An introduction to the level set method and its applications to meshing

Charles Dapogny[1]

[1] CNRS & Laboratoire Jean Kuntzmann, Université Grenoble Alpes, Grenoble, France

6th March, 2023

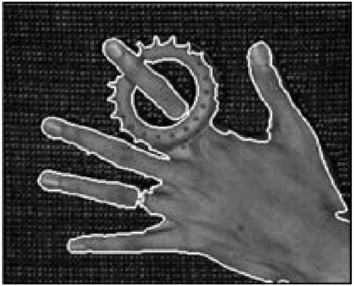
Foreword (I)

- Since [DerTho, OSe], the level set method is key in representing shapes.

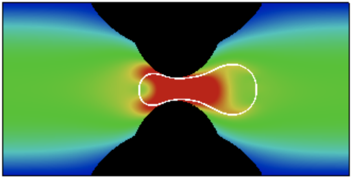
- It allows to account for dramatic motions of shapes, including topological changes:
  - It is parametrization free;
  - It is (originally) implemented on a fixed background mesh, alleviating meshing issues.

- The level set method is now ubiquitous in dynamic simulation (CFD, ...), image processing, shape optimization, etc.

- *References:* [OFed], [Sethian].



Active contour algorithm for image segmentation [CreRouDe].

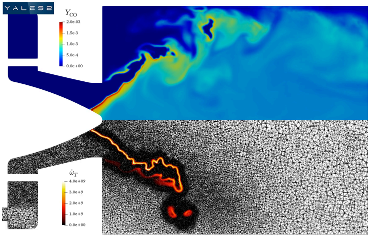


Motion of a vesicle through an obstacle [JeBruMai].

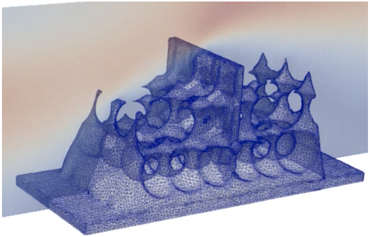
Foreword (II)

- The level set method has recently found an interesting interplay with meshing:
  - Adaptive refinement around an (evolving) interface;
  - Body-fitted interface tracking;
  - Mesh generation!

- *Goals of this course:*
  - Provide a concise introduction to the theory and practice of the level set method;
  - Present several applications in connection with meshing.



Numerical simulation of a burner (Coria).



Body-fitted optimization of an obstacle.

Disclaimer

Disclaimer

- This course merely sketches the rich and difficult subject of the level set method.
- The selected applications are biased by the knowledge of the author.

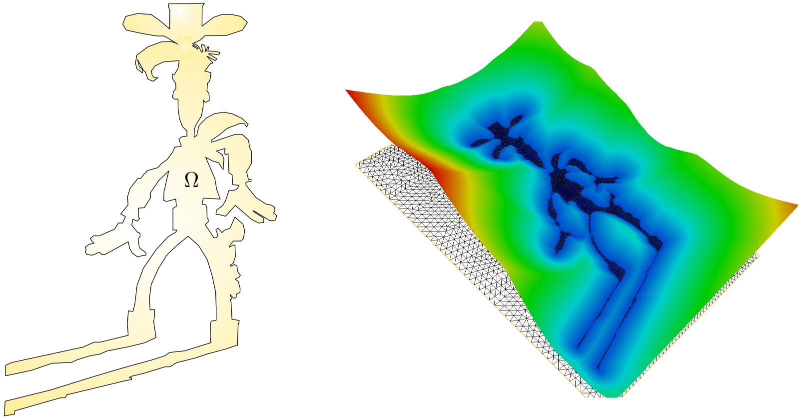
Plan of the course

1. The level set method
    - Basics about the level set method
    - Evolving domains within the level set framework
    - An interesting particular case: eikonal equations
2. Numerical algorithms for the level set method
    - Calculation of the signed distance function to a domain
    - Resolution of the level set evolution equation
    - Numerical practice of the level set method
3. (Re)meshing in connection with the level set method
    - A few definitions and key concepts
    - Mesh refinement adapted to a level set function
    - Isosurface discretization
4. Applications
    - Volume mesh generation from an invalid surface triangulation
    - Mesh adaptation to an isovalue
    - Body-fitted tracking of an interface

A paradigm: *The motion of a 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:

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



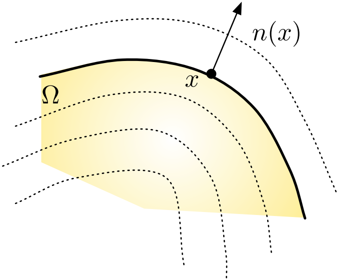
(Left) One domain $\Omega \subset \mathbb{R}^2$; (right) graph of an associated level set function.

Basics about the level set method (II)

Let $\Omega \subset \mathbb{R}^d$ be a domain with smooth boundary Γ, and let $\phi : \mathbb{R}^d \to \mathbb{R}$ be a smooth level set function for Ω, such that $\nabla\phi(x) \neq 0$ on a neighborhood of Γ.

- The unit normal vector n to Γ pointing outward Ω reads:

$$\forall x \in \Gamma, \;\; n(x) = \frac{\nabla\phi(x)}{|\nabla\phi(x)|}.$$



Normal vector n to the boundary Γ of Ω; some isolines of the function ϕ are dotted.

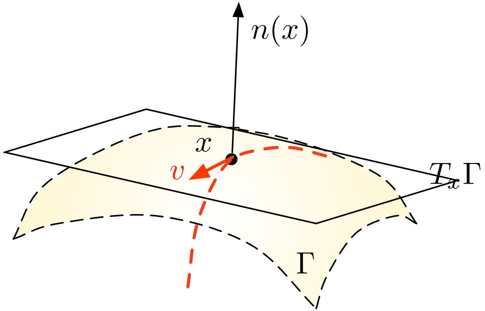
Basics about the level set method (III)

- The second fundamental form II of Γ is:

$$\forall x \in \Gamma, \;\; \mathrm{II}(x) = \nabla \left(\frac{\nabla \phi(x)}{|\nabla \phi(x)|} \right).$$

- The mean curvature κ of Γ is:

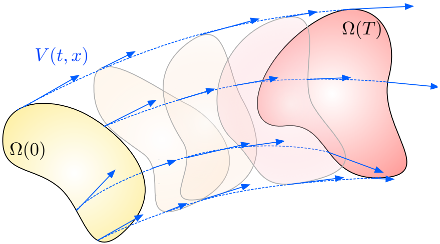
$$\forall x \in \Gamma, \;\; \kappa(x) = \mathrm{div} \left(\frac{\nabla \phi(x)}{|\nabla \phi(x)|} \right).$$



$\mathrm{II}_x(v, v)$ is the curvature of a curve drawn on Γ with tangent vector v at x.

Evolving domains (I)

- Let $\Omega(t) \subset \mathbb{R}^d$ be a domain with boundary $\Gamma(t)$, evolving over a time period $(0, T)$ according to a velocity field $V(t, x)$.



- Let $\phi(t, \cdot)$ be a level set function for $\Omega(t)$: $\begin{cases} \phi(t,x) < 0 & \text{if } x \in \Omega(t), \\ \phi(t,x) = 0 & \text{if } x \in \Gamma(t), \\ \phi(t,x) > 0 & \text{if } x \in {}^c\overline{\Omega(t)}. \end{cases}$

Questions

- How does the motion of $\Omega(t)$ translate in terms of $\phi(t, \cdot)$?
- ... To start with, what does it mean for $\Omega(t)$ to evolve according to $V(t, x)$?

Evolving domains (II)

The motion of $\Omega(t)$ may be classified into three categories depending on the nature of the velocity field $V(t,x)$.

1. $\Omega(t)$ is passively transported by $V(t,x)$ when the latter is externally prescribed, i.e. it does not depend on $\Omega(t)$.

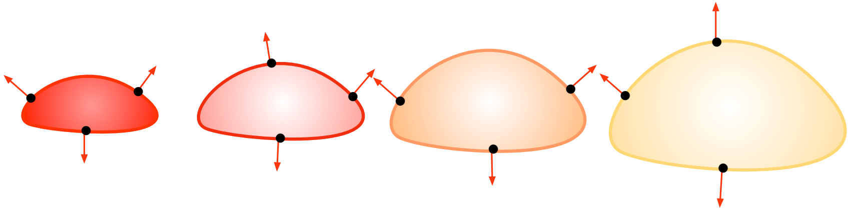
2. The velocity $V(t,x)$ depends on local features of $\Omega(t)$ or $\Gamma(t)$, such as:
   - The normal vector $n_t(x)$ at $x \in \Gamma(t)$;
   - The mean curvature $\kappa_t(x)$ of $\Gamma(t)$.

3. The field $V(t,x)$ depends on global features of the domain $\Omega(t)$, e.g. it depends on the solution to a partial differential equation (PDE) posed on $\Omega(t)$.

Example (I): velocity depending on local features of $\Omega(t)$

The flame propagation model $\Omega(t)$ represents a burnt region, whose front expands with constant, normal velocity c:

$$V(t,x) = c\, n_t(x), \text{ where } c > 0 \text{ is a constant.}$$



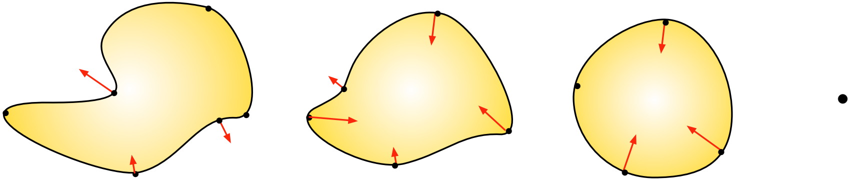
An example of the dynamics in the flame propagation model.

Example (II): another velocity depending on local features of $\Omega(t)$

The mean curvature flow The velocity field $V(t,x)$ reads:

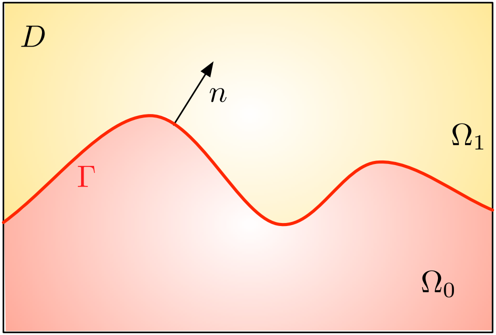
$$V(t,x) = -\kappa_t(x)\, n_t(x),$$

that is, $\Omega(t)$ evolves by "resorption of its bumps", and "filling of its creases".



An example of the dynamics of the mean curvature flow: Grayson's theorem [Grayson].

A domain $D \subset \mathbb{R}^d$ is filled with two immiscible fluids, occupying complementary phases Ω_0, Ω_1, with different densities ρ_0, ρ_1 and dynamic viscosities ν_0, ν_1.



Model configuration of a bifluid problem.

Example (III-b): velocity depending on global features of $\Omega(t)$

- The velocity $u(t,x)$ and pressure $p(t,x)$ inside D solve the unsteady Navier-Stokes equations:

$$\left\{\begin{array}{ll} \rho_i\left(\frac{\partial u}{\partial t} + u\cdot\nabla u\right) - \nu_i \Delta u + \nabla p = f_i & \text{for } (t,x)\in(0,T)\times\Omega_i(t), \\ \mathrm{div}(u_i) = 0 & \text{for } (t,x)\in(0,T)\times\Omega_i(t), \\ u_i(t,x) = 0 & \text{for } (t,x)\in(0,T)\times\partial D, \\ u_0(t,\cdot) = u_1(t,\cdot) & \text{on } \Gamma(t), \\ (\sigma_0 - \sigma_1)\cdot n_t = -\gamma\kappa_t n_t & \text{on } \Gamma(t), \\ u_i(t=0,\cdot) \text{ given} & \text{on } \Omega_i(0). \end{array}\right.$$

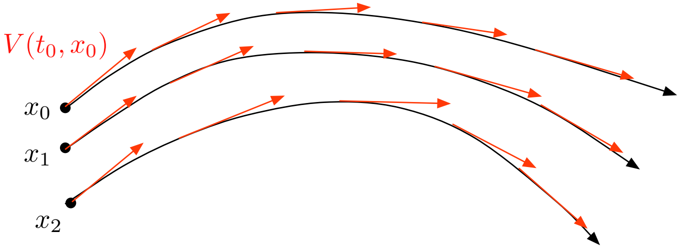
- The interface $\Gamma(t)$ between both fluids moves along the velocity of the fluid:

$$V(t,x) = u_0(t,x) = u_1(t,x), \;\; t\geq 0, \; x\in\Gamma(t).$$

Definition 1.

Let $V : \mathbb{R}_t \times \mathbb{R}^d_x \to \mathbb{R}^d$ *be a smooth velocity field. The* *characteristic curve* *emerging from a point* $x_0 \in \mathbb{R}^d$ *at time* $t = t_0$ *is the curve* $t \mapsto \chi(x_0, t, t_0)$ *defined by the ODE:*

$$\begin{cases} \frac{\mathrm{d}}{\mathrm{d}t}(\chi(x_0,t,t_0)) = V(t,\chi(x_0,t,t_0)), & \text{for } t \in (0,T) \\ \chi(x_0,t_0,t_0) = x_0. \end{cases}$$



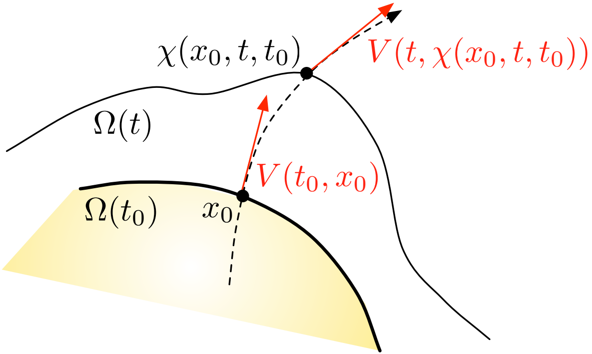
Three characteristic curves of the velocity field V *starting at* $t = t_0$ *from different points* x_0, x_1, x_2.

