(\mathbb{R}^d, \mathbb{R}^d\right)$.



Differentiation with respect to the domain : Hadamard's method

Definition 1 *Given a smooth domain Ω_0, a (scalar) function $\Omega \mapsto F(\Omega)$ is shape differentiable at Ω_0 if the function*

$$W^{1,\infty}\left(\mathbb{R}^d, \mathbb{R}^d\right) \ni \theta \mapsto F((I+\theta)(\Omega_0))$$

is Fréchet-differentiable at 0, i.e. we have the following expansion, in the vicinity of 0 :

$$F((I+\theta)(\Omega_0)) = F(\Omega_0) + dF(\Omega_0)(\theta) + o(||\theta||)$$

For instance, the following theorem provides the shape differential of $J(\Omega) = \int_{\Gamma_N} g.u_\Omega \, ds$.

Theorem 1 *Ω being a smooth domain, if $g \in H^2(\mathbb{R}^d)$, the above functional J is shape differentiable at Ω and its shape gradient reads :*

$$dJ(\Omega)(\theta) = \int_\Gamma \left(-Ae(u_\Omega) : e(u_\Omega)\right) \theta.n \, ds$$

Differentiation with respect to the domain : Hadamard's method

- This shape gradient provides plenty many natural descent directions for functional J : *for instance*, defining θ as

$$\theta = (Ae(u_\Omega) : e(u_\Omega))\, n$$

yields, for $t > 0$ sufficiently small (*to be found numerically*) :

$$J((I + t\theta)(\Omega)) = J(\Omega) - t\int_\Gamma (\theta.n)^2\, ds + o(t) < J(\Omega)$$

- This boundary variation method is one of the many that exist in shape optimization :
  1. All the shapes obtained during the process are (at least theoretically speaking) diffeomorphic to the initial one Ω ; hence, no hole can appear, whereas it could be highly benefical ; a notion of topological gradient has been devised to study the behaviour of a shape with respect with the nucleation of a small hole near each of its points.
  2. As any gradient algorithm, this method is known to be very sensitive to initialization, and to fall into local minima of the problem ; the homogenization method is a relaxation of the minimization problem that provides a method for finding the global minimum of the relaxed problem.

The generic numerical algorithm

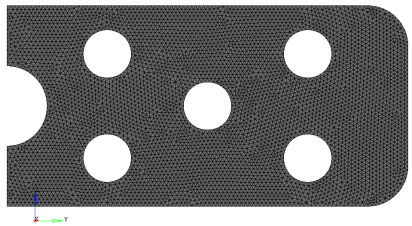
Gradient algorithm : For $n = 0, \ldots$ convergence,

1. Compute the solution u_{Ω^n} of the above elasticity system of Ω^n.
2. Compute the shape gradient $dJ(\Omega^n)$ thanks to the above formula, and infer a descent direction θ^n for the cost functional.
3. Advect the shape Ω^n according to this displacement field, so as to get Ω^{n+1}.

Problem : We need to

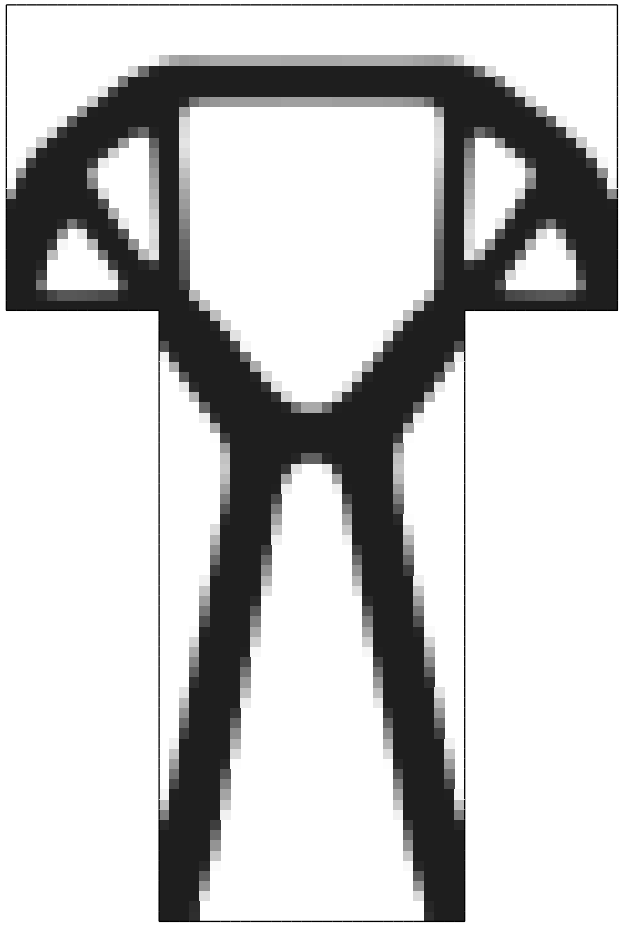
- efficiently advect the shape Ω^n at each step
- get a mesh of each shape Ω^n so as to perform the required finite element computations.

Reconciling both constraints is difficult, the bulk of approaches for moving meshes being heuristic, and at some point limited.



The level set method of Allaire-Jouve-Toader

- The shapes Ω^n are embedded in a computational box $\mathcal{D}$ equipped with a fixed mesh.
- The successive shapes Ω^n are accounted for in the level set framework, i.e. by the knowledge of a function ϕ^n defined on the whole box $\mathcal{D}$ which implicitly defines them.
- At each step n, the exact linear elasticity system on Ω^n is approximated by the Ersatz material approach : the void $\mathcal{D} \setminus \Omega^n$ is filled with a very 'soft' material, which leads to an approximate linear elasticity system, defined on $\mathcal{D}$.
- This approach is very versatile and does not require an exact mesh of the shapes at each iteration.



Shape accounted for with a level set description

The proposed method

We propose a slightly different approach which still benefits from the versatility of level set methods to account for large deformations of shapes (even topological changes), but enjoys at each step the knowledge of a mesh of the shape.

- At each step, the shape Ω^n is equipped with an unstructured mesh $\mathcal{T}^n$ when it comes to finite element computations, and is considered through an associated level set function ϕ^n, defined on a larger unstructured computational mesh when dealing with advection of the shape

$$(\Omega^n, \mathcal{T}^n) \to (\Omega^{n+1}, \mathcal{T}^{n+1}) \quad \Leftrightarrow \quad \phi^n \to \phi^{n+1}$$

- The connection between those two ways of describing shapes is made through an unstructured mesh of the computational box $\mathcal{D}$, which is allowed to evolve so that at each step n, the shape Ω^n is explicitly discretized.
  - Level set methods are performed on this unstructured mesh to account for the advection of the shapes $\phi^n \to \phi^{n+1}$.
  - Finite element computations are performed on the part on this mesh corresponding to the shape.

The proposed method

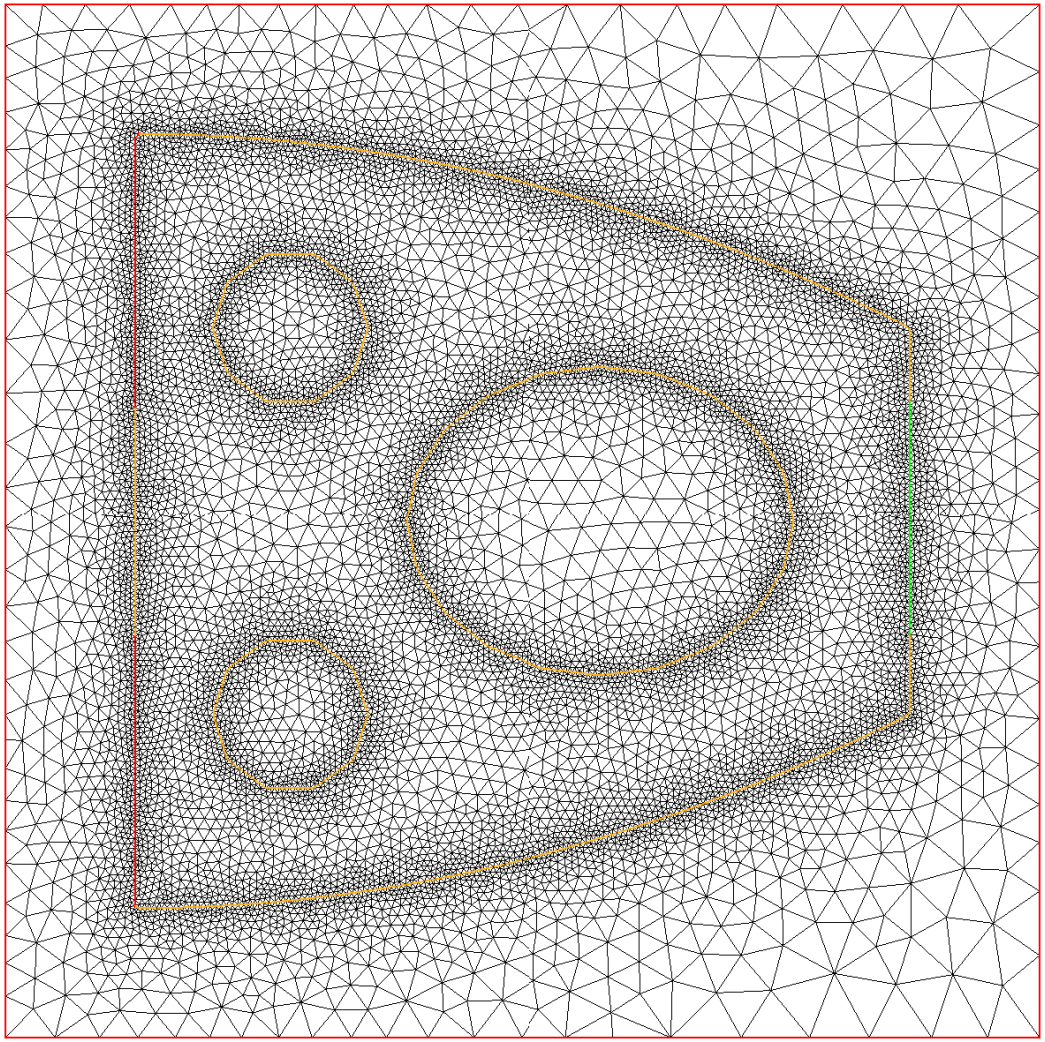


Figure 1: *Shape equipped with a mesh, conformally embedded in a mesh of the computational box.*

Outline

I. Mathematical modeling of shape optimization problems
   1. Differentiation with respect to the domain : Hadamard's method
   2. The generic numerical algorithm

II. From meshed domains to a level set description,... and conversely
   1. A few words about the level set Method
   2. Initializing level-set functions with the signed distance function
   3. Meshing the negative subdomain of a level set function : local remeshing

III. Application to shape optimization
   1. Numerical Implementation
   2. The algorithm in motion
   3. Some numerical results

A few words about the level set Method

A paradigm : *When you want to describe a surface evolution, represent it with an implicit function.*

Given a bounded domain $\Omega \subset \mathbb{R}^d$, define it with a function ϕ on the whole $\mathbb{R}^d$ such that

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

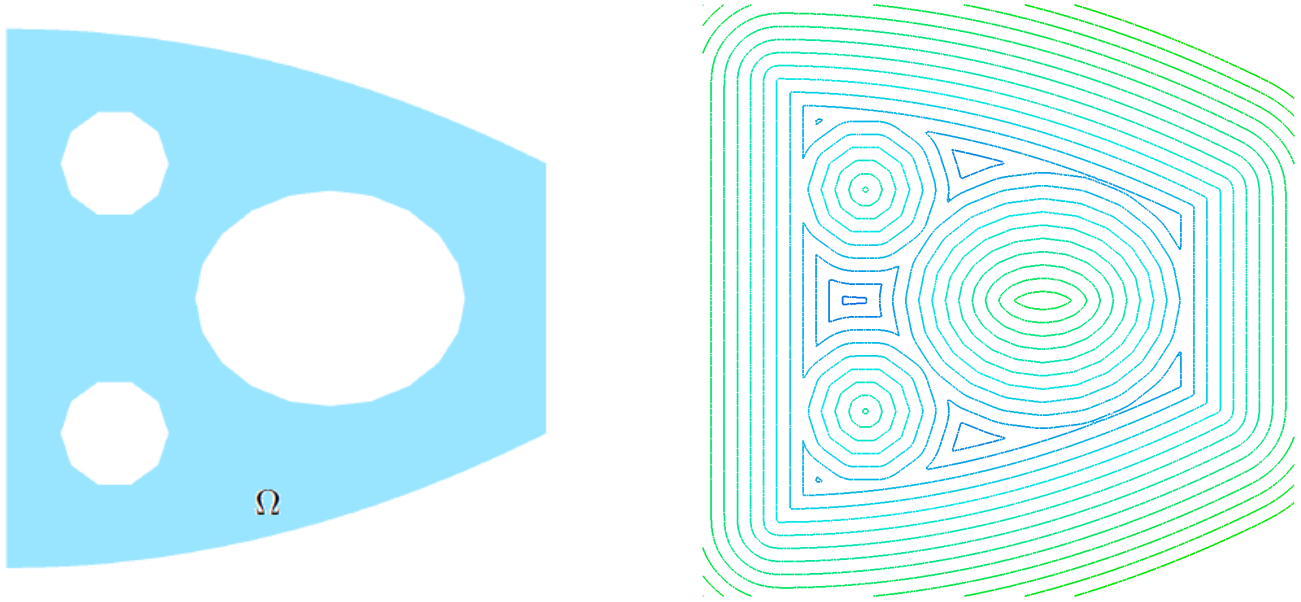


Figure 2: *A bounded domain $\Omega \subset \mathbb{R}^2$ (left), some level sets of an implicit function representing Ω (right).*

