A level-set based mesh evolution method for shape optimization

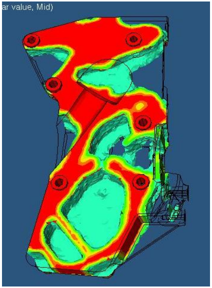
Grégoire Allaire[1], Charles Dapogny[2], Florian Feppon[1], Pascal Frey[3]

[1] CMAP, UMR 7641 École Polytechnique, Palaiseau, France
[2] Laboratoire Jean Kuntzmann, Université Joseph Fourier, Grenoble, France
[3] Laboratoire J.-L. Lions, UPMC, Paris, France

1st June, 2018

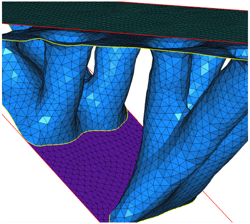
Shape optimization and industrial applications

- The increase in the cost of raw materials urges to optimize the shape of mechanical parts from the early stages of design.



- The numerical resolution of shape optimization problems is plagued by a major difficulty:
  - The evaluation of the objective criterion and its derivative involve mechanical computations, using the Finite Element method on a mesh of the shape.
  - The shape is (dramatically!) changing in the course of the iterative optimization process

    $\Rightarrow$ Need to update this computational mesh.



- This difficulty arises in many inverse problems: shape detection or reconstruction, image segmentation, etc.

Contents

A model problem in linear elasticity

A shape is a bounded domain $\Omega \subset \mathbb{R}^d$, which is

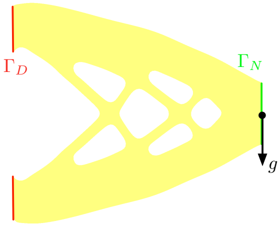
- fixed on a part Γ_D of its boundary,
- submitted to surface loads g, applied on $\Gamma_N \subset \partial\Omega$, $\Gamma_D \cap \Gamma_N = \emptyset$.

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

$$\left\{ \begin{array}{rcll} -\text{div}(Ae(u_\Omega)) & = & 0 & \text{in } \Omega \\ u_\Omega & = & 0 & \text{on } \Gamma_D \\ Ae(u_\Omega)n & = & g & \text{on } \Gamma_N \\ Ae(u_\Omega)n & = & 0 & \text{on } \Gamma \end{array} \right. ,$$

where $e(u) = \frac{1}{2}(\nabla u^T + \nabla u)$ is the strain tensor, and A is the Hooke's law of the material:

$$\forall e \in \mathcal{S}_d(\mathbb{R}), \;\; Ae = 2\mu e + \lambda \text{tr}(e) I.$$



A 'Cantilever'

The deformed cantilever

A model problem in linear elasticity

Goal: Starting from an initial structure Ω_0, find a new one Ω that minimizes a certain functional of the domain $J(\Omega)$.

Examples:

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

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

- A least-square error between u_Ω and a target displacement $u_0 \in H^1(\Omega)^d$ (useful when designing micro-mechanisms):

$$D(\Omega) = \left(\int_\Omega k(x) |u_\Omega - u_0|^\alpha dx \right)^{\frac{1}{\alpha}},$$

  where α is a fixed parameter, and $k(x)$ is a weight factor.

A volume constraint may be enforced with a fixed penalty parameter ℓ:

$$\text{Minimize } J(\Omega) := C(\Omega) + \ell \, \mathrm{Vol}(\Omega), \text{ or } D(\Omega) + \ell \, \mathrm{Vol}(\Omega).$$

1. Mathematical modeling of shape optimization problems
   - shape optimization of linear elastic structures
   - Differentiation with respect to the domain: Hadamard's method
   - Numerical implementation of shape optimization algorithms
   - The proposed method

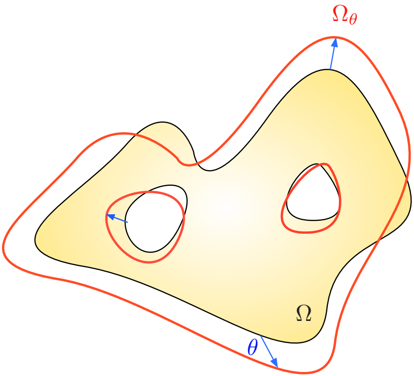
2. From meshed domains to a level set description,... and conversely
   - Initializing level set functions with the signed distance function
   - Meshing the negative subdomain of a level set function: local remeshing

3. Application to shape optimization
   - Numerical implementation
   - The algorithm in motion
   - Numerical results

Hadamard's boundary variation method describes variations of a reference, Lipschitz domain Ω of the form:

$$\Omega_\theta := (I + \theta)(\Omega),$$

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



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.

Definition 1.