Evolving domains (IV): definition

"Intuitive" notion of an evolving domain

A domain $\Omega(t)$ evolves from an initial configuration $\Omega(t_0)$ according to $V(t,x)$ if it is obtained by advection of its points along V:

$$\Omega(t) = \left\{ \chi(x_0, t, t_0),\; x_0 \in \Omega(t_0) \right\}.$$



Evolving domains (V): the level set point of view

- Let $\Omega(t)$ be a (smooth) domain, moving over $(0, T)$ according to the (smooth) velocity field $V(t, x)$.

- Let $\phi(t, \cdot)$ be a smooth level set function for $\Omega(t)$, i.e:

$$\forall t \in (0, T),\ x \in \mathbb{R}^d,\ \begin{cases} \phi(t,x) < 0 & \text{if } x \in \Omega(t), \\ \phi(t,x) = 0 & \text{if } x \in \Gamma(t), \\ \phi(t,x) > 0 & \text{if } x \in {}^c\overline{\Omega(t)}. \end{cases}$$

- Let $x_0 \in \Gamma(0)$ be fixed; by the intuitive definition of an evolving domain, it comes:

$$\forall t \in (0, T),\ \ \phi(t, \underbrace{\chi(x_0, t, 0)}_{\in \Gamma(t)}) = 0.$$

- Differentiating and using the chain rule, we obtain:

$$\frac{\partial \phi}{\partial t}(t, \chi(x_0, t, 0)) + \frac{\mathrm{d}}{\mathrm{d}t}(\chi(x_0, t, 0)) \cdot \nabla \phi(t, \chi(x_0, t, 0)) = 0.$$

Evolving domains (VI): the level set point of view

- Since this holds for any point $x_0 \in \Gamma(0)$, we obtain the level set advection equation ($\neq$ "classical" advection equation):

$$\text{(LS-ADV)} \qquad \forall t \in (0,T),\ x \in \mathbb{R}^d,\ \frac{\partial \phi}{\partial t}(t,x) + V(t,x)\cdot \nabla\phi(t,x) = 0.$$

- If, in addition, the velocity is consistently oriented along the normal vector $n_t(x)$ to $\Omega(t)$, that is:

$$V(t,x) = v(t,x)\underbrace{\frac{\nabla\phi(t,x)}{|\nabla\phi(t,x)|}}_{n_t(x)}, \text{ for some scalar field } v(t,x),$$

the equation rewrites as the Level Set Hamilton-Jacobi equation ($\neq$ "classical" Hamilton-Jacobi equation):

$$\text{(LS-HJ)} \qquad \forall t \in (0,T),\ x \in \mathbb{R}^d,\ \frac{\partial \phi}{\partial t}(t,x) + v(t,x)|\nabla\phi(t,x)| = 0.$$

Evolving domains: comments (I)

- Strictly speaking, (LS-ADV) and (LS-HJ) only hold for pairs (t, x) with $x \in \Gamma(t)$. However, the previous analysis applies *mutatis mutandis* when

$$x_0 \in \Gamma_c(0) := \left\{ x \in \mathbb{R}^d, \ \phi(0, x) = c \right\}, \text{ for arbitrary } c \in \mathbb{R}.$$

Thus, the equation

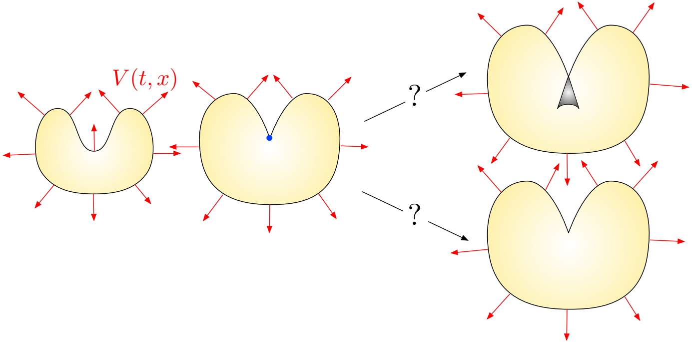
$$\forall t \in (0, T), \ x \in \mathbb{R}^d, \ \frac{\partial \phi}{\partial t}(t, x) + V(t, x) \cdot \nabla \phi(t, x) = 0$$

actually encodes that all the level sets of ϕ move according to $V(t, x)$.

- The velocity field $V(t, x)$ often makes sense only for $x \in \Gamma(t)$. In the above derivation, it is implicitly assumed that $V(t, x)$ has been extended to the whole space $\mathbb{R}^d$.

Evolving domains: comments (II)

- This analysis requires that $\Omega(t)$, $V(t,x)$ and $\phi(t,x)$ be "smooth" over $(0,T)$.
- Unfortunately, even when $\Omega(0)$ is smooth, the evolution of $\Omega(t)$ under very "simple" velocity fields $V(t,x)$ develops singularities in finite time.
- It is unclear how to even define the motion of $\Omega(t)$ after the onset of singularities.



In the flame propagation model, singularities develop in finite time (blue dot). Several definitions are possible for the subsequent motion of $\Omega(t)$.

- This ambiguity reflects that (LS-ADV) and (LS-HJ) have "too many" solutions.

Evolving domains: comments (III)

- This dilemma can be overcome thanks to the theory of viscosity solutions for Hamilton-Jacobi equations [CraILi].
- Under very mild assumptions, (LS-ADV) and (LS-HJ) have unique viscosity solutions, enjoying "nice" physical properties.

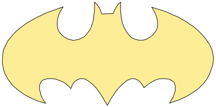
Mathematical notion of an evolving domain

1. Let $\phi_0(x)$ be one (any) level set function for $\Omega(0)$;
2. Let $\phi(t,x)$ be the unique viscosity solution to the evolution equation (LS-ADV) or (LS-HJ), with velocity field $V(t,x)$ and initial data ϕ_0.
3. The domain $\Omega(t)$ is defined by:

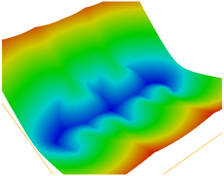
$$\Omega(t) = \left\{ x \in \mathbb{R}^d, \ \phi(t,x) < 0 \right\}.$$

- Mathematical references about this point of view: [AmDa], [Giga].

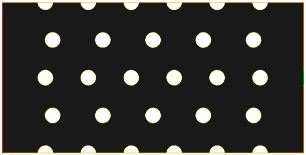
- Domain $\Omega \subset \mathbb{R}^d$.



- Level set function $\phi(t,x)$.

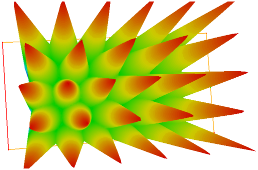


- Evolution w.r.t. a vector field $V(t,x)$.



- Resolution of the level set equation.

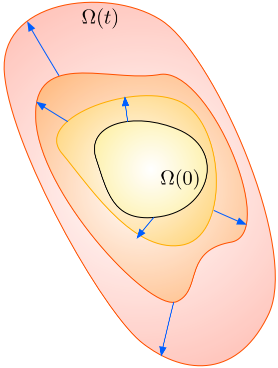
$$\frac{\partial \phi}{\partial t}(t,x) + V(t,x) \cdot \nabla\phi(t,x) = 0.$$



A stationary PDE for initial value problems (I)

- An interesting particular case of the above framework: $\Omega(t)$ expands (resp. retracts) along $n_t(x)$,

$$V(t,x) = c(x)n_t(x), \text{ where } c(x) > 0$$
$$(\text{resp. } c(x) < 0).$$



- A stationary PDE can be derived in terms of the time function $T(x)$:

$$T(x) = \inf\left\{t \geq 0,\ x \in \Omega(t)\right\}.$$

- The derivation of this PDE follows the same trail as that of the level set equations:
  1. It is first established rigorously when $\Omega(t)$, $V(t,x)$ and $T(x)$ are smooth,
  2. Then, a generalized notion of viscosity solutions is introduced to select a "physical" behavior for the solutions to the PDE where they are not smooth.

A stationary PDE for initial value problems (II)

- We rely again on the intuitive notion of an evolving domain.

  Let $x_0 \in \Gamma(0)$, and $t \mapsto x(t)$ be the characteristic curve of $V(t,x)$, emerging from x_0 at $t = 0$:

$$\begin{cases} x'(t) = c(x(t))n_t(x(t)), \\ x(0) = x_0. \end{cases}$$

- By definition of the time function, it holds:

$$\Omega(t) = \left\{ x \in \mathbb{R}^d, \;\; T(x) < t \right\}, \text{ and } \Gamma(t) = \left\{ x \in \mathbb{R}^d, \;\; T(x) = t \right\}.$$

- In particular, $\phi(x) := T(x) - t$ is one level set function for $\Omega(t)$. Hence,

$$\forall t \geq 0, \;\; x \in \Gamma(t), \quad n_t(x) = \frac{\nabla T(x)}{|\nabla T(x)|}.$$

- On the other hand, differentiating the relation $T(x(t)) = t$, we obtain:

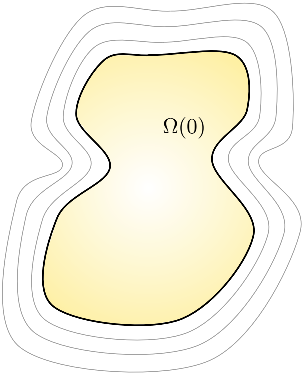
$$\forall t > 0, \quad x'(t) \cdot \nabla T(x(t)) = 1.$$

- Inserting

$$x'(t) = c(x(t)) \frac{\nabla T(x(t))}{|\nabla T(x(t))|},$$

it follows that T is solution to the Eikonal equation:

$$\begin{cases} c(x)|\nabla T(x)| = 1 & \text{for } x \in \mathbb{R}^d \setminus \overline{\Omega(0)}, \\ T(x) = 0 & \text{for } x \in \Gamma(0). \end{cases}$$



Some isolines of the time function T in the particular case where $c \equiv 1$.

A stationary PDE for initial value problems (IV)

- A similar analysis holds in the case where $\Omega(t)$ constantly retracts in the normal direction:

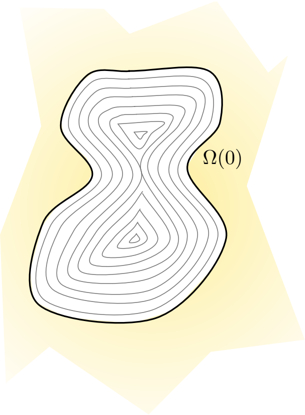
$$V(t,x) = -c(x)n_t(x), \text{ where } c(x) > 0.$$

- The time function $T : \Omega(0) \to \mathbb{R}$ is then defined by:

$$T(x) = \inf \left\{ t \geq 0, \; x \in \mathbb{R}^d \setminus \overline{\Omega(t)} \right\}.$$

- It turns out that T is solution to the Eikonal equation:

$$\begin{cases} c(x)|\nabla T(x)| = 1 & \text{for } x \in \Omega(0), \\ T(x) = 0 & \text{for } x \in \Gamma(0). \end{cases}$$



Some isolines of the time function T in the particular case where $c \equiv 1$.

A stationary PDE for initial value problems (V)

- Again, this derivation is rigorous only when $\Omega(t)$, $c(x)$ and $T(x)$ are smooth, ... which usually fails after some time $t > 0$.

- In general, the eikonal equation has to be understood in the framework of the theory of viscosity solutions, which guarantees its well-posedness under mild conditions.

A key example: distance functions

Theorem 1.

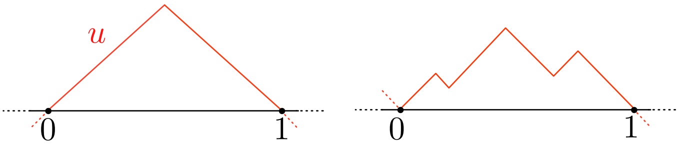
Assume that $c(x) > 0$ is continuous; the Eikonal equation

$$\begin{cases} c(x)|\nabla u(x)| = 1 & \text{in } \Omega, \\ u(x) = 0 & \text{on } \Gamma \end{cases}.$$

has a unique viscosity solution $u \in \mathcal{C}(\overline{\Omega})$.

In the particular case $c(x) \equiv 1$, u is the Euclidean distance function:

$$u(x) = d(x, \Gamma) = \inf_{y \in \Gamma} |x - y|.$$



(Left) graph of the distance function $u = d(\cdot, \Gamma)$, (right) graph of a function satisfying $|u'(x)| = 1$ a.e. which is not a viscosity solution of the equation $|u'| = 1$.

Calculation of the signed distance function (I)

- Often, a shape Ω is numerically encoded as a CAD model, a mesh, ...

- A preliminary step to the practice of the level set method is thus to create one level set function for Ω from such datum.