Surface evolution equations in the level set framework

Suppose that, for every time t, the domain $\Omega(t) \subset \mathbb{R}^d$ is represented by an implicit function $\phi(t,.)$ on $\mathbb{R}^d$, and is subject to an evolution defined by velocity $v(t,x) \in \mathbb{R}^d$. Then

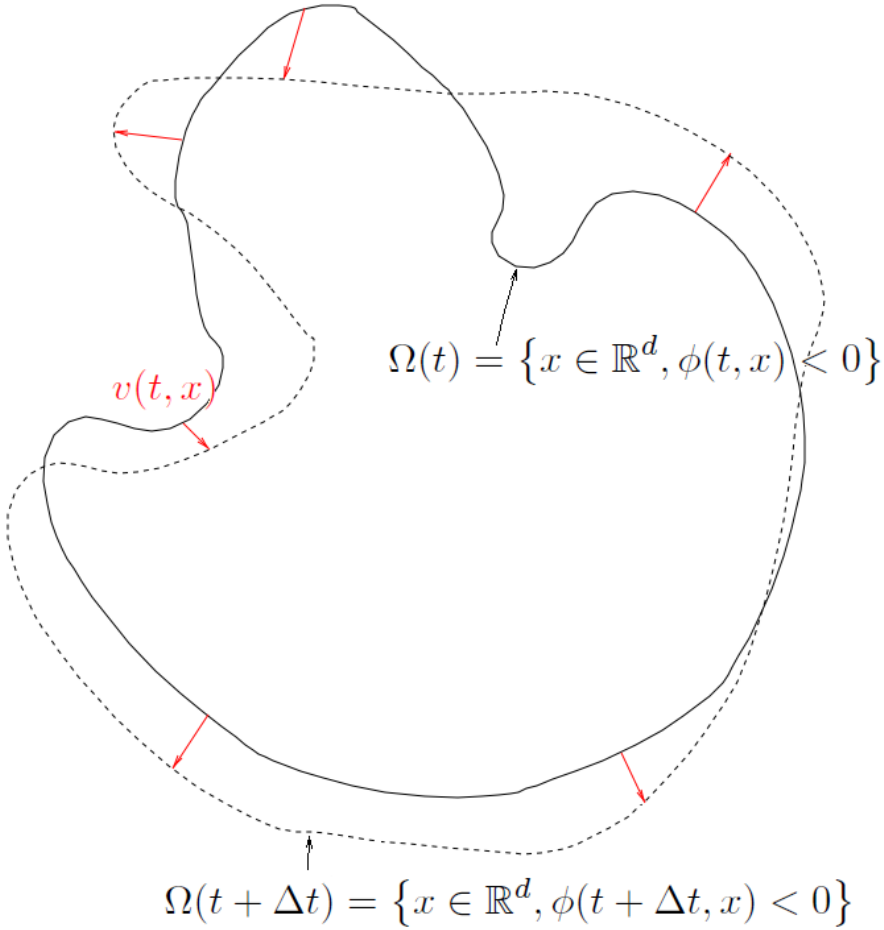
$$\boxed{\forall t,\ \forall x \in \mathbb{R}^d,\ \frac{\partial \phi}{\partial t}(t,x) + v(t,x).\nabla\phi(t,x) = 0}$$

In many applications, the velocity $v(t,x)$ is normal to the boundary $\partial\Omega(t)$:

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

Then the evolution equation rewrites as a Hamilton-Jacobi type equation

$$\boxed{\forall t,\ \forall x \in \mathbb{R}^d,\ \frac{\partial \phi}{\partial t}(t,x) + V(t,x)||\nabla\phi(t,x)|| = 0}$$



Outline

I. Mathematical modeling of shape optimization problems
   1. Differentiation with respect to the domain : Hadamard's method
   2. The generic numerical algorithm

II. From meshed domains to a level set description,... and conversely
   1. A few words about the level set Method
   2. Initializing level-set functions with the signed distance function
   3. Meshing the negative subdomain of a level set function : local remeshing

III. Application to shape optimization
   1. Numerical Implementation
   2. The algorithm in motion
   3. Some numerical results

Initializing level-set functions with the signed distance function

DEFINITION 2 *Let $\Omega \subset \mathbb{R}^d$ a bounded domain. The signed distance function to Ω is the function $\mathbb{R}^d \ni x \mapsto u_\Omega(x)$ defined by :*

$$d_\Omega(x) = \begin{cases} -d(x, \partial\Omega) & \text{if } x \in \Omega \\ 0 & \text{if } x \in \partial\Omega \\ d(x, \partial\Omega) & \text{if } x \in \overline{{}^c\Omega} \end{cases}$$ *, where $d(\cdot, \partial\Omega)$ is the usual Euclidean distance*

- The signed distance function to a domain $\Omega \subset \mathbb{R}^d$ is the 'canonical' way to initialize an associated level set function : it enables good approximations of $n(x)$, $\kappa(x)$,... and decreases numerical instabilities related to 'bad localization' of the domain, owing to its property of unitary gradient.

$$||\nabla d_\Omega(x)|| = 1, \text{ a.e. } x \in \mathbb{R}^d.$$

- We present here a PDE-based method, working in any dimension, on any simplicial mesh for computing the signed distance function to Ω that dates back to [Chopp] (see also [Sethian] or [Zhao] for different approaches).

The signed distance function as the steady state of a PDE

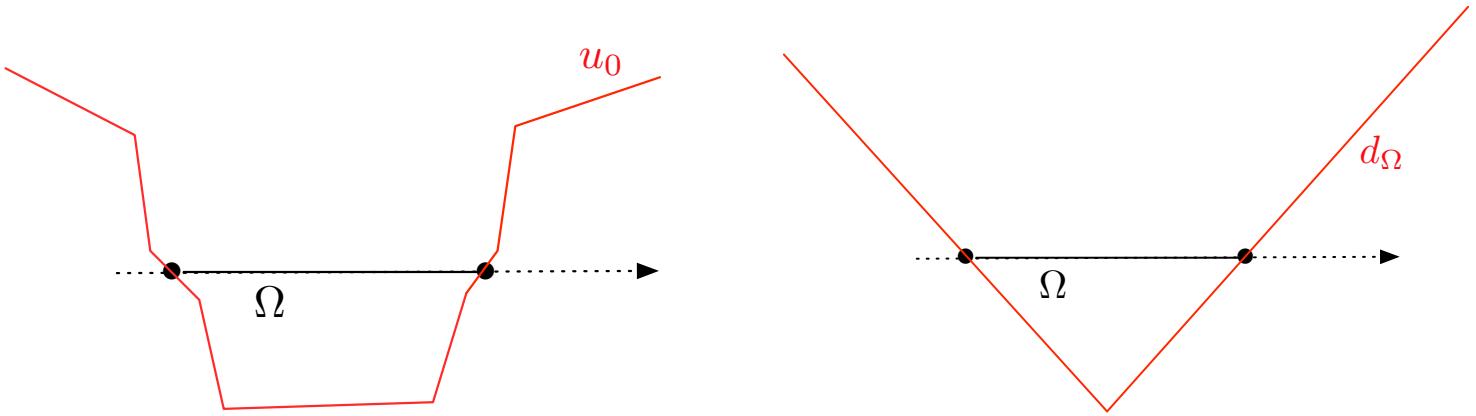


Figure 3: (left) Initializing function for $\Omega = (0,1) \subset \mathbb{R}$; (right) signed distance function to Ω.

Suppose $\Omega \subset \mathbb{R}^d$ is implicitly known as

$$\Omega = \left\{ x \in \mathbb{R}^d ; u_0(x) < 0 \right\} \ \ and \ \ \partial\Omega = \left\{ x \in \mathbb{R}^d ; u_0(x) = 0 \right\},$$

where u_0 is a function we only suppose continuous. Then the function u_Ω can be considered as the steady state of the so-called unsteady Eikonal equation

$$\begin{cases} \dfrac{\partial u}{\partial t} + sgn(u_0)(||\nabla u|| - 1) = 0 & \forall t > 0, x \in \mathbb{R}^d \\ u(t = 0, x) = u_0(x) & \forall x \in \mathbb{R}^d \end{cases} \tag{1}$$

The proposed algorithm

THEOREM 2 *Define function* u, $\forall x \in \mathbb{R}^d$, $\forall t \in \mathbb{R}_+$,

$$u(t,x) = \begin{cases} sgn(u_0(x)) \inf\limits_{||y|| \le t} (sgn(u_0(x))u_0(x+y) + t) & \text{if } t \le d(x, \partial\Omega) \\ sgn(u_0(x))d(x, \partial\Omega) & \text{if } t > d(x, \partial\Omega) \end{cases} \quad (2)$$

Let $T \in \mathbb{R}_+$. *Then* u *is the unique uniformly continuous viscosity solution of (1) such that, for all* $0 \le t \le T$, $u(t,x) = 0$ *on* $\partial\Omega$.

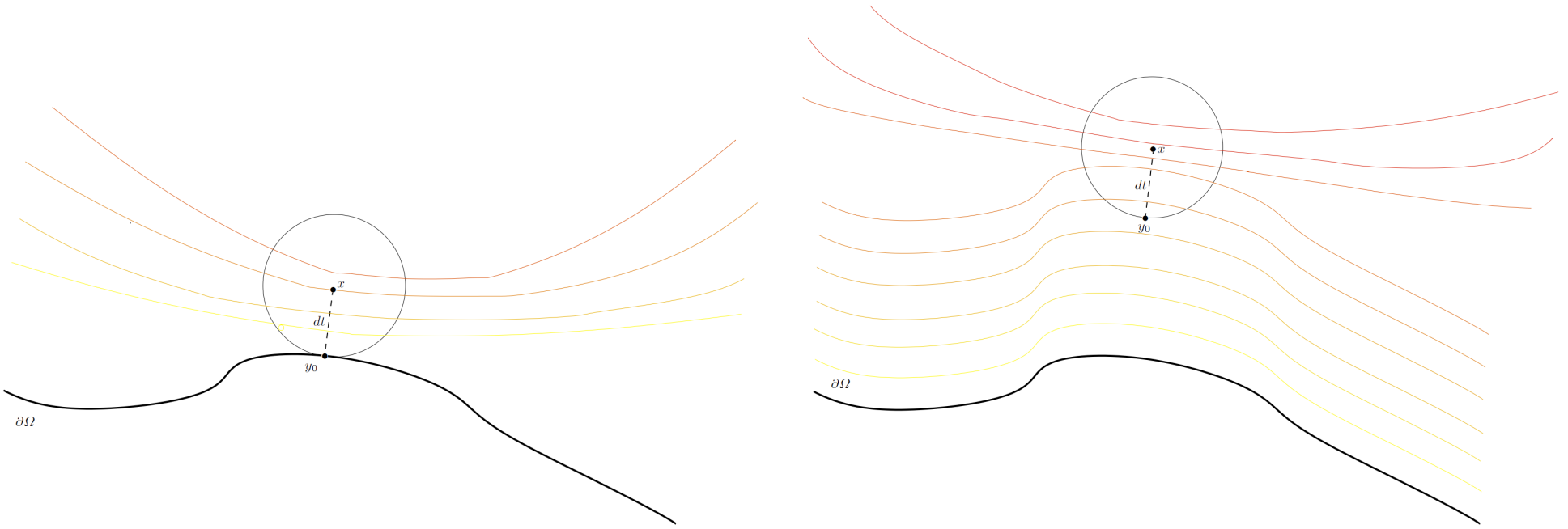
Idea : *Compute iteratively the solution* $u(t,x)$, *using the exact formula.*

Let dt a small time step, and denote $t^n = ndt$. This formula can be made iterative, denoting $u^n(x) = u(t^n, x)$, we have, for $n = 0, \ldots$

$$\forall x \in {}^c\Omega,\ u^{n+1}(x) = \inf_{||y|| \le dt} u^n(x+y) + dt$$

$$\forall x \in \Omega,\ u^{n+1}(x) = \sup_{||y|| \le dt} u^n(x+y) - dt$$

A geometric intuition of the proposed algorithm



(a) (b)

Figure 4: At a given iteration n, the proposed numerical scheme amounts to 'regularize' the value of u^n at point x from its value at point y_0 such that $u^n(y_0) = \inf_{y \in B(x,dt)} u^n(y)$ with the property of unitary gradient, (a) e.g. for a point x at distance dt from $\partial\Omega$, $u^1(x) = u_0(y_0) + dt = dt = d(x, \partial\Omega)$. (b) The property of unit gradient 'propagates' from the boundary $\partial\Omega$, near which values of u^n are 'regularized' at an early stage.