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

$$W^{1,\infty}(\mathbb{R}^d, \mathbb{R}^d) \ni \theta \mapsto F(\Omega_\theta)$$

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

$$F(\Omega_\theta) = F(\Omega) + F'(\Omega)(\theta) + o\left(||\theta||_{W^{1,\infty}(\mathbb{R}^d,\mathbb{R}^d)} \right).$$

Differentiation with respect to the domain: Hadamard's method (III)

- Techniques from optimal control make it possible to compute shape gradients; in the case of 'many' shape functionals $J(\Omega)$, the shape derivative has the structure:

$$J'(\Omega)(\theta) = \int_\Gamma v_\Omega \, \theta \cdot n \, ds,$$

where v_Ω is a scalar field depending on u_Ω, and possibly on an adjoint state p_Ω.

- This shape gradient provides a natural descent direction for $J(\Omega)$: *for instance*, defining θ as

$$\theta = -v_\Omega n$$

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

$$J(\Omega_{t\theta}) = J(\Omega) - t \int_\Gamma v_\Omega^2 ds + o(t) < J(\Omega)$$

Example: If $J(\Omega) = C(\Omega) = \int_{\Gamma_N} g \cdot u_\Omega \, ds$ is the compliance, $v_\Omega = -Ae(u_\Omega) : e(u_\Omega)$.

1. Mathematical modeling of shape optimization problems
   - shape optimization of linear elastic structures
   - Differentiation with respect to the domain: Hadamard's method
   - Numerical implementation of shape optimization algorithms
   - The proposed method
2. From meshed domains to a level set description,... and conversely
   - Initializing level set functions with the signed distance function
   - Meshing the negative subdomain of a level set function: local remeshing
3. Application to shape optimization
   - Numerical implementation
   - The algorithm in motion
   - Numerical results

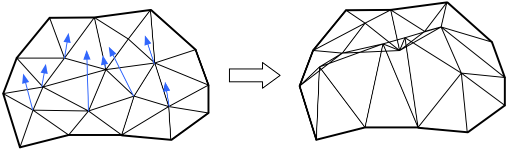
The generic numerical algorithm

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

1. Compute the solution u_{Ω^n} (and p_{Ω^n}) of the elasticity system on Ω^n.
2. Compute the shape gradient $J'(\Omega^n)$ thanks to the previous formula, and infer a descent direction θ^n for the cost functional.
3. Advect the shape Ω^n according to θ^n, so as to get $\Omega^{n+1} := (I + \theta^n)(\Omega^n)$.

Problem: We need to

- efficiently advect the shape Ω^n at each step
- get a mesh of each shape Ω^n, for finite element computations.



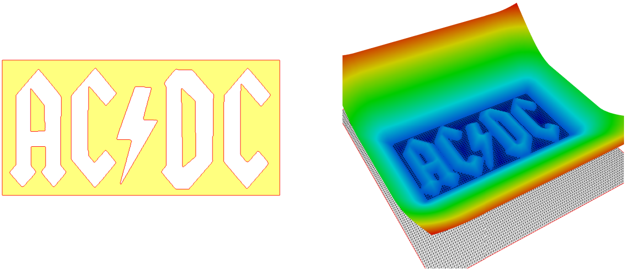
Pushing nodes according to the velocity field may result in an invalid configuration.

A short detour by the Level Set Method

A paradigm: [OSe] *the motion of an evolving domain is best described in an implicit way.*

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

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



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

Surface evolution equations in the level set framework

The motion of an evolving domain $\Omega(t) \subset \mathbb{R}^d$ along a velocity field $v(t,x) \in \mathbb{R}^d$ translates in terms of an associated 'level set function' $\phi(t,.)$ into the level set advection equation:

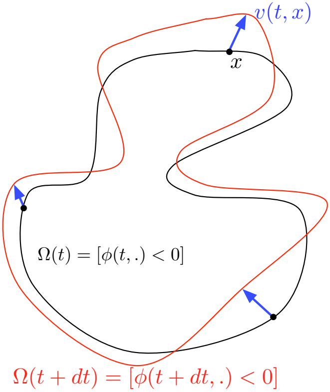
$$\forall t,\ \forall x \in \mathbb{R}^d,\ \frac{\partial \phi}{\partial t}(t,x) + v(t,x).\nabla\phi(t,x) = 0$$

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

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

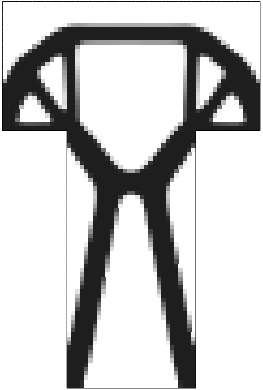
Then the evolution equation rewrites as a Hamilton-Jacobi equation:

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



The level set method of Allaire-Jouve-Toader [AlJouToa]

- The shapes Ω^n are embedded in a working domain D equipped with a fixed mesh.
- The successive shapes Ω^n are accounted for in the level set framework, i.e. via a function $\phi^n : D \to \mathbb{R}$ 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 $D \setminus \Omega^n$ is filled with a very 'soft' material, which leads to an approximate system posed on D.
- This approach is very versatile and does not require a mesh of the shapes at each iteration.