- Among all the level set functions for Ω, the signed distance function d_Ω enjoys multiple appealing features:
  - It helps the numerical stability of the level set method [Chopp];
  - It allows a more simple calculation of morphological quantities related to Ω (projection operator onto Γ, thickness, etc.).

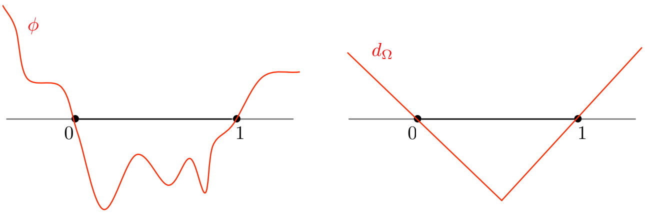
- Efficient algorithms exist to calculate d_Ω, such as the Fast Marching algorithm [SethianFMM], the Fast Sweeping algorithm [Zhao], ...

Definition 2.

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* $\partial\Omega$.

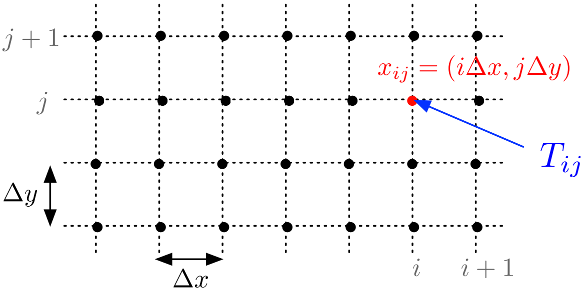


Two level set functions for the domain $\Omega = (0, 1) \subset \mathbb{R}$.

The fast marching method (I)

We skim over the fast marching method in the following simple context:

- Let Ω be a (smooth) bounded domain in $\mathbb{R}^2$.
- Let D be a large computational domain, equipped with a Cartesian grid $\mathcal{G}$ with steps Δx, Δy, and nodes $x_{ij} = (i\Delta x, j\Delta y)$.

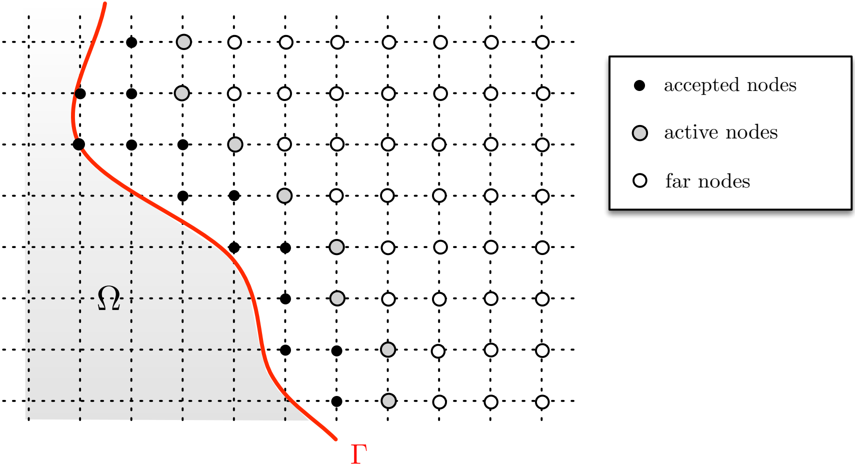


- We calculate the values $T_{ij} = T(x_{ij})$ of the unsigned distance function $T(x) = d(x, \Gamma)$ at the nodes $x_{ij} \in D \setminus \overline{\Omega}$.

The Fast Marching Method (II)

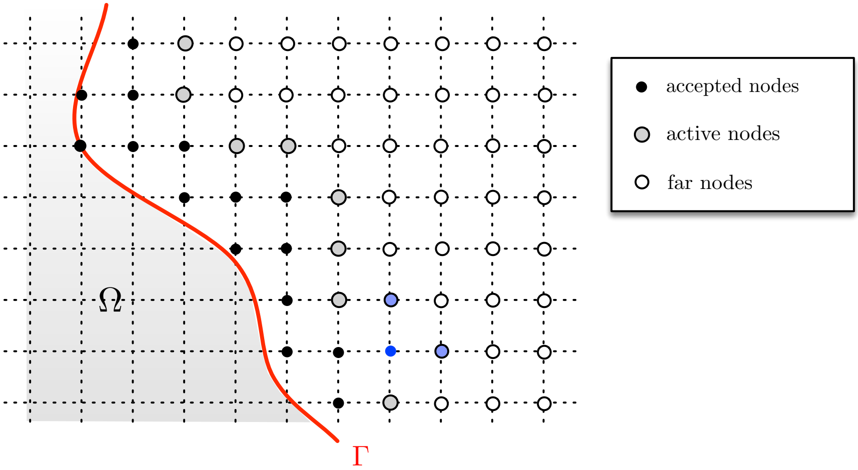
- The Fast Marching Method mimicks the propagation of a front.
- It is an iterative algorithm, producing a sequence $\left\{T^n_{ij}\right\}_{ij}$, $n = 0, \ldots$ of closer and closer approximations to the collection $\left\{T(x_{ij})\right\}_{ij}$.
- At each iteration n, the nodes of the grid are divided into three categories:
  - Accepted nodes are those x_{ij} "where the front has passed"; the value T^n_{ij} is no longer subject to modification.
  - Active nodes are "on the front", as the neighbors of accepted nodes. Trial values T^n_{ij} are available, which are still likely to be updated.
  - Far nodes are those "far from the front".
- Each iteration $n \to n+1$ hinges on
  - A marching procedure: the active node x_{ij} with lowest trial value T^n_{ij} is accepted, and the set of active nodes is updated accordingly.
  - A local update procedure: trial values at the neighbors of this node are re-computed.

The marching procedure



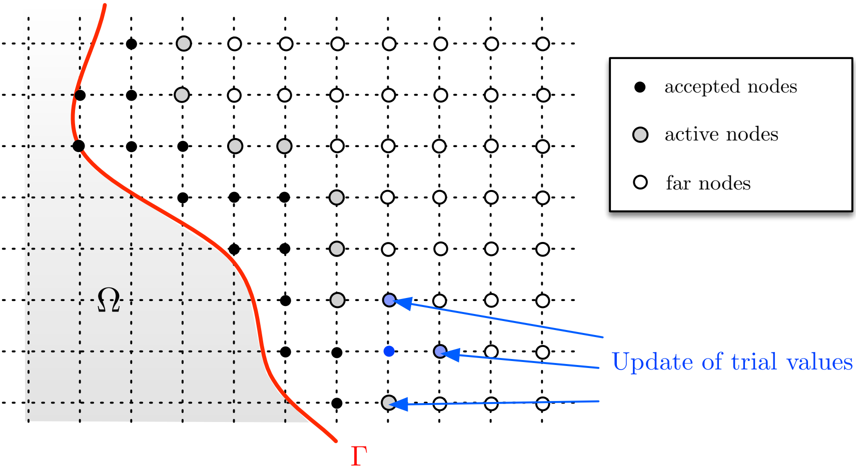
Setting of the fast marching method

The marching procedure



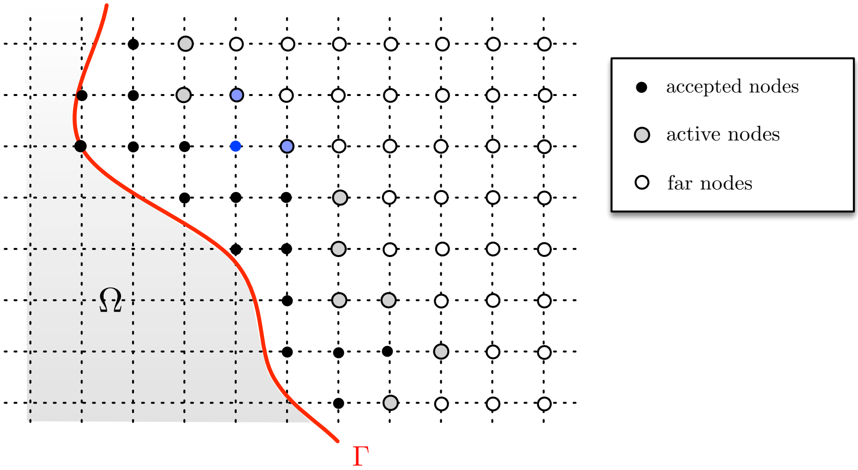
Setting of the fast marching method

The marching procedure



Setting of the fast marching method

The marching procedure



Setting of the fast marching method

The local update

- At an iteration $n \to n+1$, a temporary value $\widetilde{T_{ij}^n}$ is calculated at each active node x_{ij}, thanks to a discretization of the Eikonal equation:

$$|\nabla T(x)| = 1.$$

- The discretization is:

$$\sqrt{\begin{array}{l} \max\left(\max\left(\frac{\widetilde{T_{ij}^n} - T_{i-1j}^n}{\Delta x}, 0\right), -\min\left(\frac{T_{i+1j}^n - \widetilde{T_{ij}^n}}{\Delta x}, 0\right)\right)^2 \\ + \max\left(\max\left(\frac{\widetilde{T_{ij}^n} - T_{ij-1}^n}{\Delta y}, 0\right), -\min\left(\frac{T_{ij+1}^n - \widetilde{T_{ij}^n}}{\Delta y}, 0\right)\right)^2 \end{array}} = 1.$$

- The calculation of $\widetilde{T_{ij}^n}$ from the $\{T_{kl}^n\}_{kl}$ is upwind:
  - Only the accepted values within the set $\{T_{i-1j}^n, T_{i+1j}^n, T_{ij-1}^n, T_{ij+1}^n\}$ are used in the above formula.
  - Only solutions $\widetilde{T_{ij}^n}$ larger than these accepted values are retained.

- In the end, the new trial value T_{ij}^{n+1} is defined by:

$$T_{ij}^{n+1} = \min\left(\widetilde{T_{ij}^n}, T_{ij}^n\right).$$

The fast marching algorithm

- **Initialization:**

  1. Compute the exact distance function at the nodes of the cells which intersect Γ, and mark them as accepted.
  2. Use the local update procedure to compute a trial value at those neighbors of accepted points which are not accepted, and mark them as active.
  3. Mark all the remaining nodes as far, and assign them the value ∞.

- **Loop (while the set of active nodes is non empty):**

  1. Travel the set of active nodes, and identify that with minimum trial value. This node becomes accepted.
  2. Identify the new set of active nodes, and compute a new trial value for each one of them by using the local update solver for the Eikonal equation.

The fast marching algorithm: comments

- The fast marching method extends straightforwardly to the 3d case.
- It can also be adapted to general Eikonal equations:

$$c(x)|\nabla T(x)| = 1, \text{ where } c(x) > 0.$$

- *Computational cost:* The fast marching method requires $\mathcal{O}(M \log(M))$ operations, where M is the total number of nodes in the grid:
  - At each iteration, one node is accepted.
  - The only costly operation within one iteration is the search for the smallest element in the list of trial values.
  - In practice, a heapsort algorithm is used to make this search effficient - in $\mathcal{O}(\log(\widetilde{M}))$, where $\widetilde{M}$ is the number of trial values.
- Under mild hypotheses, one proves that the fast marching algorithm converges to the solution to the Eikonal equation [CriFa].

The fast marching algorithm: extension to a simplicial mesh

This method can be adapted to a (2d, surface, or 3d) simplicial mesh [KiSe].

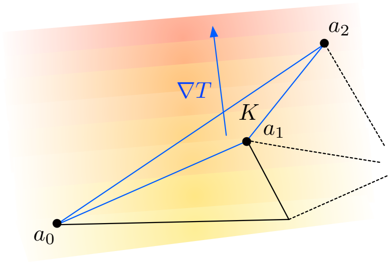
- The marching strategy is identical.
- A similar local update formula can be constructed from the equation

$$|\nabla T| = 1$$

  to infer a trial value at an active node of a triangle K from accepted values

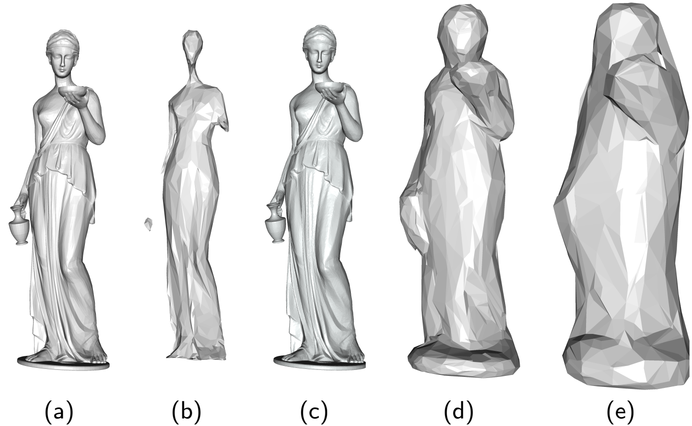
- An additional "causality" condition has to be enforced:

  The update made from the information in K has to "come from K."



Violation of "causality": the prediction based on the front approximated from triangle K fetches information outside K.

The signed distance function: a 3d example.



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

Resolution of the level set equations (I)

- Given an initial datum $\phi_0 : \mathbb{R}^d \to \mathbb{R}$, we aim to solve the level set evolution equation

$$\text{(LS-ADV)} \quad \begin{cases} \frac{\partial \phi}{\partial t}(t,x) + V(t,x) \cdot \nabla \phi(t,x) = 0 & \text{for } (t,x) \in (0,T) \times \mathbb{R}^d, \\ \phi(t=0,x) = \phi_0(x) & \text{for } x \in \mathbb{R}^d. \end{cases}$$

- In general, the velocity field $V(t,x)$ depends on $\phi(t,x)$ in a very complicated way (e.g. via a PDE posed on $\Omega(t)$), making the problem downright untractable.