A 2d computational example

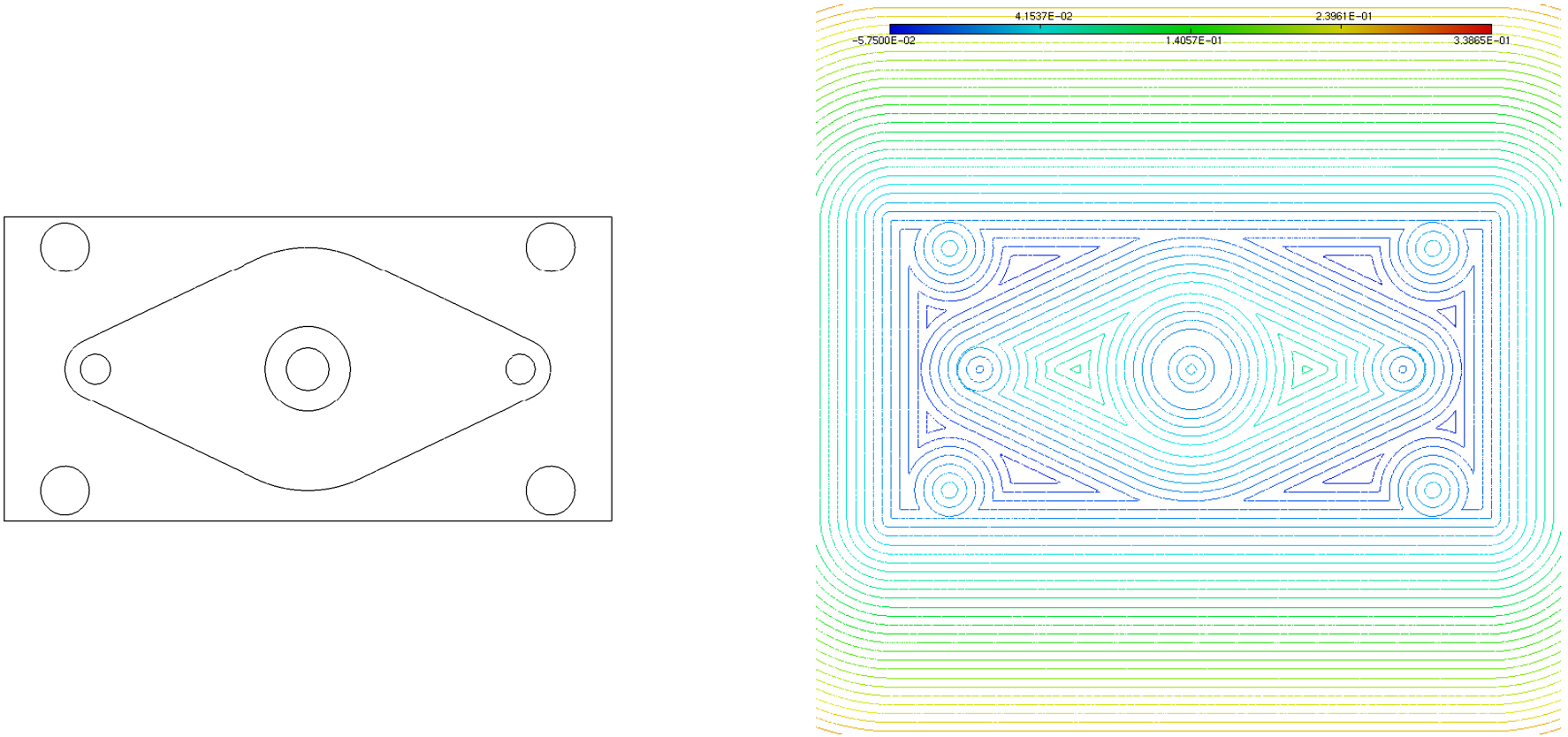


Figure 5: *Computation of the signed distance function to a discrete contour (left), on a fine background mesh (≈ 250000 vertices).*

A 3d example... the 'Venus'.

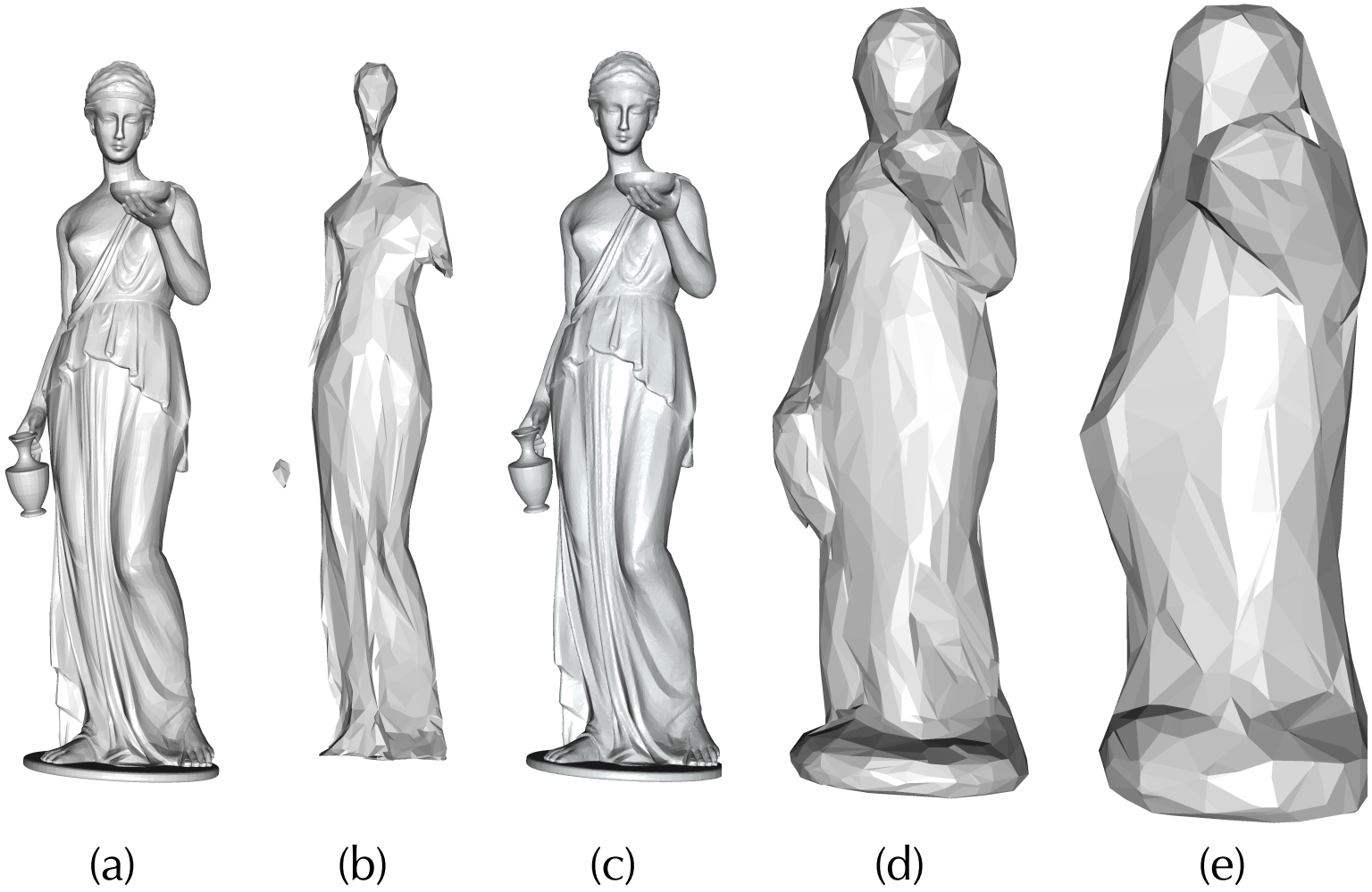


Figure 6: *Isosurfaces of the signed distance function to the 'Venus' (a) : (b) : isosurface* -0.01*, (c) : isosurface* 0*, (d) : isosurface* 0.02*, (e) : isosurface* 0.05.

Outline

I. Mathematical modeling of shape optimization problems
   1. Differentiation with respect to the domain : Hadamard's method
   2. The generic numerical algorithm

II. From meshed domains to a level set description,... and conversely
   1. A few words about the level set Method
   2. Initializing level-set functions with the signed distance function
   3. Meshing the negative subdomain of a level set function : local remeshing

III. Application to shape optimization
   1. Numerical Implementation
   2. The algorithm in motion
   3. Some numerical results

Meshing the negative subdomain of a level set function

Discretizing explicitely the 0 level set of a scalar function defined at the vertices of a simplicial mesh $\mathcal{T}$ of a computational box $\mathcal{D}$ is relatively easy, resorting to patterns.

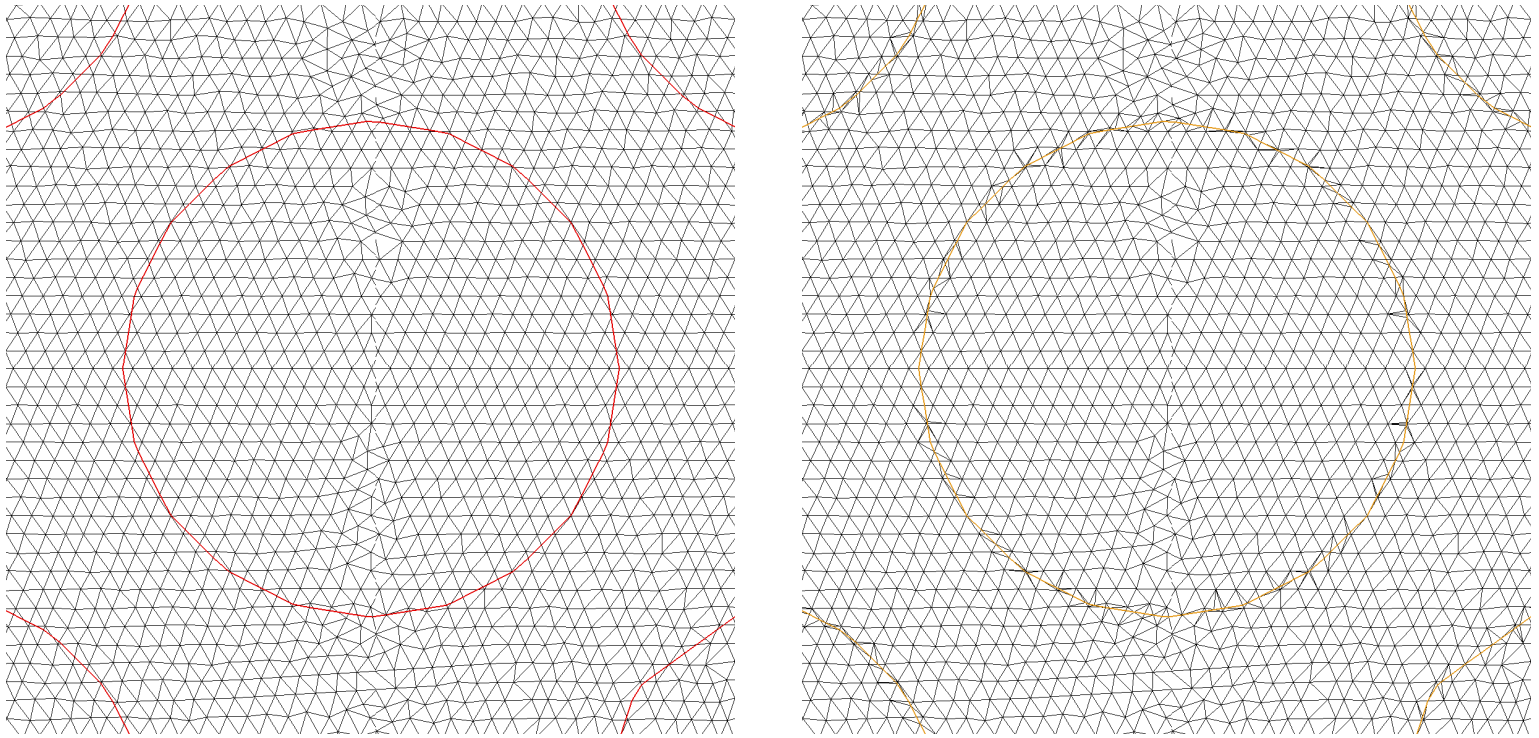


Figure 7: (left) 0 level set of a scalar function defined over a mesh ; (right) explicit discretization in the mesh.