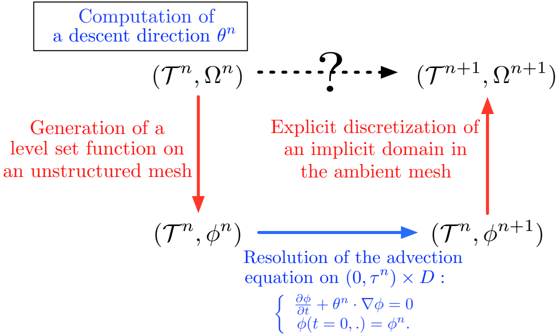


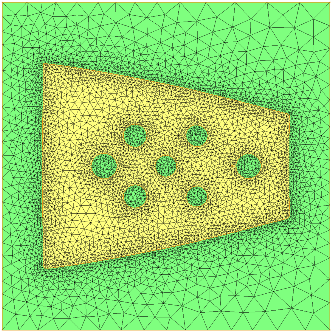
Shape accounted for with a level set description

The proposed method for handling mesh evolution

The mesh $\mathcal{T}^n$ of D is unstructured and changes at each iteration n, so that Ω^n is explicitly discretized in $\mathcal{T}^n$.

- Finite element analyses are held on Ω^n by 'forgetting' the part of $\mathcal{T}^n$ for the void $D \setminus \Omega^n$.
- The advection step $\Omega^n \to \Omega^{n+1}$ is carried out on the whole mesh $\mathcal{T}^n$, using a level set description ϕ^n of Ω^n.





Shape equipped with a mesh, conformally embedded in a mesh of the computational box.

Initializing level-set functions with the signed distance function (I)

Definition 2.

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

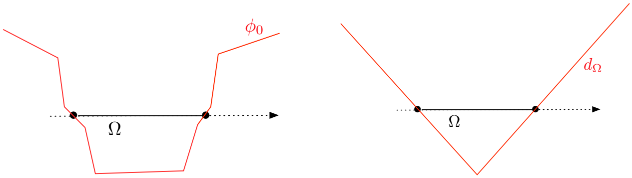
$$d_\Omega(x) = \begin{cases} -d(x, \partial\Omega) & \text{if } x \in \Omega \\ 0 & \text{if } x \in \partial\Omega \\ d(x, \partial\Omega) & \text{if } x \in \overline{{}^c\Omega} \end{cases},$$

where $d(\cdot, \partial\Omega)$ is the usual Euclidean distance function.

Initializing level-set functions with the signed distance function (II)

- The signed distance function to a domain $\Omega \subset \mathbb{R}^d$ is the 'canonical' way to initialize a level set function, owing to its unit gradient property:

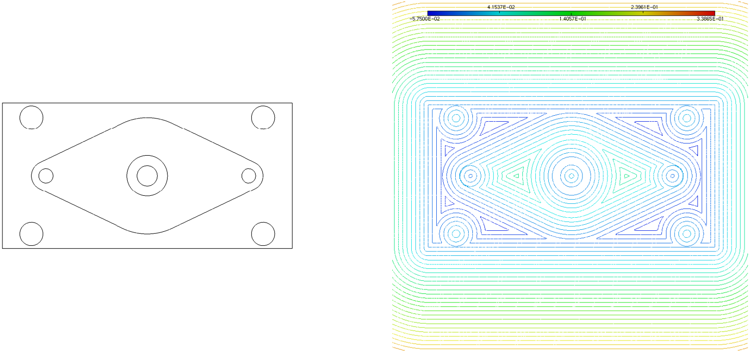
$$|\nabla d_\Omega(x)| = 1, \quad \text{p.p } x \in \mathbb{R}^d.$$



(Left) any level set function for $\Omega = (0,1) \subset \mathbb{R}$; (right) signed distance function to Ω.

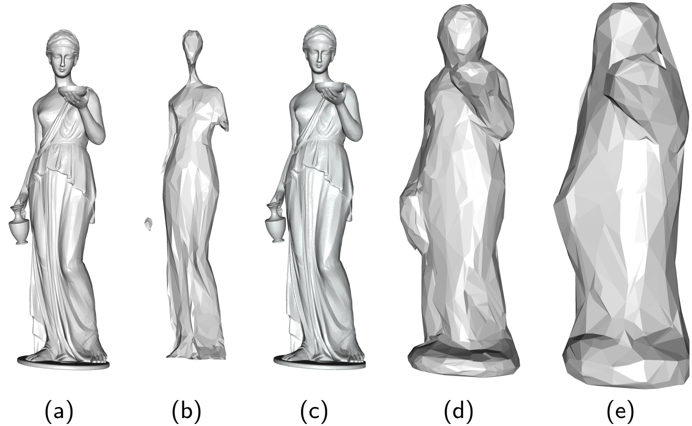
- Many existing approaches: Fast Marching Method [Se], Fast Sweeping method [Zha], mostly on Cartesian grids, or particular unstructured meshes.