- **Workaround** Split the time interval $(0,T)$ into a series of subintervals

$$(t^n, t^{n+1}), \text{ where } 0 = t^0 < t^1 < \ldots < t^N = T,$$

and approximate $V(t,x) \approx V^n(x)$ on each (t^n, t^{n+1}).

Example When $V(t,x)$ is the solution to a PDE posed on $\Omega(t)$, $V^n(x)$ is the solution to a PDE posed on $\Omega(t^n)$.

Resolution of the level set equations (II)

Two such approximations are possible:

1. The whole velocity field $V(t,x)$ is frozen over (t^n, t^{n+1}):

$$\forall t \in (t^n, t^{n+1}), \;\; V(t,x) \approx V^n(x) := V(t^n, x),$$

   and over each interval, a standard advection equation is solved:

$$\begin{cases} \frac{\partial \phi}{\partial t}(t,x) + V^n(x) \cdot \nabla \phi(t,x) = 0 & \text{on } (t^n, t^{n+1}) \times \mathbb{R}^d, \\ \phi(t = t^n, x) \text{ given} & \text{for } x \in \mathbb{R}^d. \end{cases}$$

2. Only the normal component of $V(t,x) = v(t,x)n_t(x)$ is frozen:

$$\forall t \in (t^n, t^{n+1}), \;\; V(t,x) \approx v^n(x) n_t(x), \text{ where } v^n(x) = v(t^n, x).$$

   Over each interval, a "classical" Hamilton-Jacobi equation is solved:

$$\begin{cases} \frac{\partial \phi}{\partial t}(t,x) + v^n(x)|\nabla \phi(t,x)| = 0 & \text{on } (t^n, t^{n+1}) \times \mathbb{R}^d, \\ \phi(t = t^n, x) \text{ given} & \text{for } x \in \mathbb{R}^d. \end{cases}$$

Resolution of the advection equation (I)

- We focus on the resolution of the advection equation over a generic time period $(0, T)$ ($=$ any of the (t^n, t^{n+1}) in the previous context):

$$\text{(ADV)} \qquad \begin{cases} \dfrac{\partial \phi}{\partial t}(t,x) + V(x) \cdot \nabla \phi(t,x) = 0 & \text{on } (0,T) \times \mathbb{R}^d, \\ \phi(0,.) = \phi_0 & \text{on } \mathbb{R}^d, \end{cases}$$

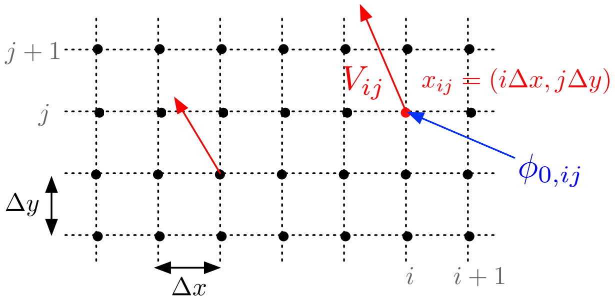
for given velocity field $V(x)$ $(= V^n(x))$, and initial function ϕ_0 $(= \phi(t^n, \cdot))$.

- Such equations are quite well-known in numerical analysis, and efficient numerical schemes exist.

- We present an algorithm based on the method of characteristics, see [Pironneau] and [Strain].

Resolution of the advection equation (II)

- Let $t^n = n\Delta t$ be a discretization of the time interval $(0, T)$.
- The computational domain D is equipped with a Cartesian grid $\mathcal{G}$ with steps Δx, Δy, and nodes $x_{ij} = (i\Delta x, j\Delta y)$.
- The initial datum ϕ_0 and the velocity field V are discretized at the vertices of $\mathcal{G}$.
- They are linearly interpolated from these values when needed elsewhere.



- We calculate the values $\phi_{\text{out}}(x_{ij}) \approx \phi(T, x_{ij})$ of the solution to (ADV).

The method of characteristics (I)

The exact solution $\phi(t,x)$ to the advection equation (ADV) is:

$$\underbrace{\phi(t,x)}_{\text{Value of } \phi \text{ at } (t,x)} = \underbrace{\phi(0,\chi(x,0,t))}_{\substack{\text{Initial value of } \phi \\ \text{at the initial position of the particle at } x \text{ at } t}},$$

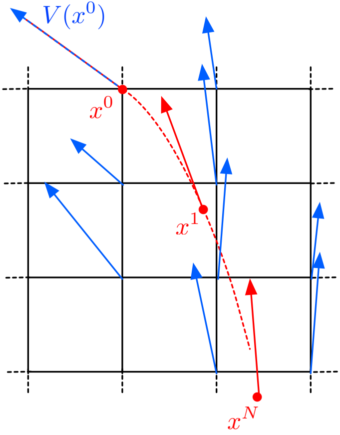
where the characteristic curve $t \mapsto \chi(x,t,t_0)$ emerging from a point $x \in \mathbb{R}^d$ at time $t = t_0$ is defined by the ODE:

$$\text{(C)} \qquad \begin{cases} \frac{\mathrm{d}}{\mathrm{d}t}(\chi(x,t,t_0)) = V(t,\chi(x,t,t_0)), & \text{for } t \in (0,T) \\ \chi(x,t_0,t_0) = x. \end{cases}$$

The method of characteristics (II)

A simple implementation of this strategy relies on Euler's method for (C):

- **Initialization:** Level set function ϕ_0 at the vertices of $\mathcal{T}$.
- **For all vertices** $x \in \mathcal{G}$**:**
  - Set $x^0 = x$;
  - **For** $n = 0, \dots, N-1$**:**
    1. Find $E \in \mathcal{G}$ such that $x^n \in E$;
    2. Calculate V at x^n by interpolation from its values at the vertices of E;
    3. $x^{n+1} = x^n - \Delta t V(x^n)$.
  - $\phi_{\text{out}}(x) = \phi_0(x^N)$.



The efficiency of this strategy can be improved by a Runge-Kutta scheme for (C).

Resolution of the Hamilton-Jacobi equation (I)

- We now discuss the resolution of the Hamilton-Jacobi equation over a generic time period $(0, T)$:

$$\text{(HJ)} \qquad \left\{ \begin{array}{ll} \frac{\partial \phi}{\partial t} + v(x)|\nabla \phi| = 0 & \text{on } (0,T) \times \mathbb{R}^d, \\ \phi(0,.) = \phi_0 & \text{on } \mathbb{R}^d, \end{array} \right.$$

  for given normal velocity $v(x)$, and initial function ϕ_0.

- The induced approximation of the "true" level set equation (LS-ADV) retains the information that the velocity field is consistently oriented along the normal vector $n_t(x)$ to $\Omega(t)$, and is thus appealing in many cases.

- The device of efficient algorithms for solving this equation relies on the theory of numerical schemes for first order Hamilton-Jacobi equations:

$$\left\{ \begin{array}{ll} \frac{\partial \phi}{\partial t} + H(x, \nabla \phi) = 0 & \text{on } (0,T) \times \mathbb{R}^d, \\ \phi(0,.) = \phi_0 & \text{on } \mathbb{R}^d, \end{array} \right. \qquad \text{(HJ)}$$

  in the particular case where $H(x,p) = v(x)|p|$ [Sethian].

Resolution of the Hamilton-Jacobi equation (II)

- For $i, j \in \mathbb{Z}$, we denote the finite difference quantities:

$$D_{ij}^{+x}\phi = \frac{\phi_{i+1j} - \phi_{ij}}{\Delta x} \quad ; \quad D_{ij}^{-x}\phi = \frac{\phi_{ij} - \phi_{i-1j}}{\Delta x},$$

and:

$$D_{ij}^{+y}\phi = \frac{\phi_{ij+1} - \phi_{ij}}{\Delta y} \quad ; \quad D_{ij}^{-y}\phi = \frac{\phi_{ij} - \phi_{ij-1}}{\Delta y}.$$

- Sethian's first-order scheme reads:

$$\begin{cases} \forall n \in \mathbb{N}, i, j \in \mathbb{Z}, & \phi_{ij}^{n+1} = \phi_{ij}^n - \Delta t \left(\max(v_{ij}, 0) \nabla_{ij}^+ \phi^n + \min(v_{ij}, 0) \nabla_{ij}^- \phi^n \right), \\ \forall i, j \in \mathbb{Z}, & \phi_{ij}^0 = \phi_0(i\Delta x, j\Delta y), \end{cases}$$

with the discretizations $\nabla_{ij}^+\phi$ and $\nabla_{ij}^-\phi$ of the gradient norm $|\nabla\phi|$ defined by:

$$\nabla_{ij}^+\phi = \left(\begin{array}{c} \max(\max(D_{ij}^{-x}\phi, 0), -\min(D_{ij}^{+x}\phi, 0))^2 \\ + \max(\max(D_{ij}^{-y}\phi, 0), -\min(D_{ij}^{+y}\phi, 0))^2 \end{array} \right)^{\frac{1}{2}},$$

and

$$\nabla_{ij}^-\phi = \left(\begin{array}{c} \max(\max(D_{ij}^{+x}\phi, 0), -\min(D_{ij}^{-x}\phi, 0))^2 \\ + \max(\max(D_{ij}^{+y}\phi, 0), -\min(D_{ij}^{-y}\phi, 0))^2 \end{array} \right)^{\frac{1}{2}}.$$

Resolution of the Hamilton-Jacobi equation (III)

- The quantity $\nabla^+_{ij}\phi$ (resp. $\nabla^-_{ij}\phi$) is upwind (resp. downwind): it is a finite difference approximation of $|\nabla\phi|$ at x_{ij} based only on the values among $\{\phi_{i-1j}, \phi_{i+1j}, \phi_{ij-1}, \phi_{ij+1}\}$ which are smaller (resp. larger) than ϕ_{ij}.

- The discretization of the (exact) Hamiltonian $H(x,p) = v(x)|p|$ by the numerical counterpart:

$$H(x_{ij}, \nabla\phi(x_{ij})) \approx \mathcal{H}_{ij}(\{\phi^n_{kl}\}_{k,l\in\mathbb{Z}}) := \max(v_{ij},0)\nabla^+_{ij}\phi^n + \min(v_{ij},0)\nabla^-_{ij}\phi^n$$

  is upwind: for given i,j,n, the update $\phi^n_{ij} \to \phi^{n+1}_{ij}$ is only carried out using information coming from

  - smaller values than ϕ^n_{ij} if v_{ij} is positive,
  - larger values than ϕ^n_{ij} if it is negative.

Resolution of the Hamilton-Jacobi equation (IV)

- This scheme can be proved to be convergent to the unique viscosity solution ϕ to (HJ), under the following CFL condition:

$$\left(\sup_{i,j} v_{ij}\right) \frac{\Delta t}{\min(\Delta x, \Delta y)} \leq 1, \text{ i.e.}$$

"The information cannot travel more than one cell during one time step".

- In addition, the following error estimate can be proved:

$$\forall i, j \in \mathbb{Z}, \forall n \leq N, \ |\phi_{ij}^n - \phi(t^n, x_{ij})| \leq C\sqrt{\Delta t}.$$

- The time accuracy of the scheme can be improved thanks to Runge-Kutta strategy.

- Its space accuracy can also be improve by using high-order, (Weighted) Essentially Non Oscillatory (W)ENO finite differences [OShu].

Resolution of the level set evolution equation on a simplicial mesh

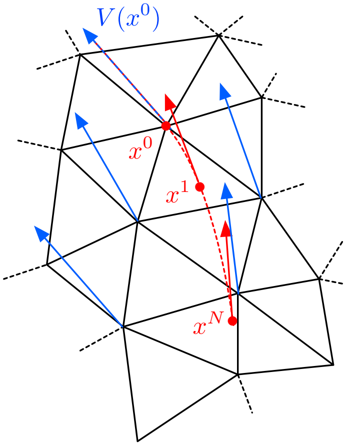
- The theory of schemes for Hamilton-Jacobi equations can be adapted to the context of a simplicial mesh $\mathcal{T}$, but it is a difficult task [Abgrall] [BaSe].

- On the contrary, the method of characteristics for the advection equation (ADV) extends pretty readily.

- The only additional difficulty is to efficiently locate the intermediate points

$$x^{n+1} = x^n - \Delta t V(x^n)$$

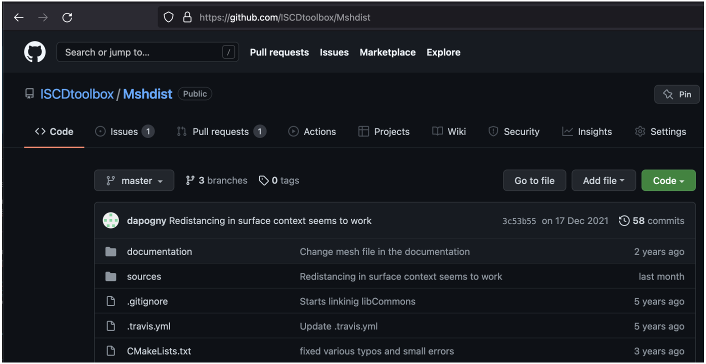
in the mesh $\mathcal{T}$.