However, doing so is bound to produce a very low-quality mesh, on which finite element computations will prove slow, inaccurate, not to say impossible.

Hence the need to improve the quality of the mesh while retaining its geometric features.

Local remeshing in $3d$

- Let $\mathcal{T}$ an initial - valid, yet potentially ill-shaped - tetrahedral mesh $\mathcal{T}$. $\mathcal{T}$ carries a triangular surface mesh $\mathcal{S}_{\mathcal{T}}$, whose elements appear as faces of tetrahedra of $\mathcal{T}$.
- $\mathcal{T}$ is intended as an approximation of an ideal domain $\Omega \subset \mathbb{R}^3$, and $\mathcal{S}_{\mathcal{T}}$ as an approximation of its boundary $\partial\Omega$.

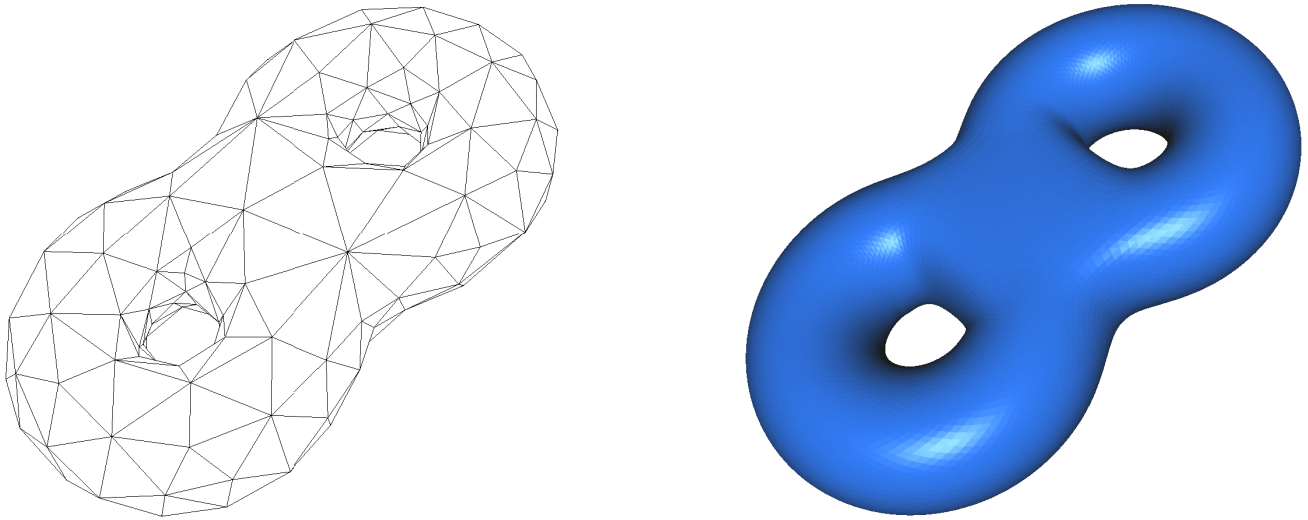


Figure 8: Poor geometric approximation (left) of a domain with smooth boundary (right)

Thanks to local mesh operations, we aim at getting a new, well-shaped mesh $\widetilde{\mathcal{T}}$, whose corresponding surface mesh $\widetilde{\mathcal{S}_{\mathcal{T}}}$ is a good approximation of $\partial\Omega$.

Local remeshing in $3d$: definition of an ideal domain

- In realistic cases, the ideal underlying domain Ω associated to $\mathcal{T}$ is unknown.
- However, from the sole data of $\mathcal{T}$ (and $\mathcal{S}_{\mathcal{T}}$), one can reconstruct approximations of geometric features of Ω : sharp angles, normal vectors at regular surface points,...
- These geometric data allow to define rules for the generation of a local parametrization of $\partial\Omega$, around a considered surface triangle $T \in \mathcal{S}_{\mathcal{T}}$, for instance as a Bézier surface.

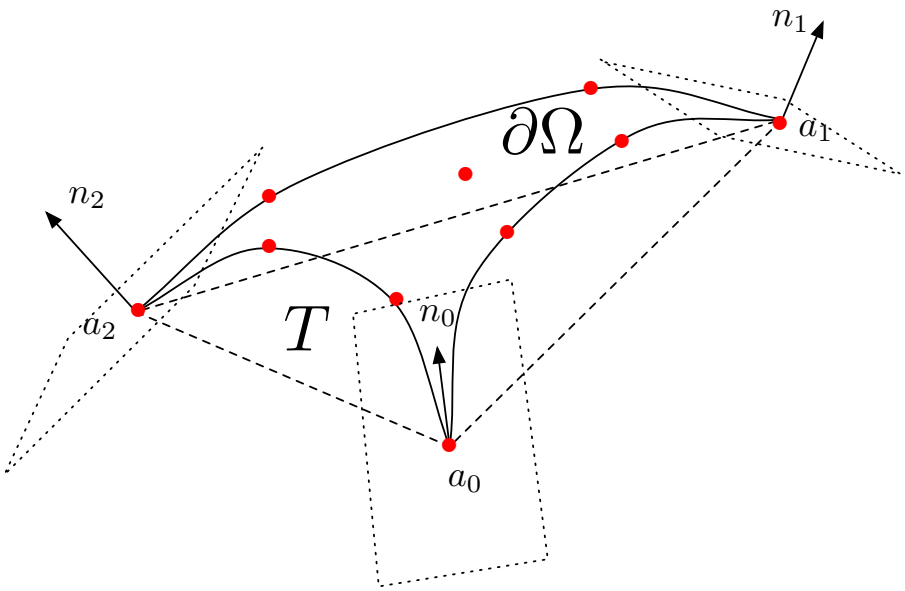


Figure 9: Generation of a cubic Bézier polynomial parametrization for the piece of $\partial\Omega$ associated to triangle T, from the approximated geometrical features (normal vectors at nodes).

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 $T \in \mathcal{S}_\mathcal{T}$, the midpoint m is inserted as the midpoint on the local piece of $\partial\Omega$ computed from T. Else, it is merely inserted as the midpoint of p and q.
- An edge may be 'too long' because it is too long when compared to the prescribed size, or because it causes a bad geometric approximation of $\partial\Omega$,...

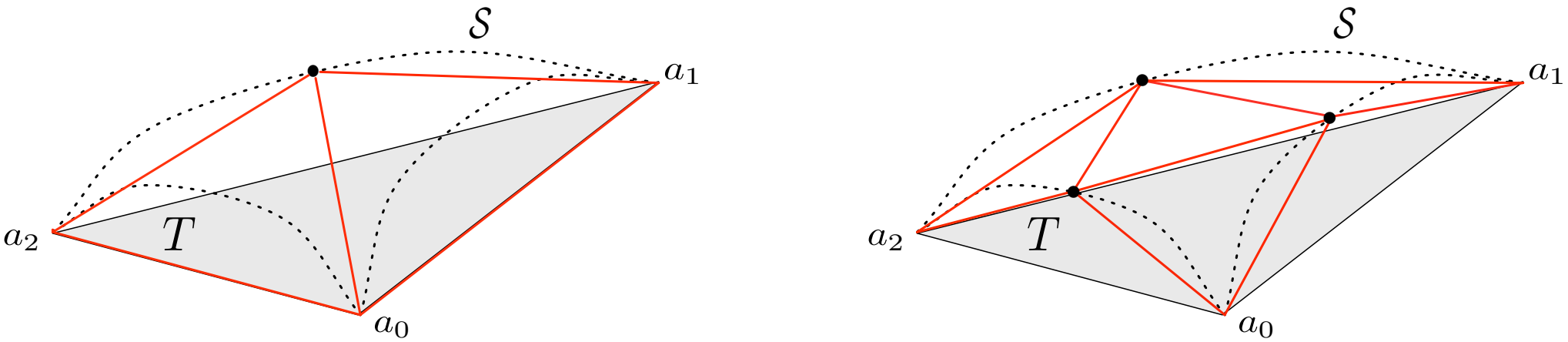


Figure 10: Splitting of one (left) or three (right) edges of triangle T, positioning the three new points on the ideal surface $\mathcal{S}$ (dotted).

Local mesh operators : edge collapse

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

- This operation may deteriorate the geometric approximation of $\partial\Omega$, and even invalidate some tetrahedra : some checks have to be performed to ensure the validity of the resulting configuration.
- An edge may be 'too short' because it is too long when compared to the prescribed size, or because it proves unnecessary to a nice geometric approximation of $\partial\Omega$,...

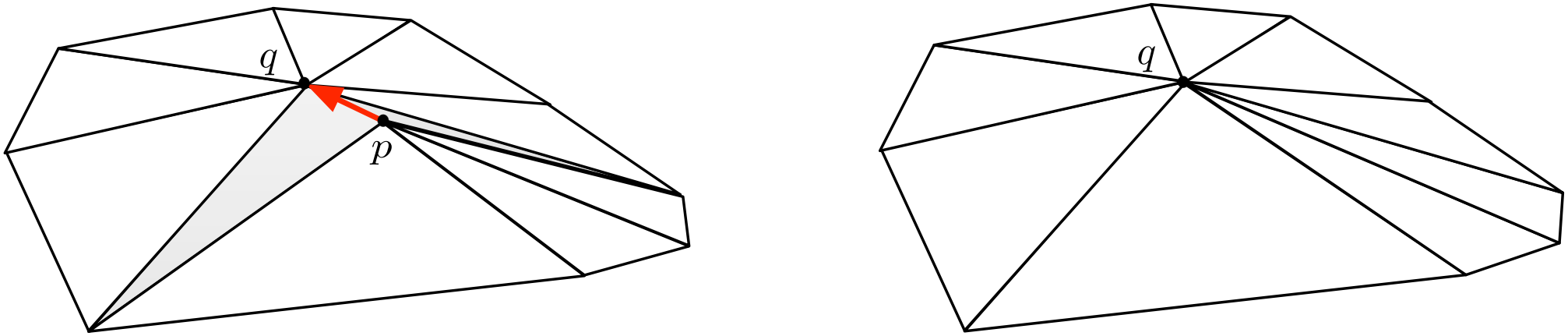


Figure 11: Collapse of point p over q.

Local mesh operators : edge swap, node relocation,...

For the sake of enhancement of the global quality of the mesh (or the geometrical approximation of $\partial\Omega$), some connectivities can be swapped, and some nodes can be slightly moved.