A 2d computational example



Computation of the signed distance function to a discrete contour (left), on a fine background mesh ($\approx$ 250000 *vertices).*

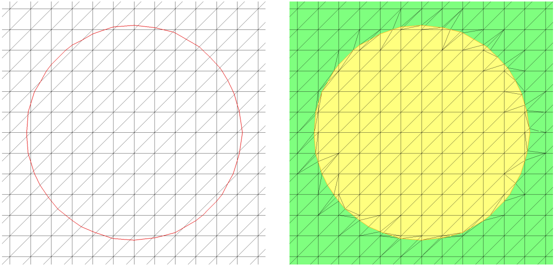
A 3d example... the 'Aphrodite'.



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

Meshing the negative subdomain of a level set function

Discretizing explicitely the 0 level set of a function $\phi : D \to \mathbb{R}$ defined at the vertices of a simplicial mesh $\mathcal{T}$ of a computational box D is fairly easy, using patterns.



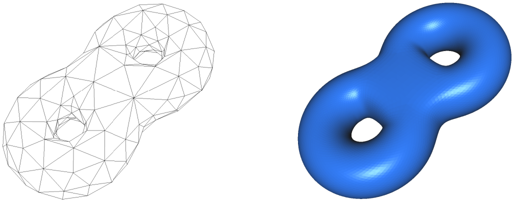
(left) 0 *level set of a scalar function defined over a mesh; (right) explicit discretization in the mesh.*

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.

$\Rightarrow$ Need to improve the quality of a mesh, while retaining its geometric features.

Local remeshing in $3d$

- Let $\mathcal{T}$ be an initial - valid, yet potentially ill-shaped - tetrahedral mesh. $\mathcal{T}$ carries a surface mesh $\mathcal{S}_{\mathcal{T}}$, whose triangles are 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 $\partial\Omega$.

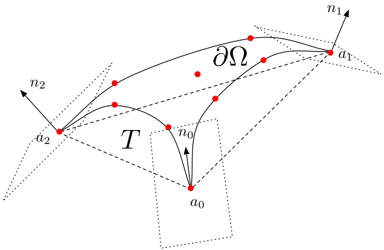


Poor geometric approximation (left) of a domain with smooth boundary (right)

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

Local remeshing in 3d: definition of an ideal domain

- In realistic cases, the underlying ideal domain Ω of $\mathcal{T}$ is unknown.
- However, from the knowledge of $\mathcal{T}$ (and $\mathcal{S}_{\mathcal{T}}$), one can reconstruct geometric features of Ω or $\partial\Omega$: normal vectors at regular points of $\partial\Omega$,...
- These features allow to set rules for the creation of a local parametrization of $\partial\Omega$ around a surface triangle $T \in \mathcal{S}_{\mathcal{T}}$, e.g. as a Bézier surface.



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

Local remeshing in $3d$: remeshing strategy

- Four local remeshing operators are intertwined, to iteratively increase the quality of the mesh $\mathcal{T}$: edge split, edge collapse, edge swap, and vertex relocation.

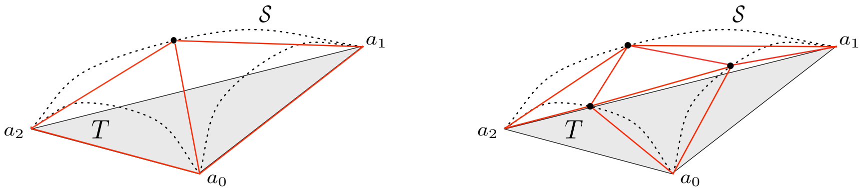
- Each one of them exists under two different forms, depending on whether it is applied to a surface configuration, or an internal one.

- A size map h is defined, to reach a good mesh sampling. It generally depends on the principal curvatures κ_1, κ_2 of $\partial\Omega$, but may also be user-defined (e.g. in a context of mesh adaptation).

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 $\partial\Omega$,...

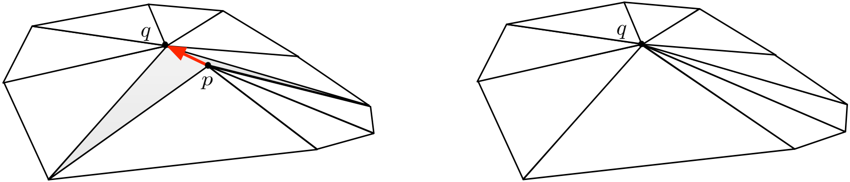


Splitting of one (left) or three (right) edges of triangle T, positioning the three new points on the ideal surface $\mathcal{S}$ (dotted).