A word of advertisement: open-source implementations

- mshdist [DaFre]: Calculation of the signed distance function on simplicial meshes.

  https://github.com/ISCDtoolbox/Mshdist



- advect [BuDaFre]: Resolution of the level set equations on simplicial meshes.

  https://github.com/ISCDtoolbox/Advect

The "classical" practice of the level set method (I)

- A fixed mesh $\mathcal{T}$ of the computational domain D is used.
- 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

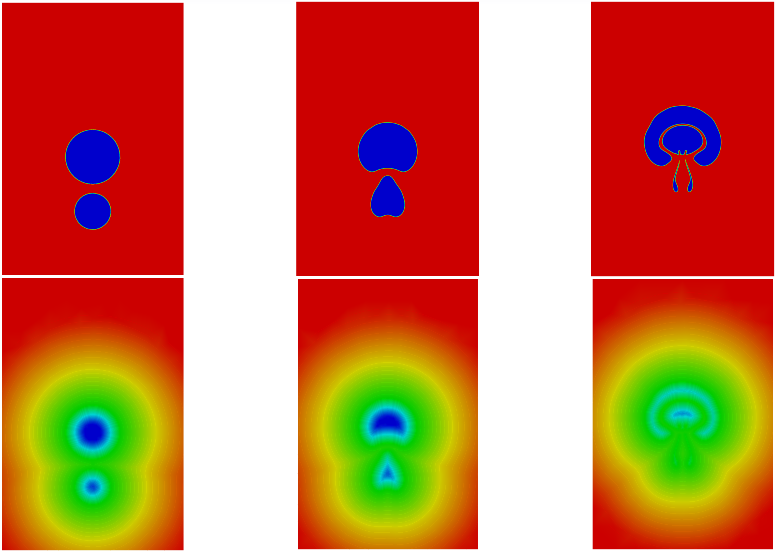
$$\Omega(t^n) = \left\{ x \in D, \;\; \phi(t^n, x) < 0 \right\},$$

  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:** $\Omega(t)$ is never discretized explicitly (with a mesh); hence, several operations may prove difficult, e.g.
  - The calculation of integrals on Ω or Γ.
  - Physical PDE on $\Omega(t)$, which are often the building blocks of the velocity field $V(t,x)$ have to be approximated by equations on D.

The "classical" practice of the level set method (II)



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

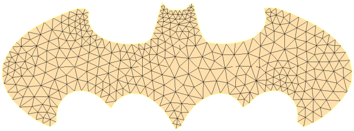
A few definitions about meshes (I)

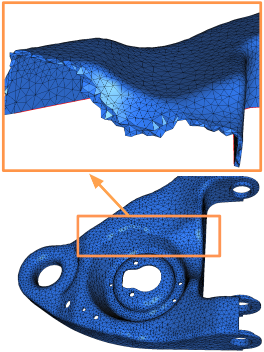
Let $\Omega \subset \mathbb{R}^d$ ($d = 2$ or 3) be a polyhedral domain.

Definition 3.

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

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

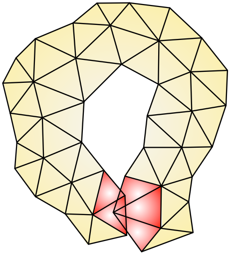




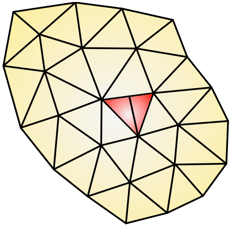
A few definitions about meshes (II)

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

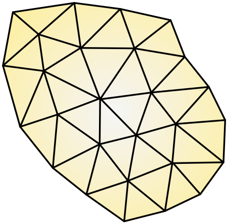
- Valid: the open simplices T_i are mutually disjoint: $T_i \cap T_j = \emptyset$ when $i \neq j$;
- Conforming: each intersection $\overline{T_i} \cap \overline{T_j}$, $i \neq j$ is reduced to either a vertex, an edge, or a face of the mesh.



Overlapping elements in an invalid mesh



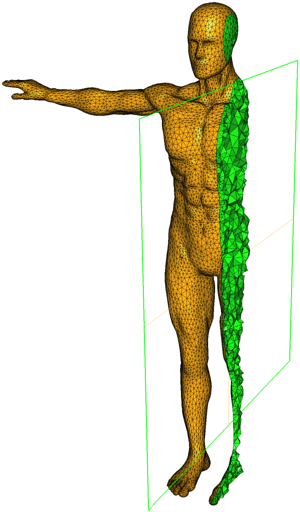
A non conforming mesh



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

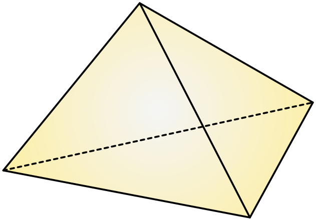
- The accuracy of most numerical methods using $\mathcal{T}$ as computational support (e.g. finite element computations) crucially depends on the quality of the simplices $T \in \mathcal{T}$.

- 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.

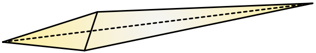
- In practice, $\mathcal{Q}(T)$ should "smoothly" discriminate "good", "bad" and "not so good" simplices T.

- 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 $\partial\Omega$, e.g.:

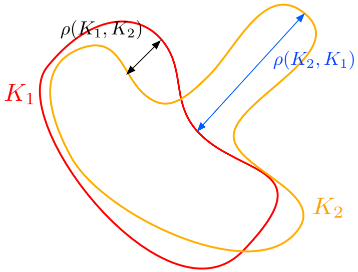
$$d^H(\mathcal{S}_{\mathcal{T}}, \partial\Omega) \leq \varepsilon,$$

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

Definition 4.

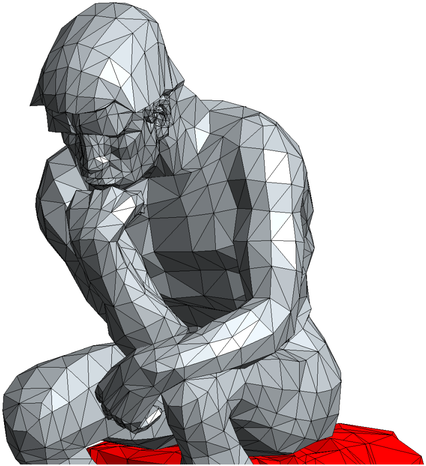
The Hausdorff distance $d^H(K_1, K_2)$ between two compact subsets $K_1, K_2 \subset \mathbb{R}^d$ is:

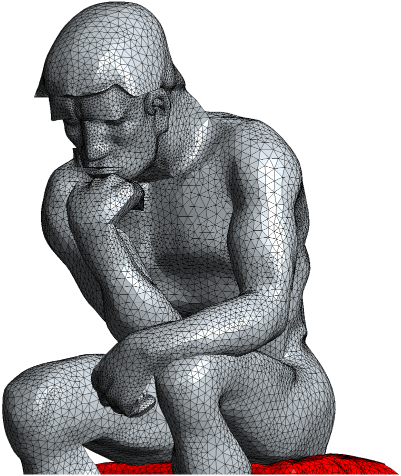
$$d^H(K_1, K_2) = \max(\rho(K_1, K_2), \rho(K_2, K_1)), \text{ where } \rho(K_1, K_2) := \max_{x \in K_1} d(x, K_2).$$



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

Quality of the geometric approximation (II)

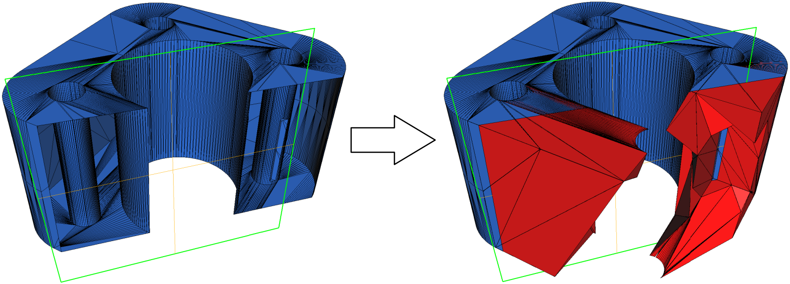




(Left) Rough approximation of a domain $\Omega \subset \mathbb{R}^3$; (right) fine geometric approximation of Ω.

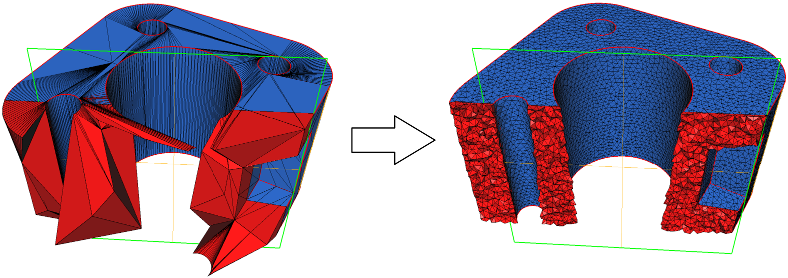
Meshing vs. remeshing (I)

- 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 $\mathcal{S}_{\mathcal{T}} = \mathcal{S}$.



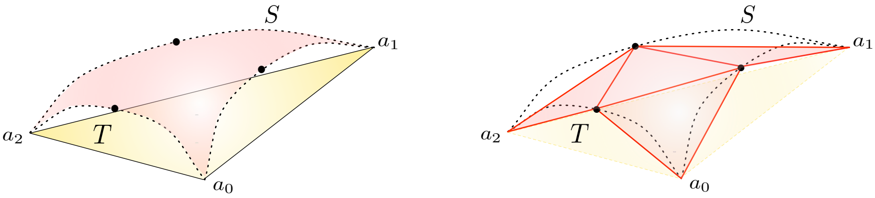
Meshing vs. remeshing (II)

- 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.

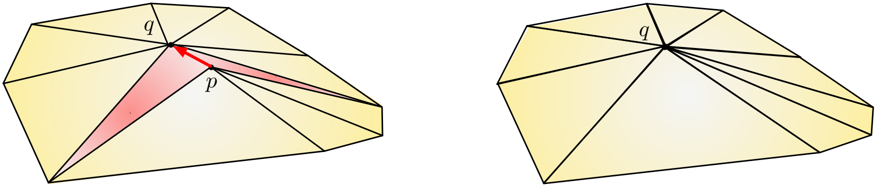


Meshing vs. remeshing (III)

Remeshing hinges on the intertwinement of four local operators, which are carefully driven to improve the mesh quality, and its geometric approximation capability.



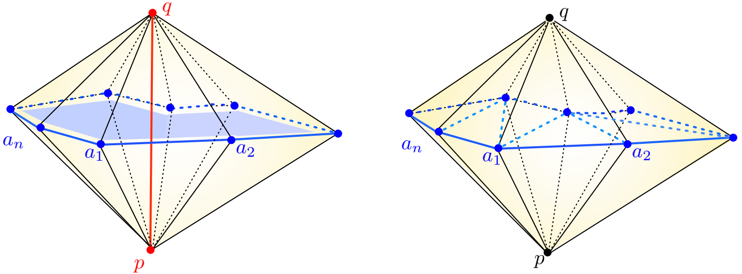
Splitting of the three edges of a surface triangle $T \in \mathcal{S}_{\mathcal{T}}$, positioning the new points on the ideal surface S.



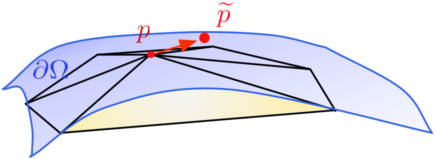
Collapse of point p over q in a surface configuration.

Meshing vs. remeshing (III)

Remeshing hinges on the intertwinement of four local operators, which are carefully driven to improve the mesh quality, and its geometric approximation capability.



3d edge swap.



Relocation of node $p \in \mathcal{S}_{\mathcal{T}}$.

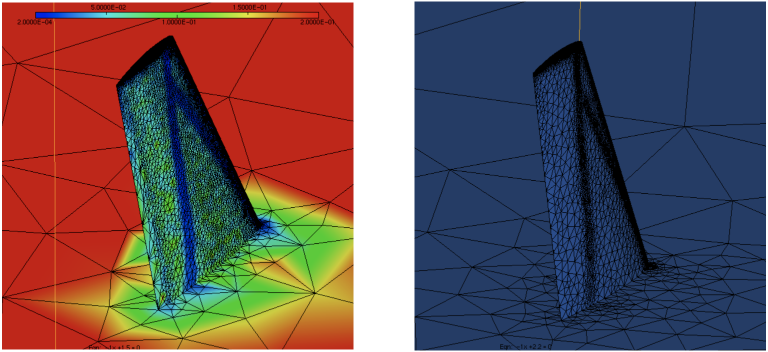
Remeshing with respect to a local size prescription (I)

Remeshing is often guided by a local size prescription, which may stem from

- A priori or a posteriori finite element estimators;
- Geometric approximation requirements of Ω.

When the size prescription is isotropic, it is encoded in a size map $h : \Omega \to \mathbb{R}$:

$$\text{For each } p \in \Omega, \quad h(p) = \text{desired size for the edges near } p.$$



(Left) Size map h defined at the vertices of a mesh; (right) modified mesh.