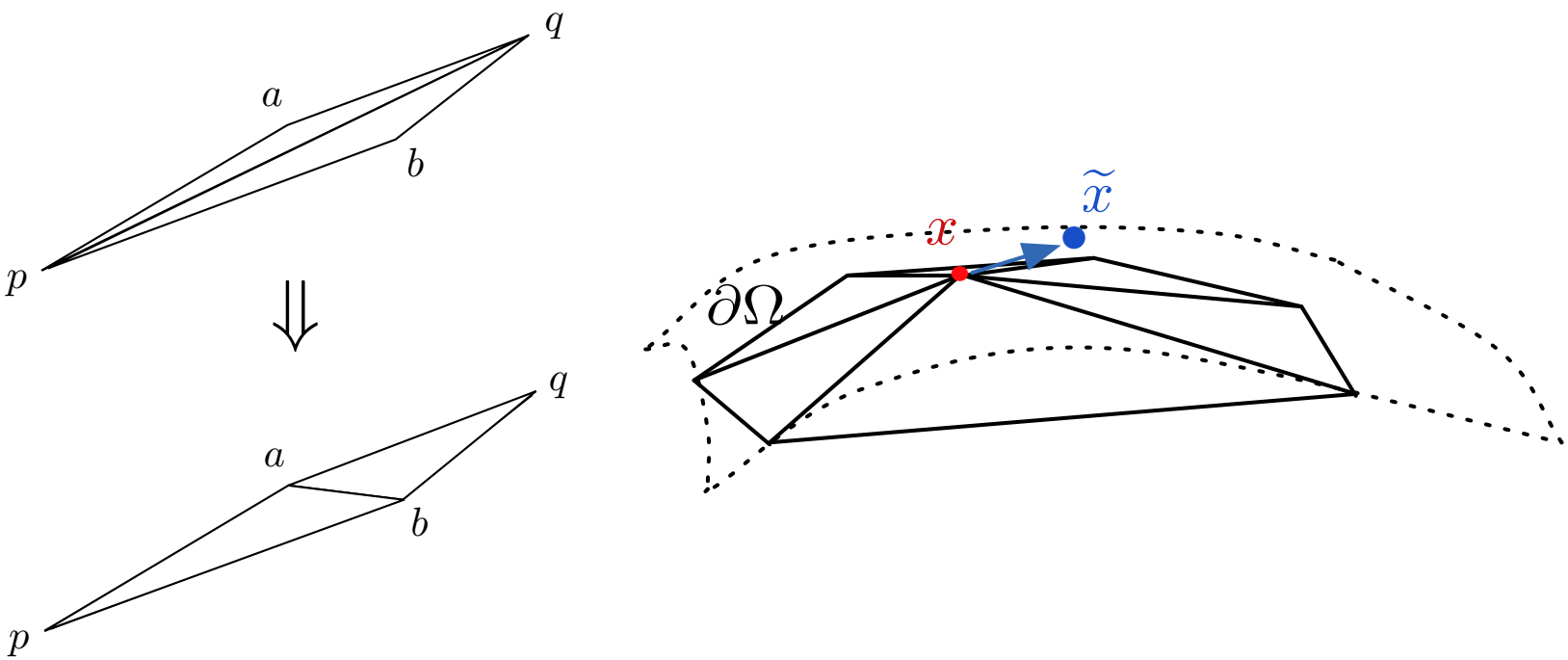


Figure 12: (left) $2d$ swap of edge pq, creating edge ab ; (right) relocation of node x to $\widetilde{x}$, along the surface.

Local remeshing in $3d$: numerical examples

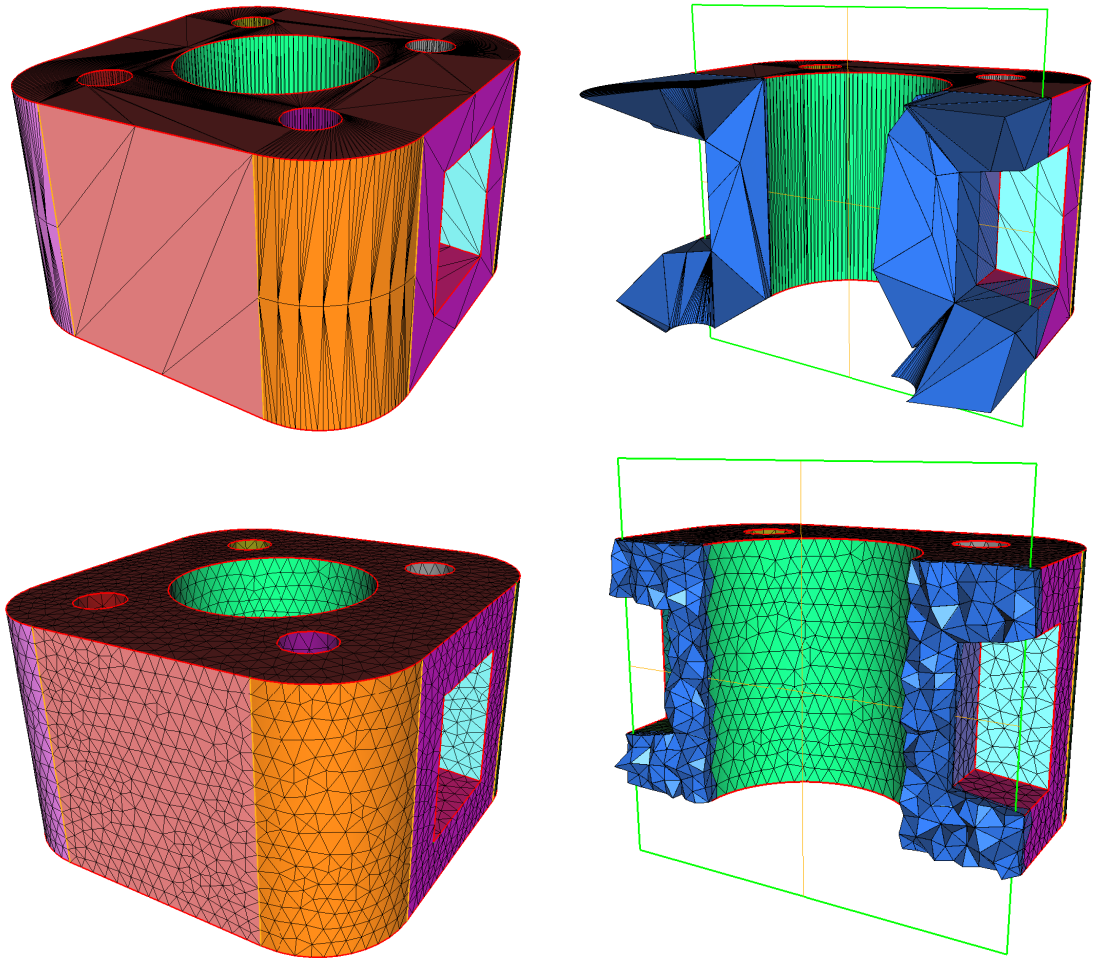


Figure 13: *Mechanical part before (left) and after (right) remeshing.*

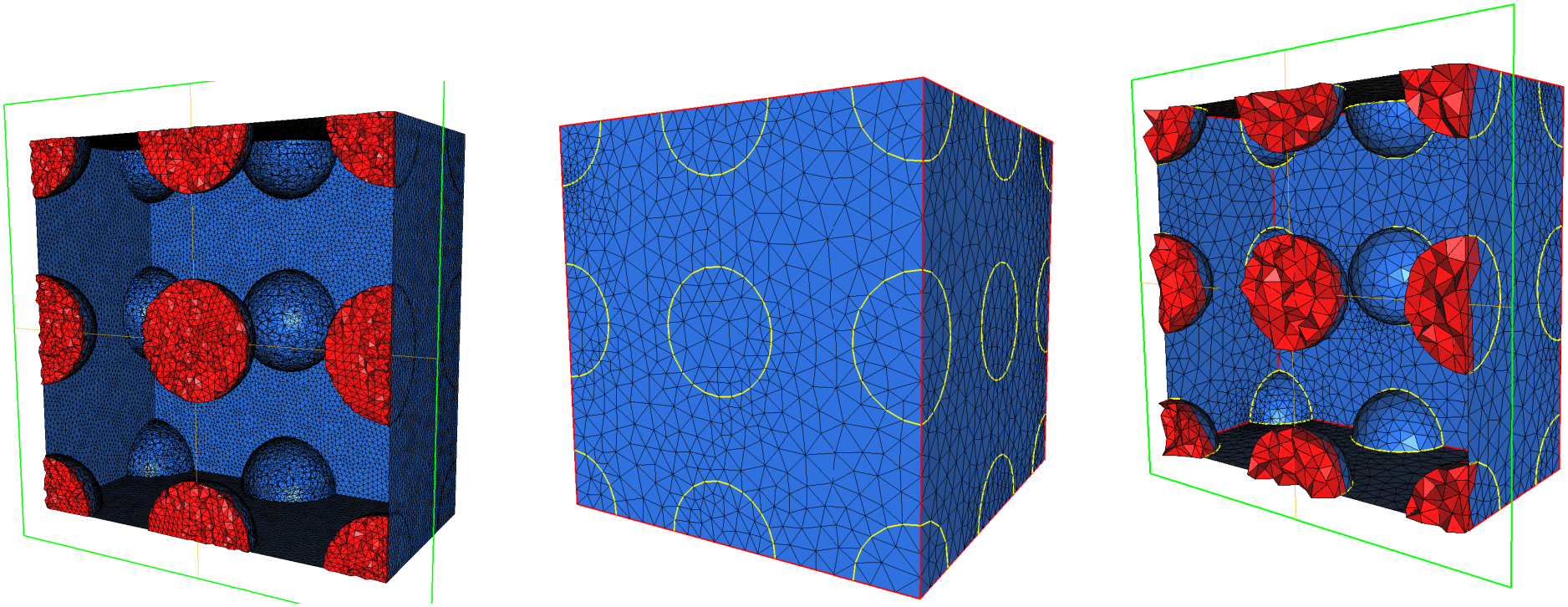


Figure 14: *(left) Ill-shaped discretization of an implicit function in a cube, (centre-right) result after local remeshing.*

Outline

I. Mathematical modeling of shape optimization problems
   1. Differentiation with respect to the domain : Hadamard's method
   2. The generic numerical algorithm

II. From meshed domains to a level set description,... and conversely
   1. A few words about the level set Method
   2. Initializing level-set functions with the signed distance function
   3. Meshing the negative subdomain of a level set function : local remeshing

III. Application to shape optimization
   1. Numerical Implementation
   2. The algorithm in motion
   3. Some numerical results

Numerical implementation

- At each iteration, the shape Ω^n is endowed with an unstructured mesh $\mathcal{T}^n$ of a larger, fixed, bounding box $\mathcal{D}$, in which a mesh of Ω^n explicitly appears as a submesh.

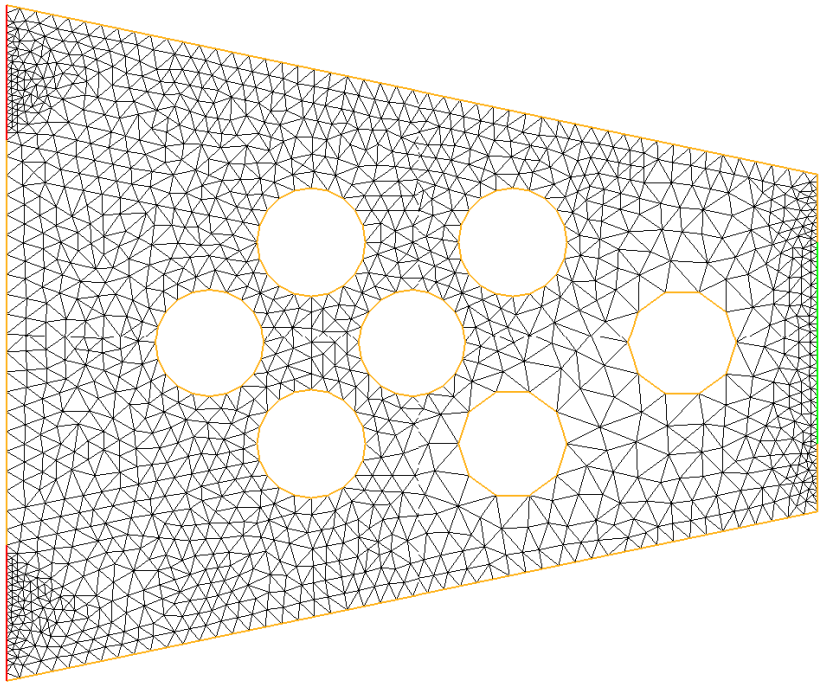
- When dealing with finite element computations on Ω^n, the part of $\mathcal{T}^n$, exterior to Ω^n is simply 'forgotten'.

- When dealing with the advection step, a level set function ϕ^n is generated on the whole mesh $\mathcal{T}^n$, and the level set advection equation is solved on this mesh, to get ϕ^{n+1}.