Local mesh operators: edge collapse

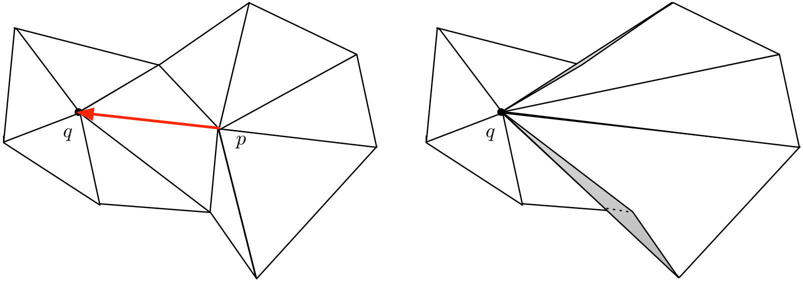
If an edge pq is too short, merge its two endpoints.

- 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 $\partial\Omega$,...



Collapse of point p over q.

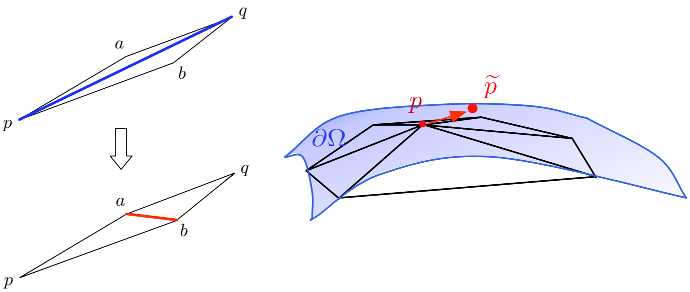
Local mesh operators: edge collapse



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

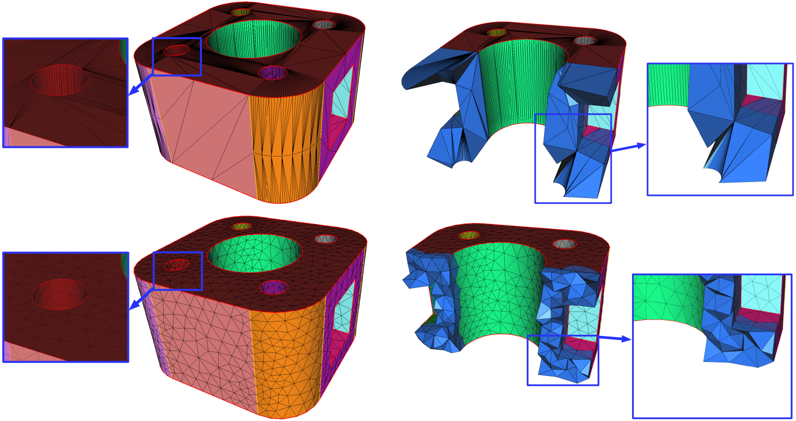
Local mesh operators: edge swap, node relocation

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.



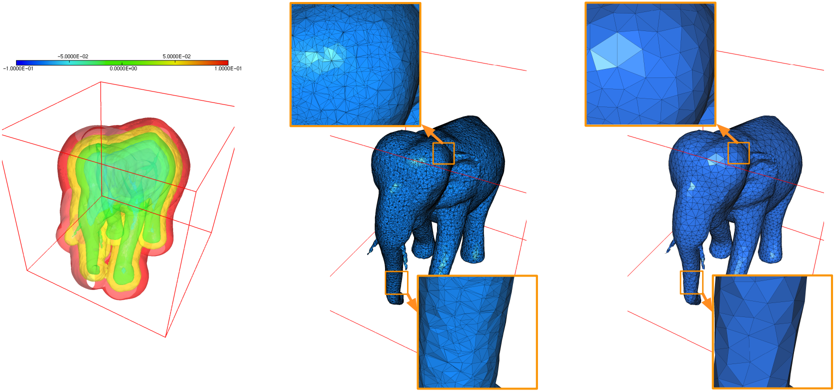
(left) 2d swap of edge pq, creating edge ab ; (right) relocation of node x to $\widetilde{x}$, along the surface.

Local remeshing in 3d: numerical examples



Mechanical part before (left) and after (right) remeshing.

Local remeshing in $3d$: numerical examples



(left) Some isosurfaces of an implicit function defined in a cube, (centre) result after rough discretization in the ambient mesh, (right) result after local remeshing.

Numerical implementation

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

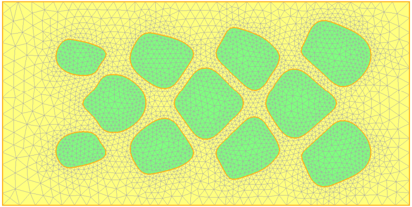
- When dealing with finite element computations on Ω^n, the part of $\mathcal{T}^n$, exterior to Ω^n is simply 'forgotten'.

- 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}.