Remeshing with respect to a local size prescription (II)

- An anisotropic size prescription can be encoded in a metric tensor $M : \Omega \rightarrow \mathbb{R}^{d \times d}$ [VaHeMan].
- **Rationale:** The length $\ell_M(e)$ of an edge $e = pq$ with respect to M is defined by:

$$\ell_M(e) = \int_0^1 \sqrt{M(p + t(q-p))(q-p) \cdot (q-p)} \, \mathrm{d}t.$$

- Introducing the eigenvalue decomposition of $M(p)$

$$M(p) = O(p) \begin{pmatrix} d_1(p) & 0 & 0 \\ 0 & d_2(p) & 0 \\ 0 & 0 & d_3(p) \end{pmatrix} O(p)^T,$$

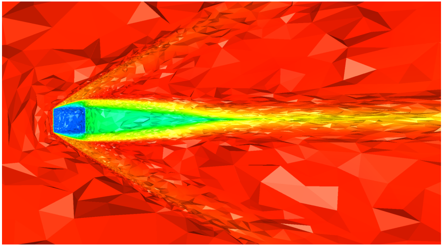
the quantities $d_i(p)$ and $O(p)$ are defined so that:

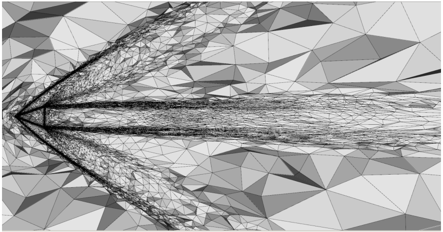
$$d_i(p) = \frac{1}{h_i^2(p)}, h_i(p) = \text{ desired size in the direction of the } i^{\text{th}} \text{ column vector } O_i(p).$$

- The mesh $\mathcal{T}$ then fulfills the size prescription if

$$\forall \text{ edge } \; e \in \mathcal{T}, \quad \ell_M(e) \approx 1.$$

Remeshing with respect to a local size prescription (III)





(Left) velocity field of a supersonic flow; (right) adapted mesh.

A(nother) word of advertisement

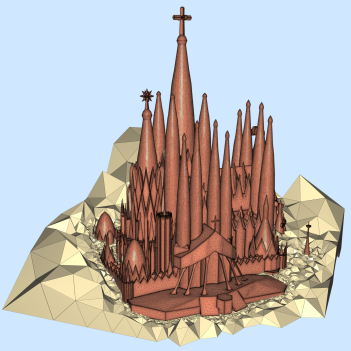
Mmg PLATFORM
Robust, Open-source & Multidisciplinary Software for Remeshing

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



Refinement of the mesh near the 0 level set of a function (I)

- Let $D \subset \mathbb{R}^d$ be a hold-all domain equipped with a simplicial mesh $\mathcal{T}$ and let $\Omega \subset D$ be smooth.
- Let $\phi : D \to \mathbb{R}$ be a smooth level set function for Ω.
- We wish to adapt $\mathcal{T}$ to the 0 level set

$$\Gamma = \{x \in D,\ \ \phi(x) = 0\}.$$

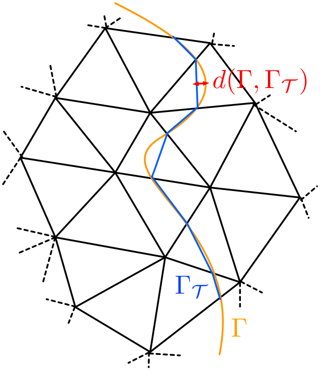
- More precisely, let $\phi_{\mathcal{T}}$ be the $\mathbb{P}_1$ finite element interpolate of ϕ, and

$$\Gamma_{\mathcal{T}} := \left\{x \in D,\ \ \phi_{\mathcal{T}} = 0\right\}.$$

We aim that:

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

where $\varepsilon > 0$ is a user-defined tolerance.



Refinement of the mesh near the 0 level set of a function (II)

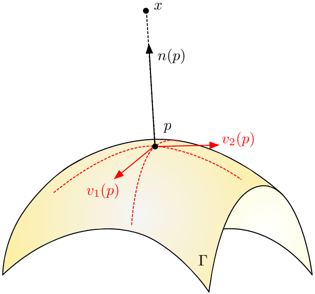
- Let $p_\Gamma(x)$ be the projection of a point $x \in D$ onto Γ.

- For $p \in \Gamma$, $v_1(p)$, $v_2(p)$ (resp. $\kappa_1(p)$, $\kappa_2(p)$) are the principal directions (resp. curvatures) of Γ at p.

- The metric tensor $M(x)$ is defined by

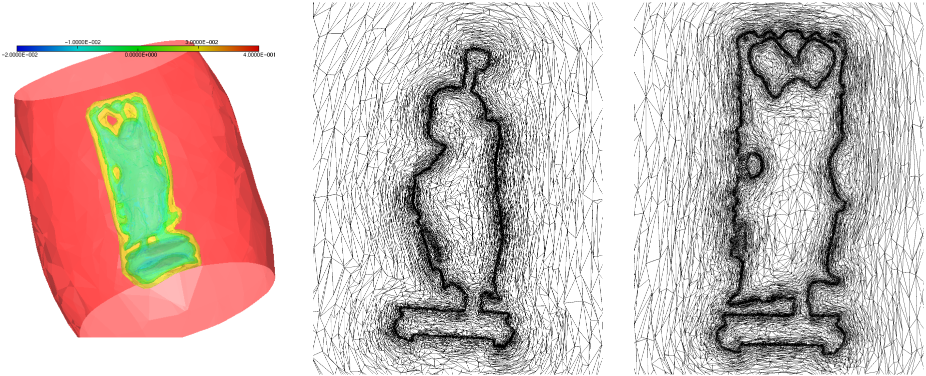
$$M(x) = \begin{pmatrix} \frac{\kappa_1(p)}{\varepsilon} & 0 & 0 \\ 0 & \frac{\kappa_2(p)}{\varepsilon} & 0 \\ 0 & 0 & \frac{1}{h_{\min}^2} \end{pmatrix}, \quad p \equiv p_\Gamma(x)$$

in the local orthonormal frame $(v_1(p), v_2(p), n(p))$.



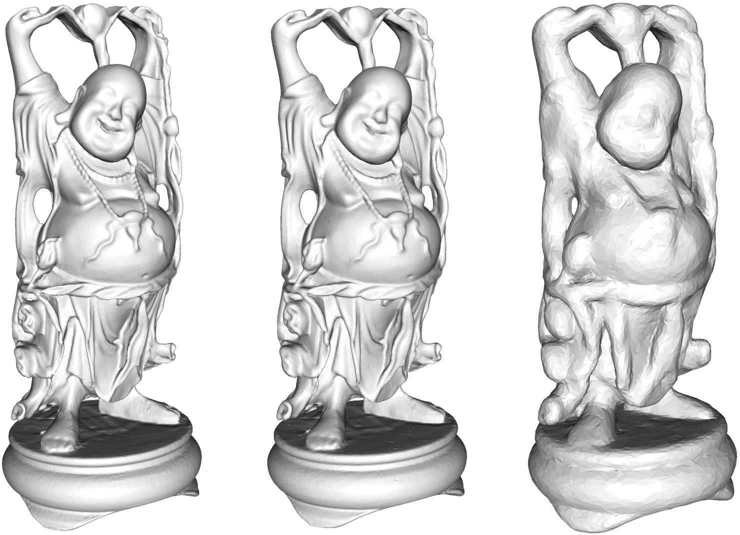
The principal curvatures of Γ at p satisfy $\kappa_2(p) < \kappa_1(p)$.

Refinement of the mesh near the 0 level set of a function (III)



Computation of the signed distance function to the Stanford "Happy Buddha" ; (left) isovalues of the signed distance function, (middle, right) two cuts in the adapted mesh.

Refinement of the mesh near the 0 level set of a function (IV)

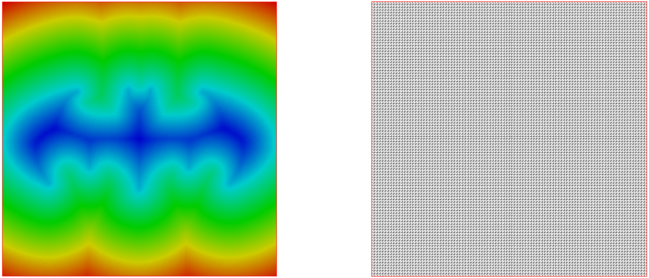


(left) The initial Buddha (middle) isovalue 0.001 *of the computed signed distance function right isovalue* 0.01.

Isosurface discretization

- In many applications of interest, the 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 $\mathcal{T}$.
- We wish to construct a surface mesh of the 0 level set Γ of ϕ, or a volume mesh of the negative subdomain Ω:

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



(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)

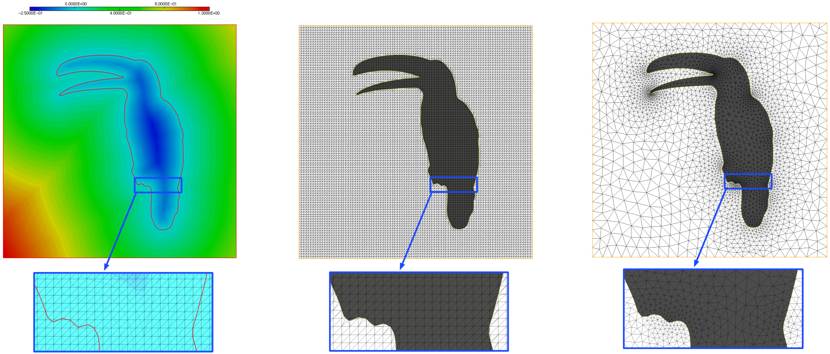
A two-step solution:

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.

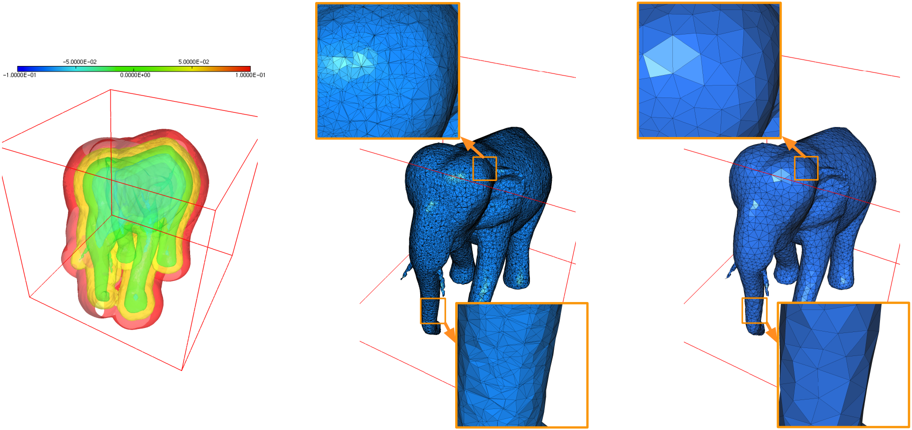
2. Improve the quality of $\mathcal{T}_{\text{temp}}$ by remeshing, to obtain $\widetilde{\mathcal{T}}$.

$\Rightarrow$ 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 $\mathcal{T}_{temp}$.

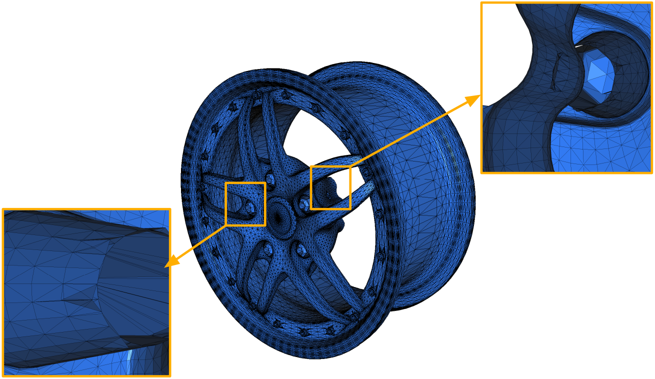
Isosurface discretization (III)



(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.

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.

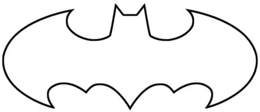


An invalid surface mesh of a domain Ω.

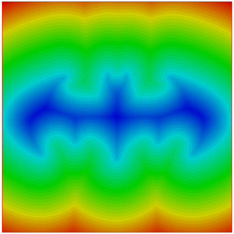
Volume mesh generation from an invalid surface triangulation (II)

One possible solution:

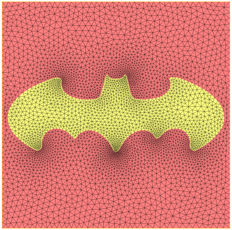
1. Calculate the signed distance function d_Ω to Ω, at the vertices of a mesh $\mathcal{T}$ of a larger, computational box D.
   - This calculation is possible even if the surface mesh $\mathcal{S}$ is invalid.
   - The mesh $\mathcal{T}$ may be adapted so that this calculation is accurate.
2. Apply 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 $\partial\Omega$.

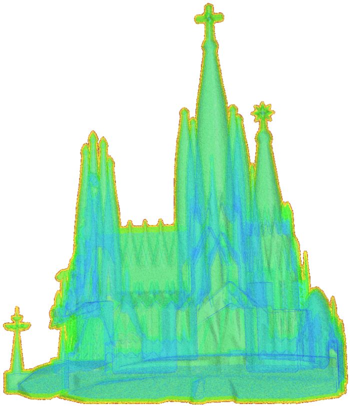


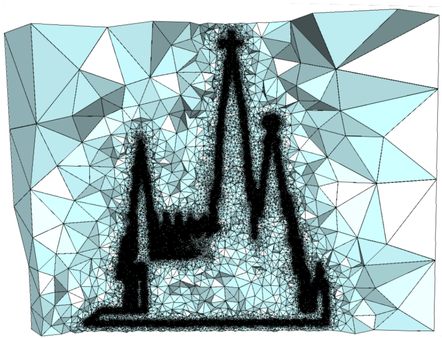
Isolines of d_Ω at the vertices of a mesh $\mathcal{T}$ of a bounding box D.