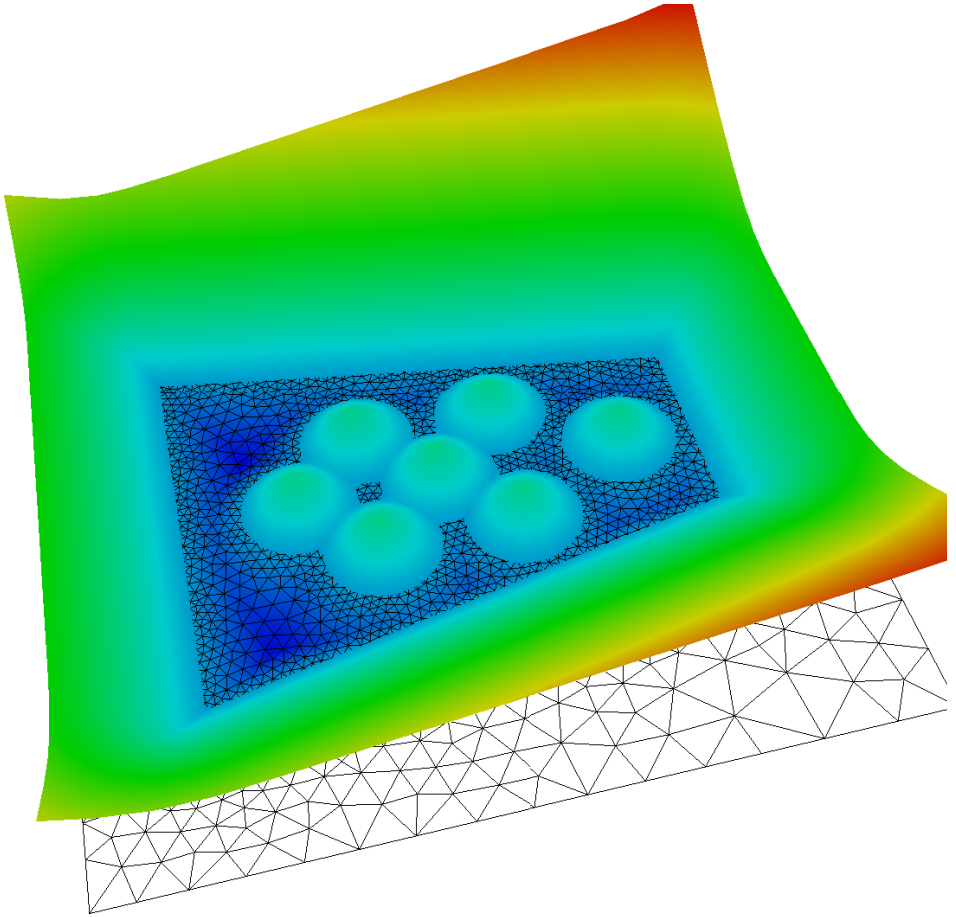
- From the knowledge of ϕ^{n+1}, a new unstructured mesh $\mathcal{T}^{n+1}$, in which the new shape Ω^{n+1} explicitly appears, is recovered, 'inserting the points of the new shape in the previous mesh.'

The algorithm in motion...

Start with an initial shape Ω_0, and generate its signed distance function over a computational domain $\mathcal{D}$, equipped with an unstructured mesh.



(a) The initial shape

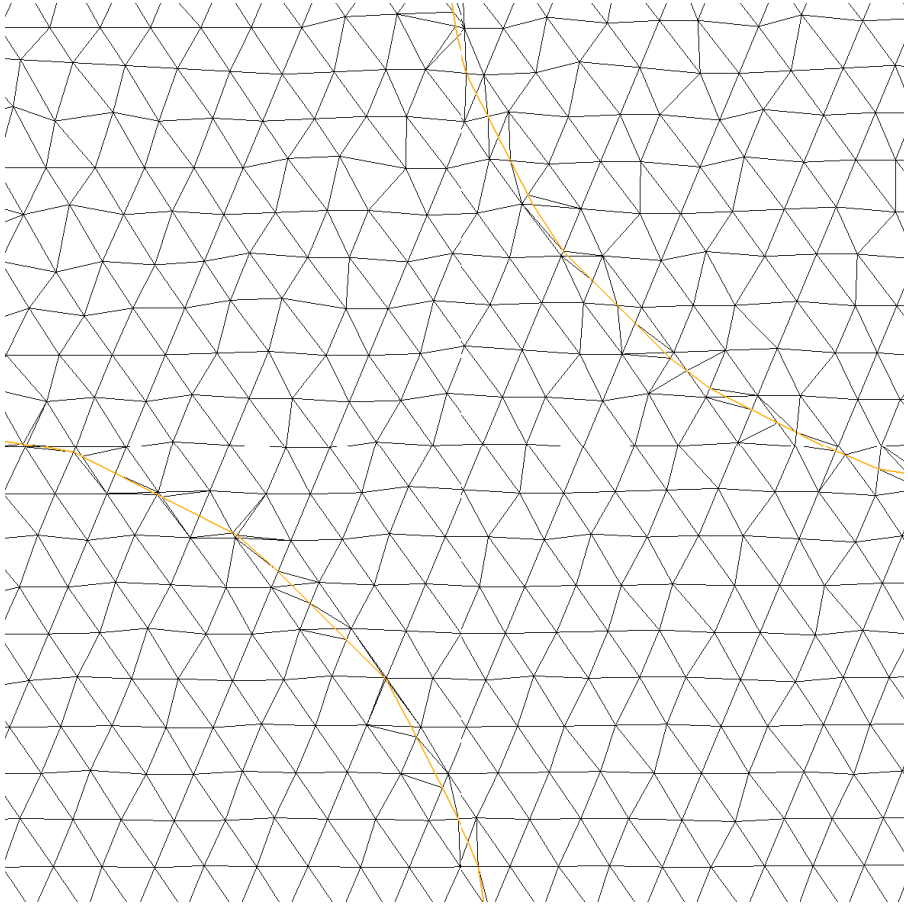


(b) Isolines of its signed distance function

The algorithm in motion...

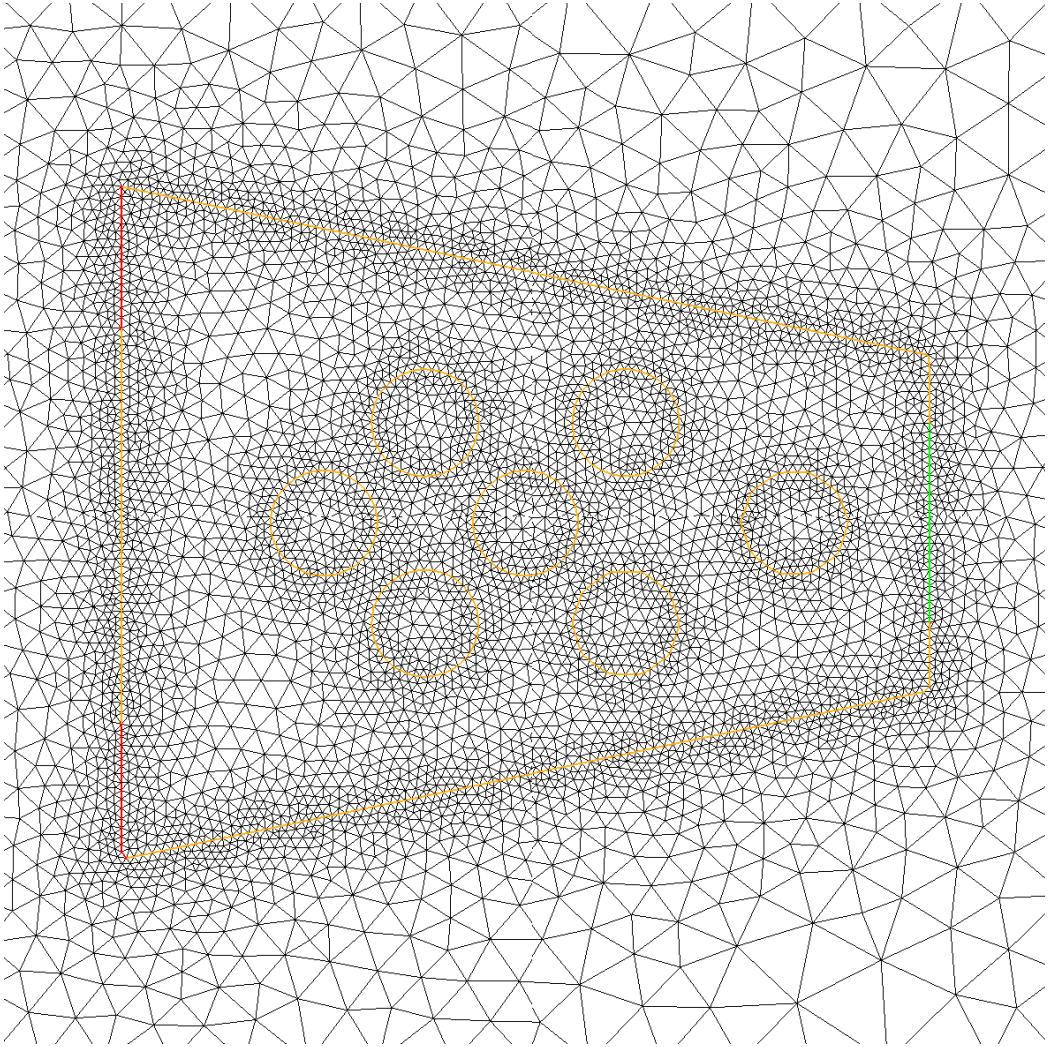
To compute the velocity field through which the shape is to be evolved, a mesh of the volume enclosed by the 0-level set of the distance ($\approx \Omega_0$) is required ; to this end, this 0-isoline is explicitely discretized in the unstructured mesh of $\mathcal{D}$.

Unfortunately, roughly "breaking" this line into the mesh generally yields a very bad-shaped mesh

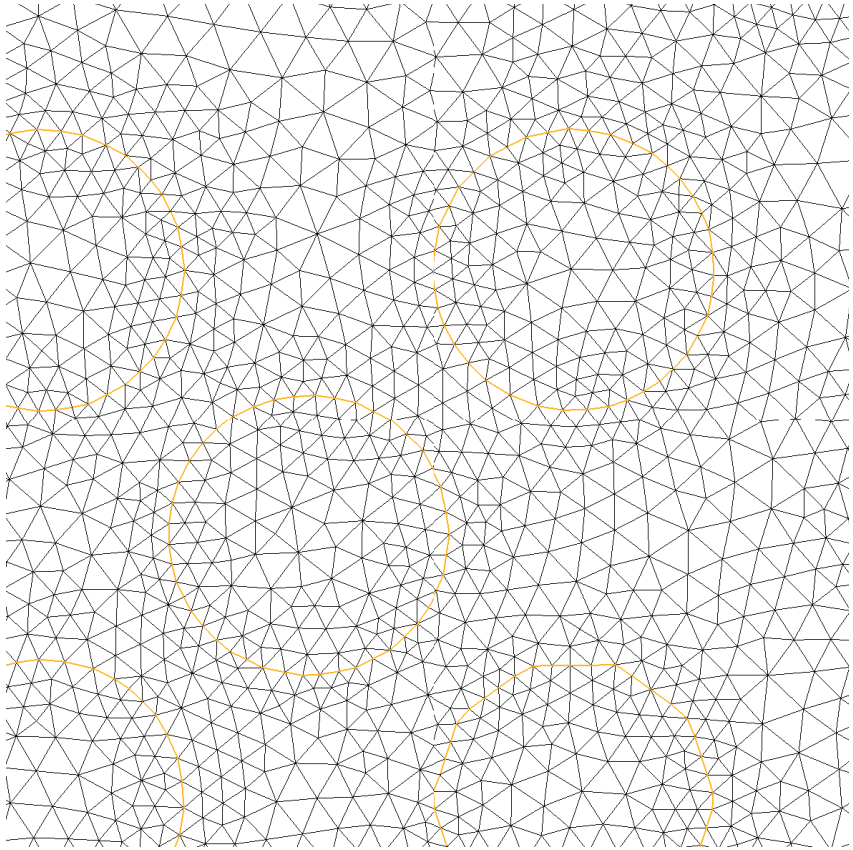


The algorithm in motion...

The mesh modification step is then performed, so as to enhance the overall quality of the mesh according to the geometry of the shape.