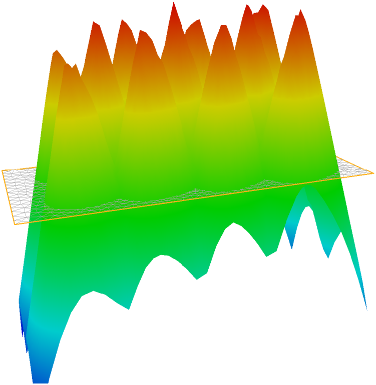
- From the knowledge of ϕ^{n+1}, a new unstructured mesh $\mathcal{T}^{n+1}$, in which the new shape Ω^{n+1} explicitly appears, is recovered.

The algorithm in motion...

Step 1: Start with the actual shape Ω^n, and generate its signed distance function d_{Ω^n} over D, equipped with the mesh $\mathcal{T}^n$.

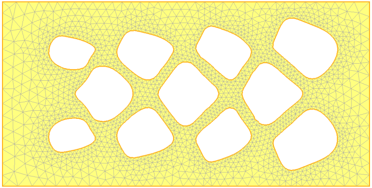


(a) The initial shape

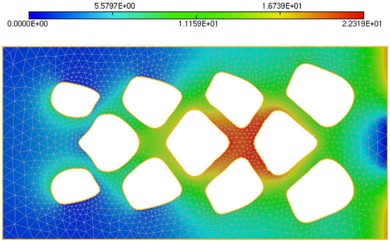


(b) Graph of d_{Ω^n}

Step 2: "Forget" the exterior of the shape $D \setminus \Omega^n$, and perform the computation of the shape gradient $J'(\Omega^n)$ on (the mesh of) Ω^n.



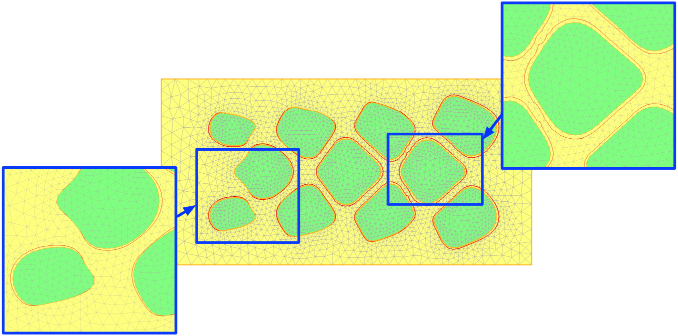
(a) The "interior mesh"



(b) Computation of $J'(\Omega^n)$

The algorithm in motion...

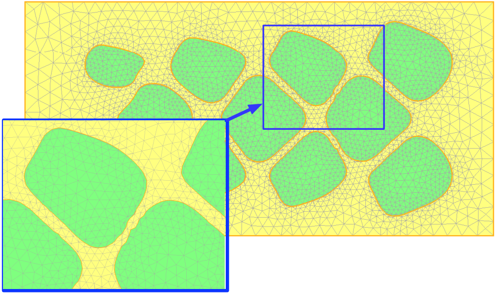
Step 3: "Remember" the whole mesh $\mathcal{T}^n$ of D. Extend the velocity field $J'(\Omega^n)$ to the whole mesh, and advect d_{Ω^n} along $J'(\Omega^n)$ for a (small) time step τ^n. A new level set function ϕ^{n+1} is obtained on $\mathcal{T}^n$, corresponding to the new shape Ω^{n+1}.



The shape Ω^n, discretized in the mesh (in yellow), and the "new", advected 0-level set (in red).

The algorithm in motion...

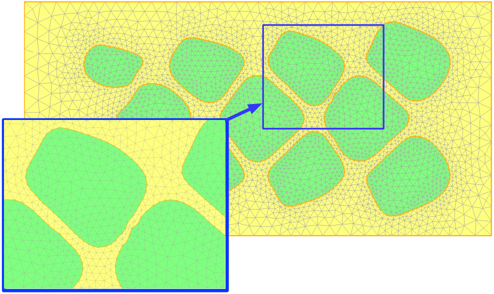
Step 4: To close the loop, the 0 level set of ϕ^{n+1} is explicitly discretized in mesh $\mathcal{T}^n$. As expected, roughly "breaking" this line generally yields a very ill-shaped mesh.



Rough discretization of the 0 level set of ϕ^{n+1} into $\mathcal{T}^n$; the resulting mesh of D is ill-shaped.

The algorithm in motion...