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

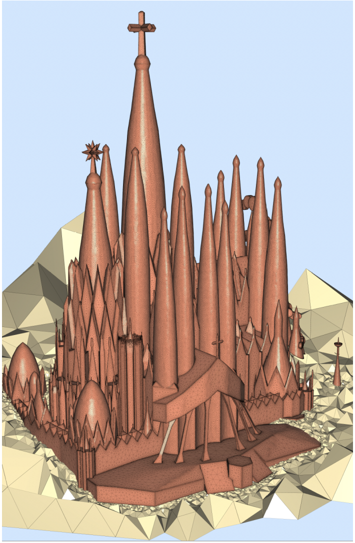
Volume mesh generation from an invalid surface triangulation (III)





(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.

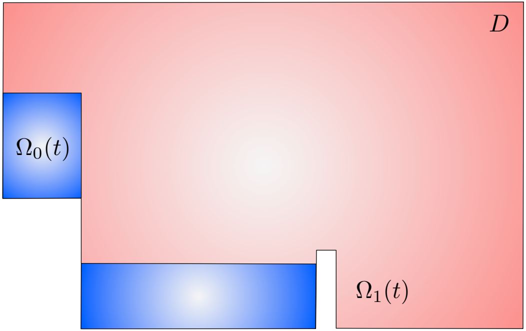
Adaptation to a size map

- In the original applications of the level set method, the computational mesh is fixed.
- 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;

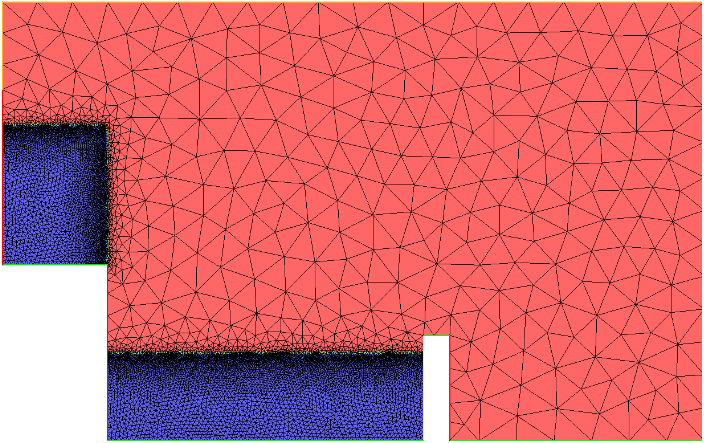
Example: bifluid flows (I)

As a result of the rupture of a dam, a water column discharges into a lower basin.

- The problem involves two complementary fluid phases $\Omega^0(t)$, $\Omega^1(t) \subset D$.
- $\Omega_0(t)$ is filled with water, $\Omega_1(t)$ is made of air.
- The velocity $V(t,x)$ of the motion is the solution to the two-phase Navier-Stokes equations.



Example: bifluid flows (II)



Evolution of a collapsing water column

A large deformation example (I)

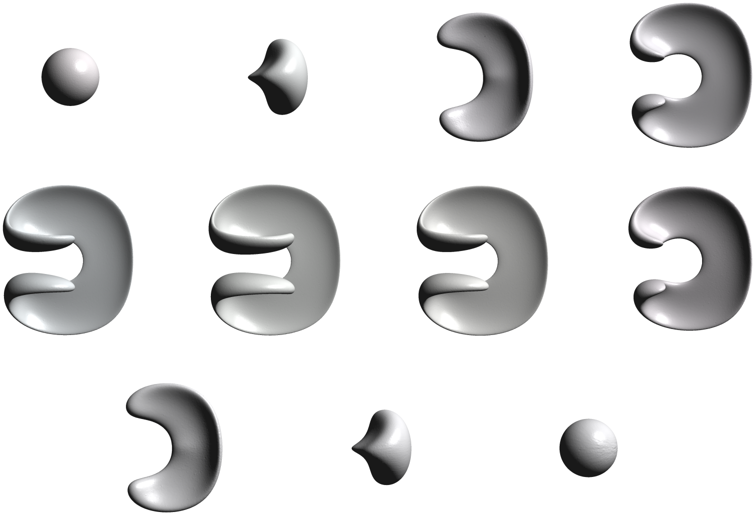
- The initial domain $\Omega(0)$ is a sphere, inside the computational domain $D = (0,1)^3$.

- The domain $\Omega(t)$ evolves according to the analytical vector field

$$V(t,x) = \left(\begin{array}{c} 2\sin^2(\pi x_1)\sin(2\pi x_2)\sin(2\pi x_3)\cos(\frac{\pi t}{T}) \\ -\sin(2\pi x_1)\sin^2(\pi x_2)\sin(2\pi x_3)\cos(\frac{\pi t}{T}) \\ -\sin(2\pi x_1)\sin(2\pi x_2)\sin^2(\pi x_3)\cos(\frac{\pi t}{T}) \end{array} \right),$$

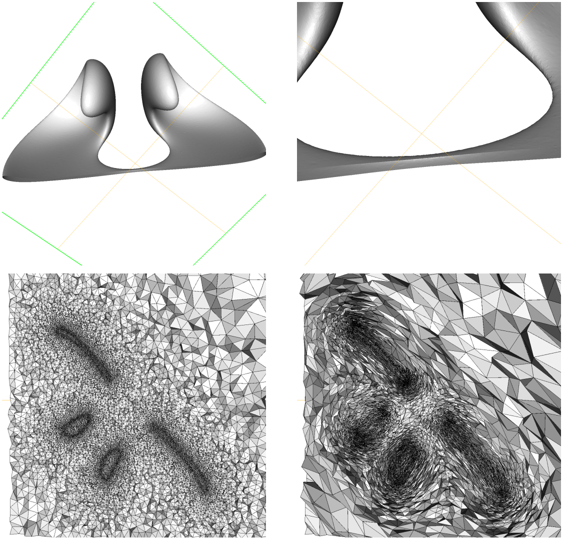
which causes an extreme stretching of $\Gamma(t)$ at $t = T/2$, before returning to the initial configuration.

A large deformation example (II)



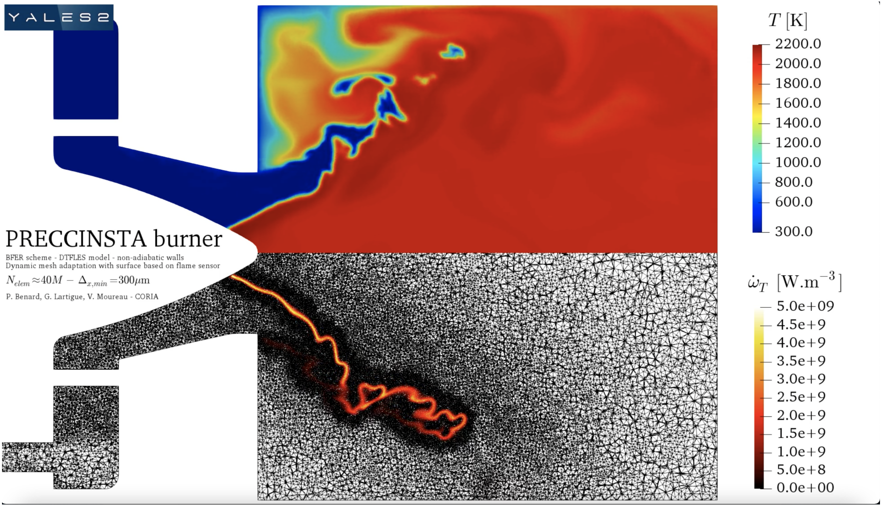
3d deformation test case : sequence of computed surfaces.

A large deformation example (III)



(Top) Cut in the interface at $t = 1.5$*; (bottom) Computational mesh using (left) isotropic (*$\approx 1,500,000$ *points), (right) anisotropic mesh adaptation (*$\approx 700,000$ *points).*

A large-scale example



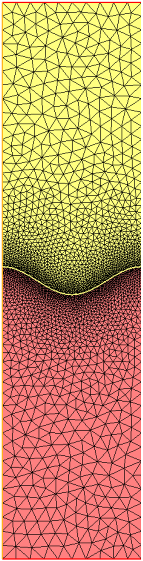
Numerical simulation of an aeronautical burner using the `Yales2` *library [Yales2].*

Body-fitted interface tracking (I)

- The numerical realization of the motion of a shape needs to reconcile the antagonistic needs to
  - account for dramatic evolutions (including topological changes) ,
  - Enjoy a mesh of the domain $\Omega(t)$ e.g. to solve PDE on $\Omega(t)$.

- "Classical" numerical methods (e.g. acting by mesh deformation) are not robust enough to handle large shape deformations.

- The level set method with the isosurface discretization operation make this possible.



Body-fitted interface tracking (II)

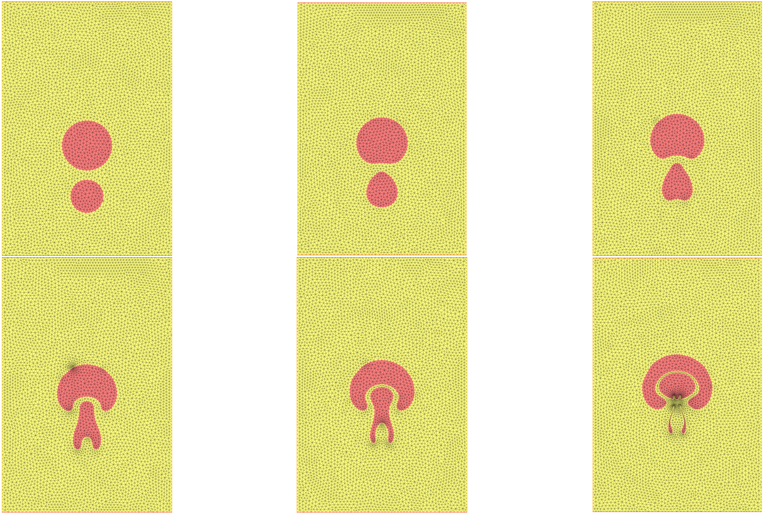
- **Initialization:** Mesh $\mathcal{T}^0$ of the computational D in which $\Omega(0)$ is explicitly discretized.
- **For** $n = 0, \ldots$ **convergence:**

  1. Calculate the signed distance function d_{Ω^n} to Ω^n on $\mathcal{T}^n$.
  2. Calculate the velocity field V^n on $\mathcal{T}^n$;
  3. Solve the level set advection equation

$$\begin{cases} \frac{\partial \phi}{\partial t}(t,x) + V^n(x) \cdot \nabla \phi(t,x) = 0 & \text{on } (0,T) \times D, \\ \phi(0,.) = \phi^n & \text{on } \mathbb{R}^d, \end{cases}$$

     on the mesh $\mathcal{T}^n$ and take $\phi^{n+1} = \phi(\Delta t, \cdot)$.
  4. Discretize the new domain Ω^{n+1} in the mesh $\mathcal{T}^n$ to obtain the new mesh $\mathcal{T}^{n+1}$.

Body-fitted interface tracking (III)



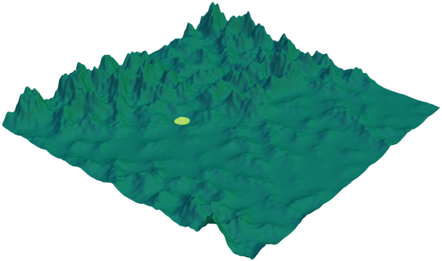
Evolution of the rising bubble by using the combination of the level set method with isosurface discretization.

Example: wild fire propagation (I)

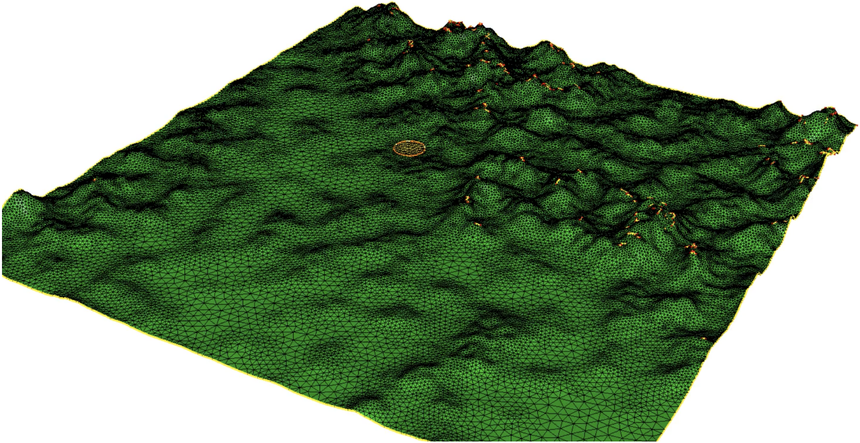
- The ambient medium is a surface $S \subset \mathbb{R}^3$, representing a land topography.

- We study the evolution of a burnt region $\Omega(t)$.