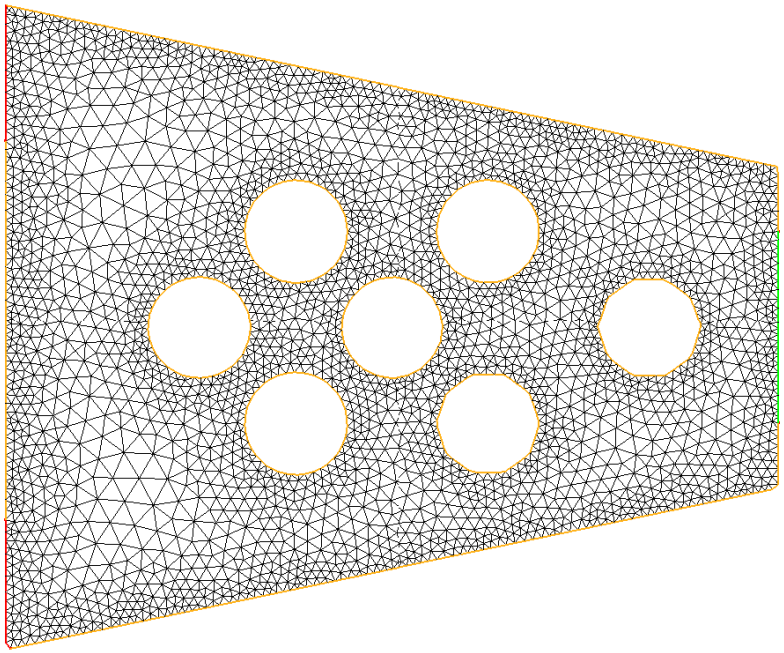
(a) Modified mesh of the computational domain



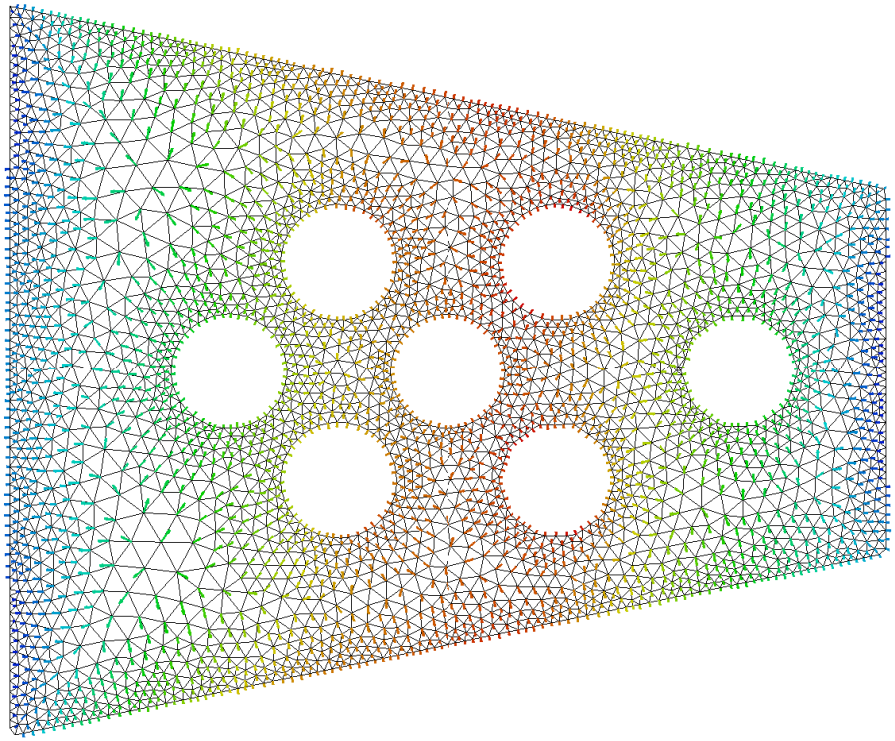
(b) Detail on the mesh

The algorithm in motion...

"Forget" the exterior of the shape, and perform the computation of the shape gradient on the shape.



(a) The "interior mesh"



(b) Computation of the gradient

The algorithm in motion...

"Remember" the computational mesh, and advect the shape as the 0-level set of its signed distance function, computed on the whole computational mesh.

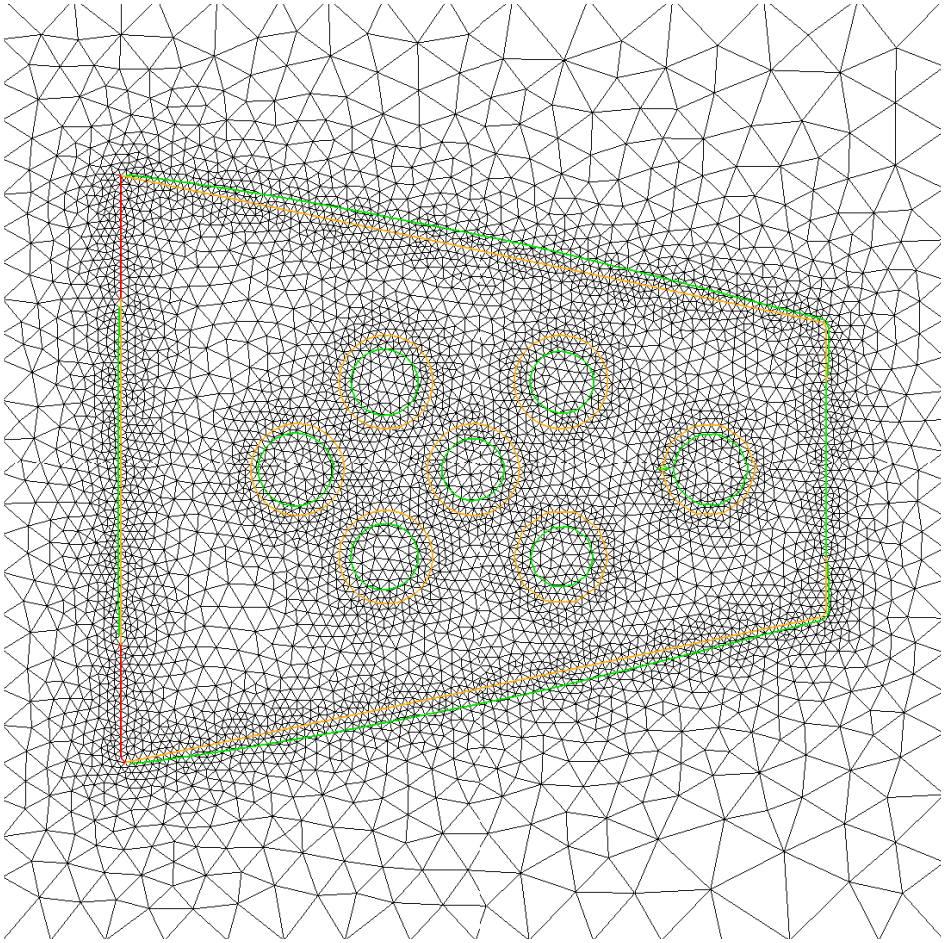
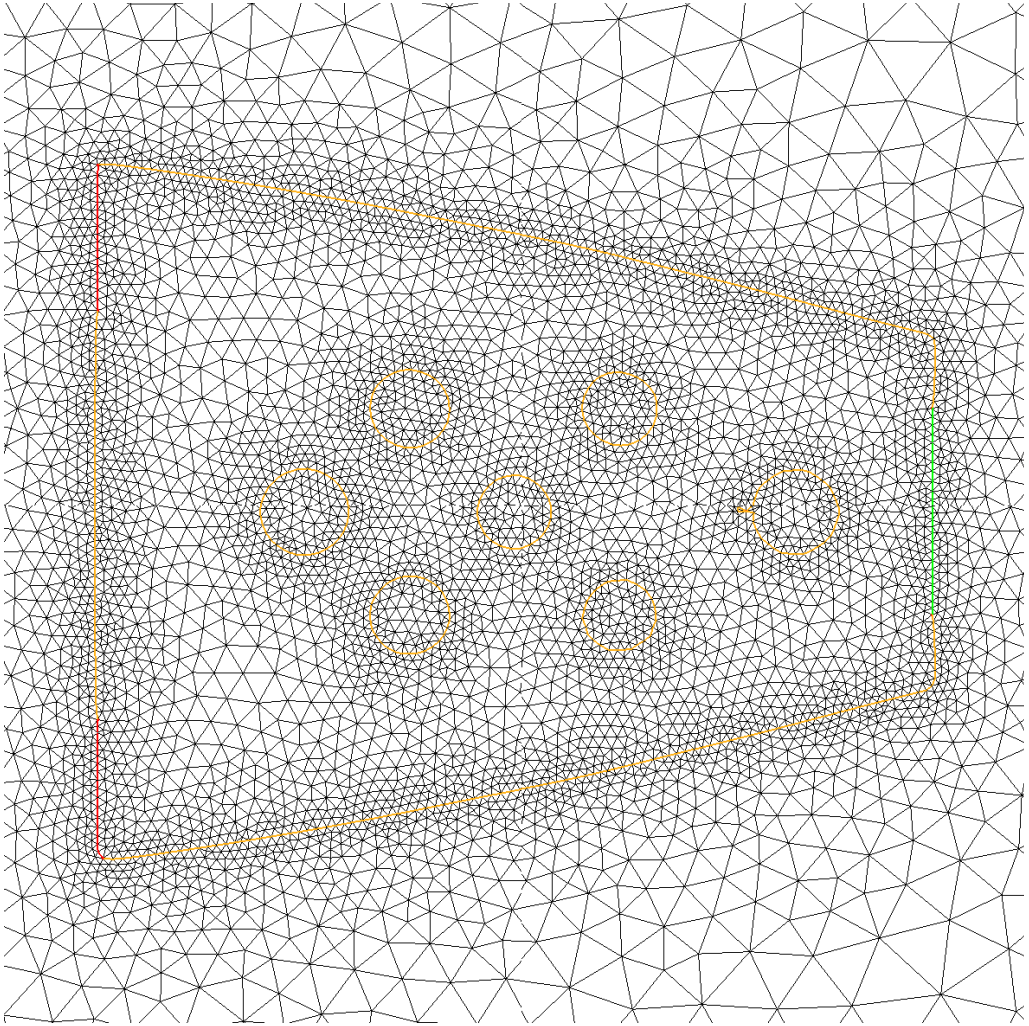


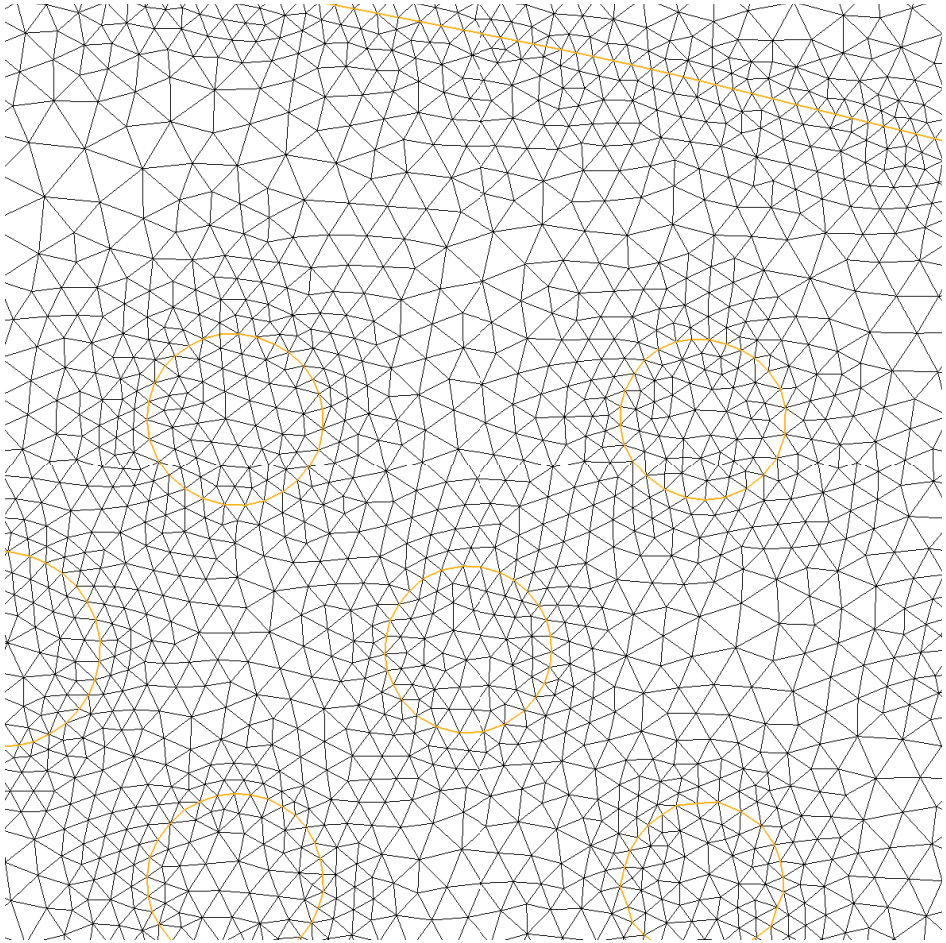
Figure 15: *The previous shape, discretized in the mesh (in yellow), and the "new", advected 0-level set (in green).*

The algorithm in motion...

Go on as before, until convergence (discretize the 0-level set in the computational mesh, clean the mesh,...).