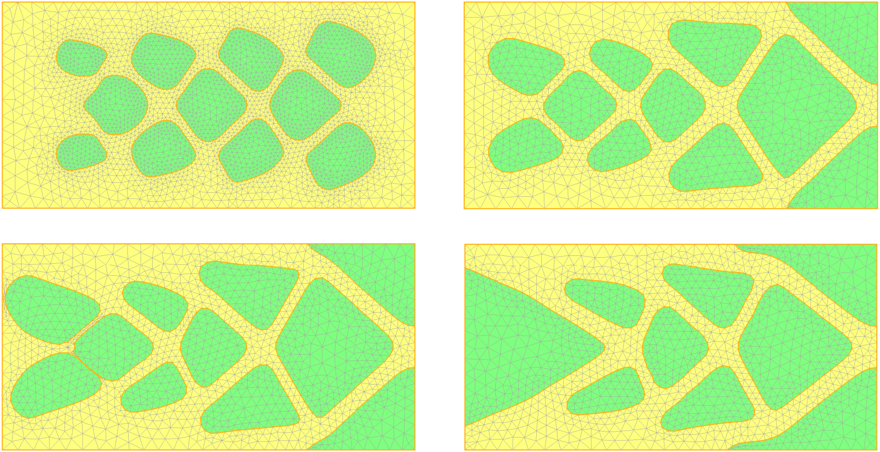
The mesh modification step is then performed, so as to enhance the overall quality of the mesh according to the geometry of the shape. $\mathcal{T}^{n+1}$ is eventually obtained.

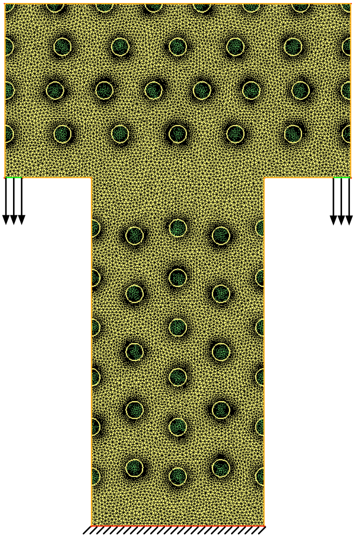


Quality-oriented remeshing of the previous mesh ends with the new, well-shaped mesh $\mathcal{T}^{n+1}$ of D in which Ω^{n+1} is explicitly discretized.

The algorithm in motion...

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





The ‘benchmark’ two-dimensional optimal mast test case.

- Minimization of the compliance

$$C(\Omega) = \int_\Omega Ae(u_\Omega) : e(u_\Omega)\, dx.$$

- A volume constraint is enforced.

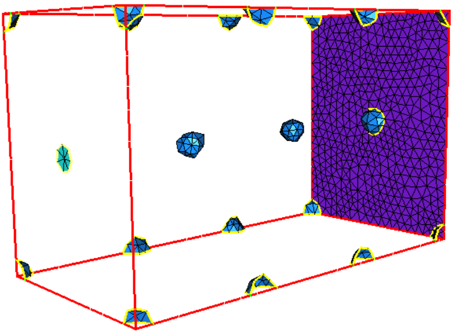
Numerical results: 3d cantilever

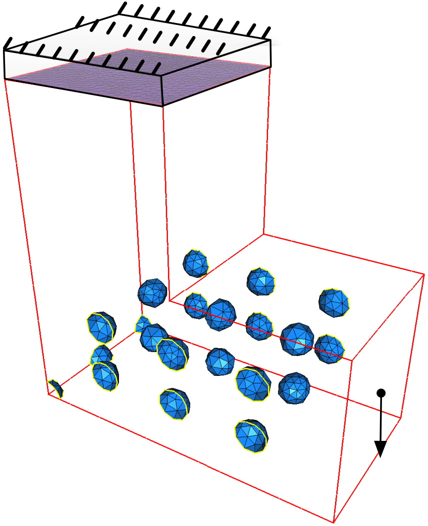
The 'benchmark' three-dimensional cantilever test case.

- Minimization of the compliance

$$C(\Omega) = \int_\Omega Ae(u_\Omega) : e(u_\Omega) \, dx.$$

- A volume constraint is enforced by means of a fixed Lagrange multiplier.





Optimal design of a 3d L-shaped beam.

- Minimization of a stress-based criterion

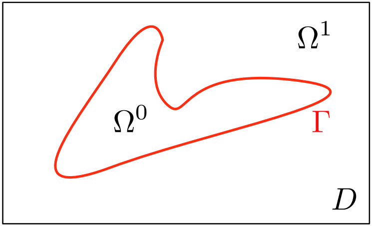
$$S(\Omega) = \int_\Omega k(x) ||\sigma(u_\Omega)||^2 \, dx,$$

where k is a weight factor, and $\sigma(u) = Ae(u)$ is the stress tensor.

- A volume constraint is enforced by means of a fixed Lagrange multiplier.

Another example in multiphase optimization

Optimal repartition of two materials A_0, A_1 occupying subdomains Ω^0 and $\Omega^1 := D \setminus \Omega^0$ of a fixed working domain D, with total (discontinuous) Hooke's law $A_{\Omega^0} := A_0 \chi_{\Omega^0} + A_1 \chi_{\Omega^1}$.