- $\Omega(t)$ evolves according to a velocity field $V(t,x)$ depending on
  - The geometry of $\Omega(t)$ (normal vector, curvature);
  - The geometry of S (slope, ...);
  - External factors (wind, ...)



Example: wild fire propagation (II)



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

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:

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

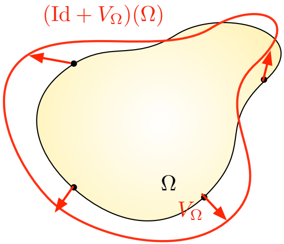
where

  - $J(\Omega)$ 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.
  - $\mathcal{U}_{\text{ad}}$ 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:

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



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

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

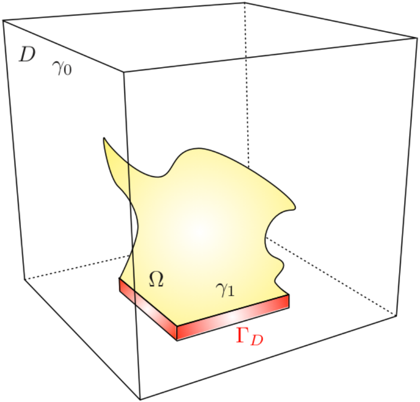
evolves by decreasing the criterion $J(\Omega)$.

Optimization of the shape of a heat diffuser (I)

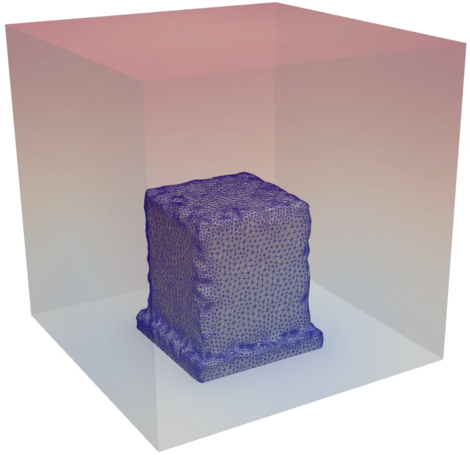
- A thermal chamber D is divided into
  - A phase Ω with high conductivity γ_1
  - A phase $D \setminus \overline{\Omega}$ with low conductivity γ_0.
- A temperature $T_0 = 0$ is imposed on Γ_D and the remaining boundary $\partial D \setminus \overline{\Gamma_D}$ is insulated from the outside.
- A heat 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)



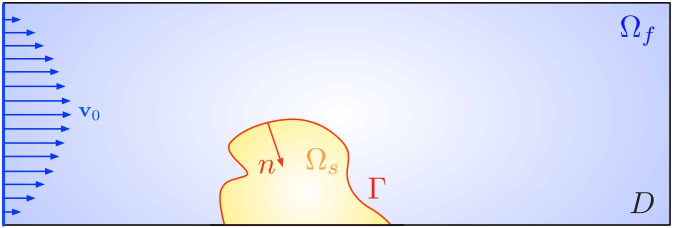
Optimization of the shape of a heat diffuser, from [FeAlDaJo].

An advanced example in fluid-structure interaction (I)

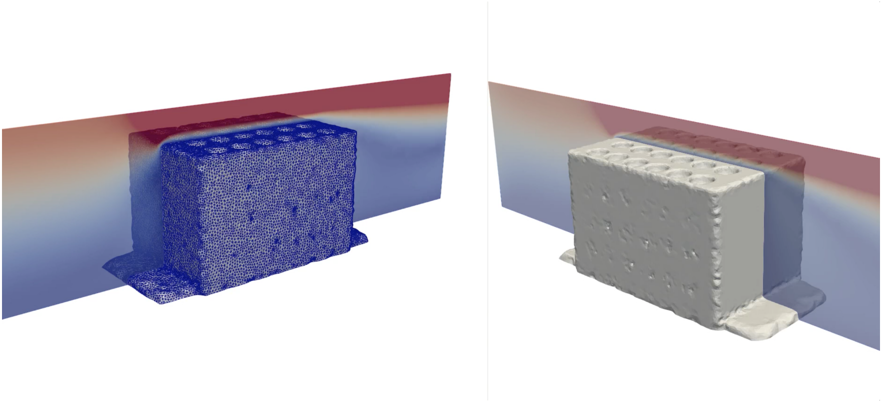
- A solid obstacle $\Omega_s := \Omega$ is placed inside a fixed cavity D where a fluid is flowing, occupying the phase $\Omega_f := D \setminus \Omega_s$.
- The fluid obeys the Navier-Stokes equations ($Re = 60$), and the solid is governed by the linearized elasticity system.
- Weak coupling between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

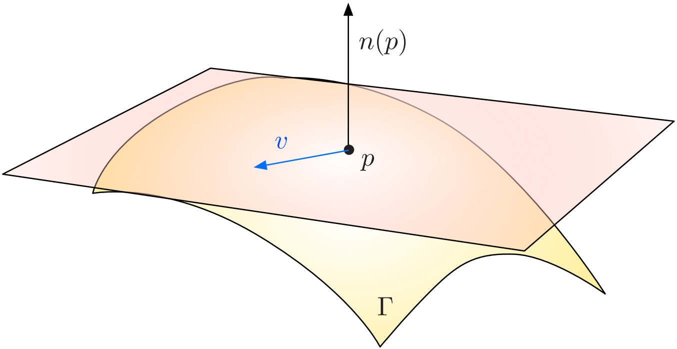
Thank you for your attention !

Technical appendix

Surfaces and curvature (I)

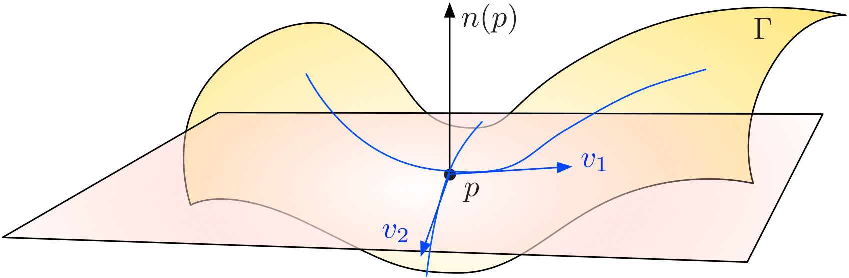
At first order, in the neighborhood of a point $p \in \Gamma$, a surface Γ behaves like a plane, the tangent plane,

- With normal vector $n(p)$,
- Which contains the tangential directions to Γ.



Surfaces and curvature (II)

- At second order in the neighborhood of $p \in \Gamma$, the surface Γ has one curvature in each tangential direction.
- The principal directions at p are those tangential directions $v_1(p)$ et $v_2(p)$ associated to the lower and larger curvatures $\kappa_1(p)$ et $\kappa_2(p)$.
- The mean curvature $\kappa(p)$ is the sum $\kappa(p) = \kappa_1(p) + \kappa_2(p)$.

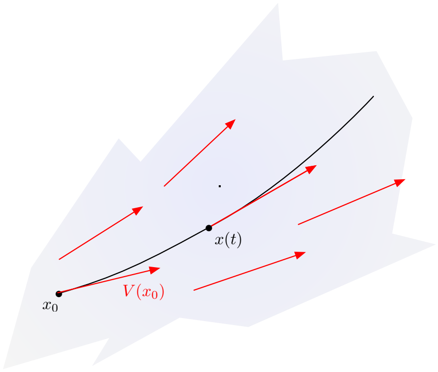


Runge-Kutta integration of dynamical systems (I)

Let $V : \mathbb{R}^d \to \mathbb{R}^d$ be a (smooth) vector field; we consider the dynamical system

$$\left\{ \begin{array}{rl} x'(t) = V(x(t)) & \text{for } t \in (0,T), \\ x(0) = x_0, & \end{array} \right.$$

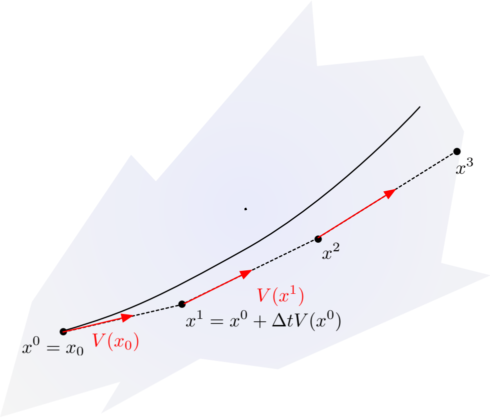
for the trajectory $t \mapsto x(t)$ of a particle with velocity V.



Introducing a subdivision $t^n = n\Delta t$ of $(0,T)$, $n = 0, \ldots, N := T/\Delta t$, we aim to calculate an approximation x^n of $x(t^n)$.

The first-order, explicit Euler approximation of this dynamical system reads:

$$\begin{cases} x^{n+1} = x^n + \Delta t V(x^n) \quad \text{for } n = 0, \dots, N-1, \\ x^0 = x_0. \end{cases}$$



This method is only first-order accurate as $\Delta t \to 0$:

$$\forall n \in 0, \dots, N, \quad |x(t^n) - x^n| \leq C\Delta t \text{ for some constant } C > 0.$$

Runge-Kutta integration of dynamical systems (III)

According to the Runge-Kutta 2 method, the iterate x^{n+1} is obtained from x^n by:

1. An attempt step is performed with the 1st-order Euler method:

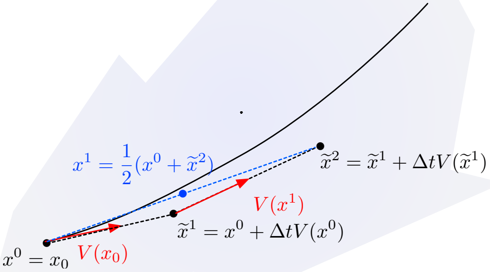
$$\widetilde{x}^{n+1} := x^n + \Delta t V(x^n).$$

2. Another attempt step is performed from $\widetilde{x}^{n+1}$:

$$\widetilde{x}^{n+2} := \widetilde{x}^{n+1} + \Delta t V(\widetilde{x}^{n+1}).$$

3. The point x^{n+1} is obtained by averaging:

$$x^{n+1} = \frac{1}{2}(x^n + \widetilde{x}^{n+2}).$$



This method is second-order accurate:

$$\forall n = 0, \ldots, N, \quad |x(t^n) - x^n| \leq C\Delta t^2 \text{ for some constant } C > 0.$$

References I

[Abgrall] R. Abgrall, *Numerical Discretization of the First-Order Hamilton-Jacobi Equation on Triangular Meshes*, Com. Pure Applied Math. XLIX, (1996) pp. 1339–1373.

[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.

[AmDa] L. Ambrosio and N. Dancer, *Geometric evolution problems, distance function and viscosity solutions*, Springer Berlin Heidelberg, (2000), pp. 5–93.

[BaSe] T.J. Barth and J.A. Sethian, *Numerical Schemes for the Hamilton–Jacobi and Level Set Equations on Triangulated Domains*, J. Comput. Phys., 145 (1998) pp. 1–40.

[BriDa] C. Brito-Pacheco and C. Dapogny, *Body-fitted tracking within a surface via a level set based mesh evolution method*, in preparation, 70(7), (2023).

References II

[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.

[Chopp] D. Chopp, *Computing minimal surfaces via level-set curvature flow*, Journal of Computational Physics, 106 (1993), pp. 77–91.

[CreRouDe] D. Cremers, M. Rousson and R. Deriche, *A review of statistical approaches to level set segmentation: integrating color, texture, motion and shape*, International journal of computer vision, 72, (2007), pp. 195–215.

[CraILi] M. G. Crandall, H. Ishii and P.-L. Lions, *User's guide to viscosity solutions of second order partial differential equations*, Bulletin of the American mathematical society, 27(1), (1992) pp. 1–67.

[CriFa] E. Cristiani and M. Falcone, *Fast semi-Lagrangian schemes for the Eikonal equation and applications*, SIAM Journal on Numerical Analysis, 45(5), (2007) pp. 1979–2011.

References III

[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.

[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.

[DerTho] A. Dervieux and F. Thomasset, *A finite element method for the simulation of a Rayleigh-Taylor instability*, in proc. of the IUTAM symposium, (1979), pp. 145–158.

[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).

References IV

[Giga] Y. Giga, *Surface Evolution Equations, a Level Set Approach*, Monographs in Mathematics, 99. Birkhäuser, Basel-Boston-Berlin, (2006).

[Grayson] M.A. Grayson, *The heat equation shrinks embedded plane curves to round points*, J. Differential Geometry, 26, (1987), pp. 285–314.

[VaHeMan] M.-G. 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.

[JeBruMai] M. Jedouaa, C.-H. Bruneau and E. Maitre, *An efficient interface capturing method for a large collection of interacting bodies immersed in a fluid*, J. Comput. Phys., 378, (2019), pp. 143–177.

[KiSe] R. Kimmel and J.A. Sethian, *Computing geodesic paths on manifolds*, Proceedings of the national academy of Sciences, 95(15), (1998), pp.8431–8435.

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

References V

- [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
- [OShu] S. J. Osher and C.-W. Shu, *High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations*, SIAM Journal on numerical analysis 28.4, (1991), pp. 907–922
- [Pironneau] O. Pironneau, *Finite element methods for fluids*, Chichester: Wiley, (1989).
- [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.
- [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).
- [Strain] J. Strain, *Semi-Lagrangian methods for level set equations*, Journal of Computational Physics, 151(2), (1999), pp. 498–533.

References VI

[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`.