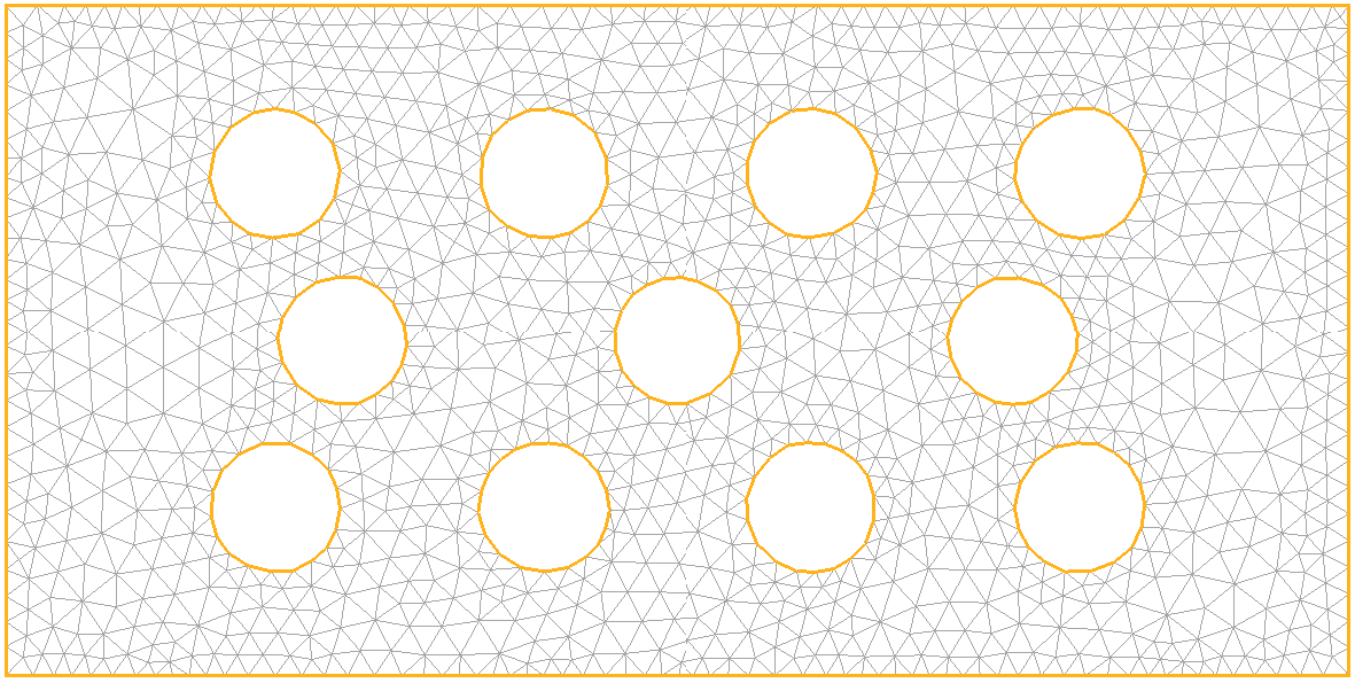


(a) The "interior mesh"

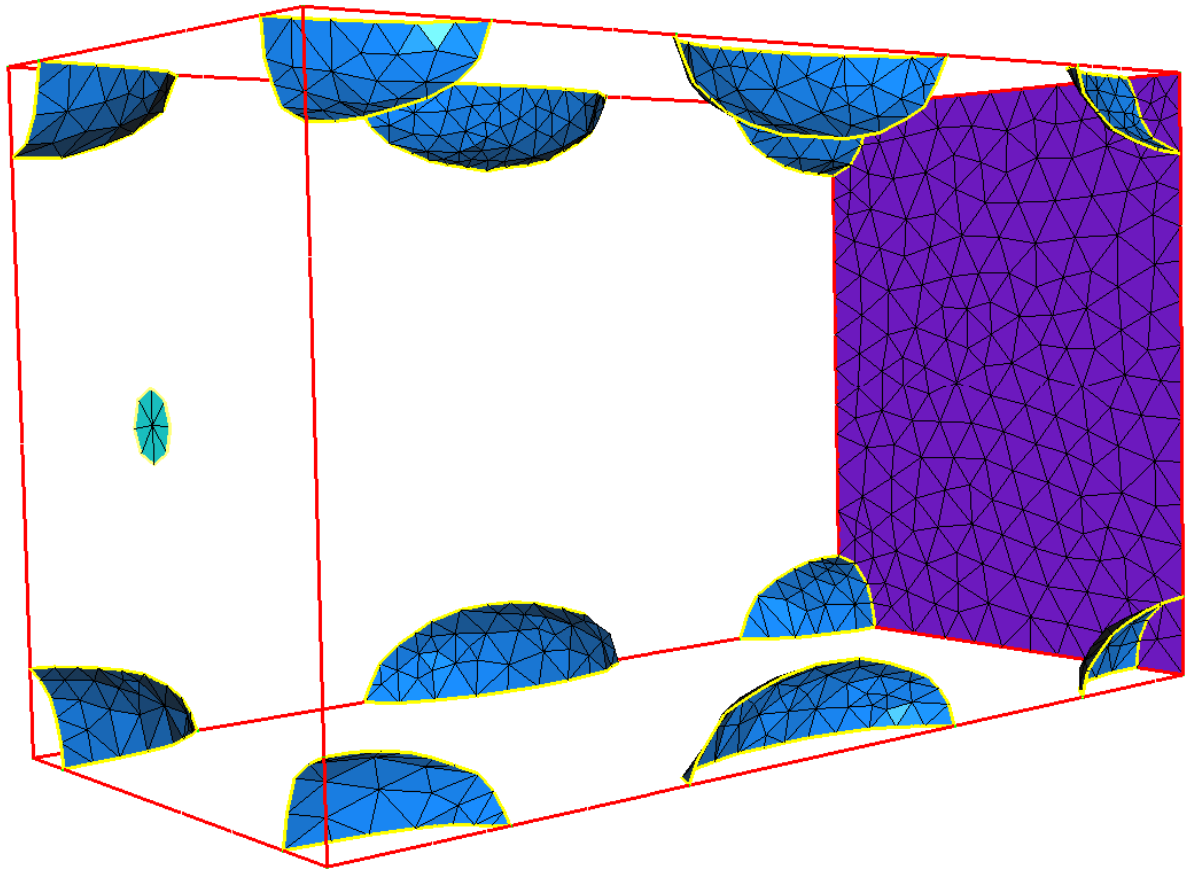


(b) Computation of the gradient

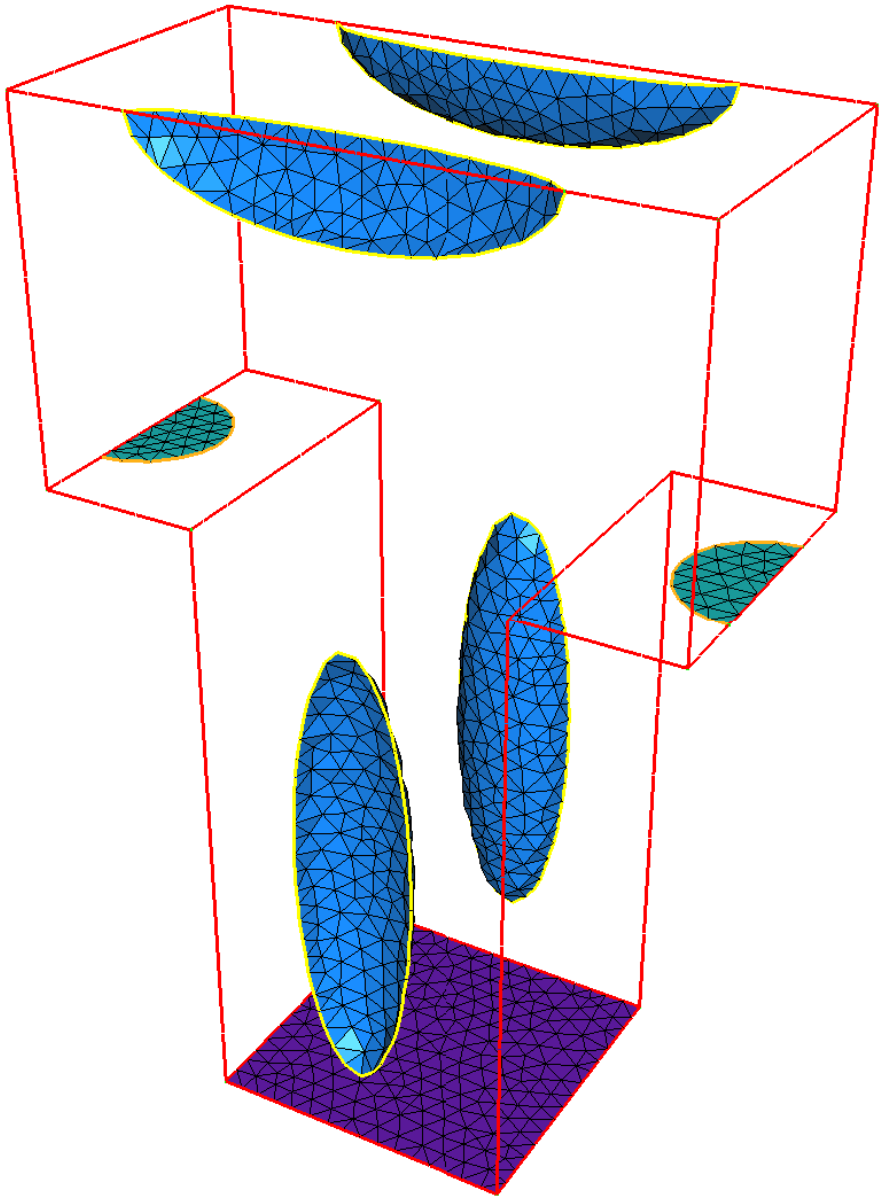
Some numerical results



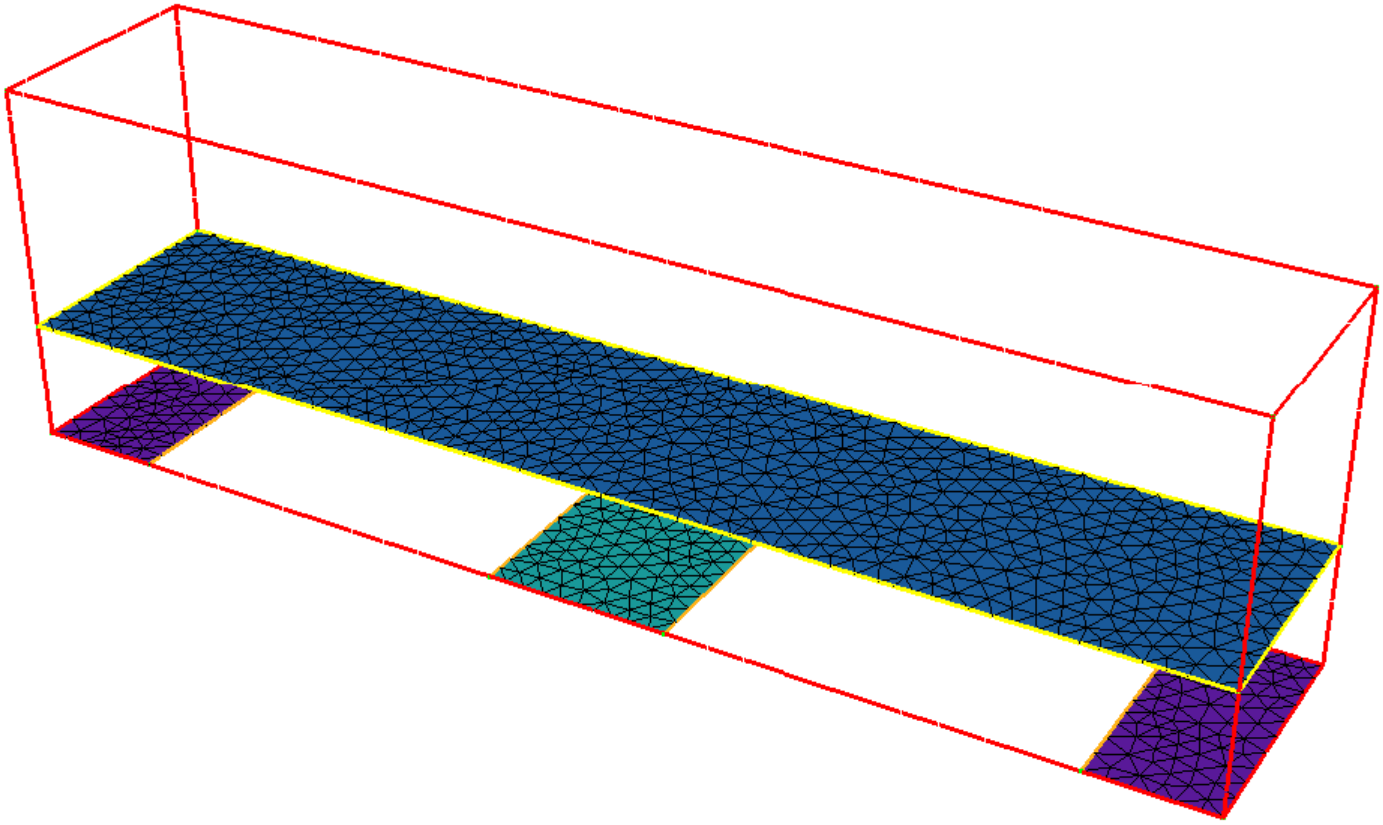
Some numerical results



Some numerical results



Some numerical results



Thank you !