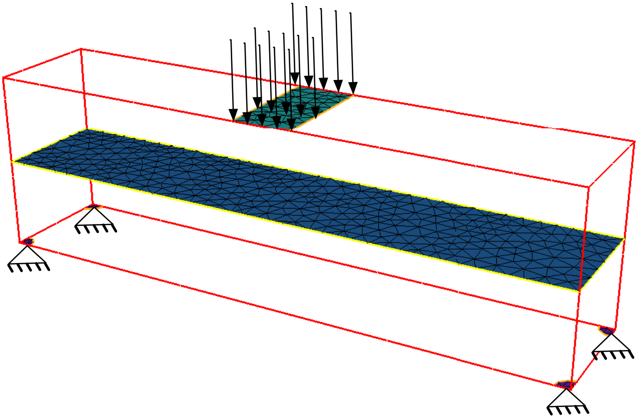
- Minimization of the compliance $C(\Omega^0) = \int_D A_{\Omega^0} e(u_{\Omega^0}) : e(u_{\Omega^0}) \, dx$ of D
- Shape derivative (see [Allaire, Jouve, Van Goethem]):

$$C'(\Omega^0)(\theta) = \int_\Gamma \mathcal{D}(u,u) \, \theta \cdot n \, ds.$$

- Evaluating $\mathcal{D}(u,u)$ is awkward in a fixed mesh context, for it involves jumps of the (discontinuous) strain and stress tensors $e(u)$ and $\sigma(u)$ at the interface Γ.

Numerical results: a multiphase beam

- Minimization of the compliance of a beam D, with respect to the repartition of the constituent materials A_0, A_1 ($E^1 = E^0/3$).
- A constraint on the volume of the stiffer material is enforced by means of a fixed Lagrange multiplier.

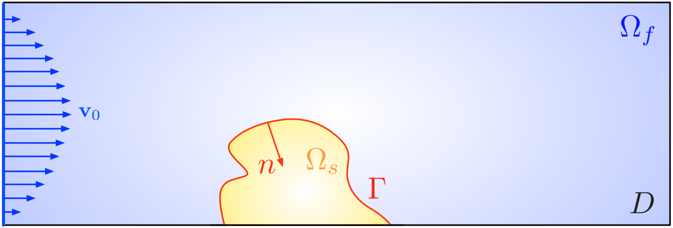


An advanced example in fluid-structure interaction (I)

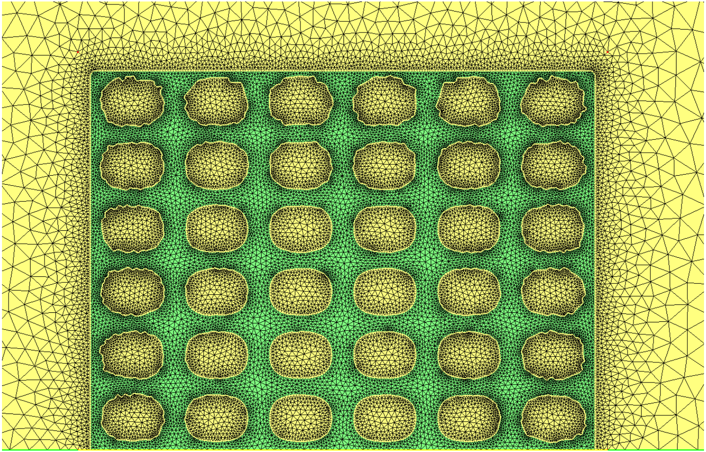
- A solid obstacle $\Omega_s := \Omega$ is placed inside a fixed cavity D where a fluid is flowing, occupying the phase $\Omega_f := D \setminus \Omega_s$.
- The fluid obeys the Navier-Stokes equations ($Re = 60$), and the solid is governed by the linearized elasticity system.
- Weak coupling between 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

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

under a volume constraint.



An advanced example in fluid-structure interaction (II)



Thank you !

References I

[AdDaFre] 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.

[AlJouToa] G. Allaire, F. Jouve and A.-M. Toader, *Structural optimization using sensitivity analysis and a level-set method*, Journal of computational physics, 194 (2004), pp. 363–393.

[FepAlBorCorDa] F. Feppon, G. Allaire, F. Bordeu, J. Cortial and C. Dapogny, *Shape optimization of a coupled thermal fluid-structure problem in a level set mesh evolution framework*, submitted, (2018).

[Cho] D. Chopp, *Computing minimal surfaces via level-set curvature flow*, J. Comput. Phys., 106, pp. 77-91 (1993).

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

References II

[DaDobFre] C. Dapogny, C. Dobrzinsky 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.

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

[MuSi] F. Murat and J. Simon, *Sur le contrôle par un domaine géométrique*, Technical Report RR-76015, Laboratoire d'Analyse Numérique (1976).

[OSe] S.J. Osher and J.A. Sethian, *Fronts propagating with curvature-dependent speed : Algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., 79 (1988), pp. 12–49.

[Se] J.A. Sethian, *A Fast Marching Method for Monotonically Advancing Fronts*, Proc. Nat. Acad. Sci., 93 (1996), pp. 1591–1595.

[Zha] H. Zhao, *A Fast Sweeping Method for Eikonal Equations*, Math. Comp., 74 (2005), pp. 603–627.