Introduction to FreeFEM. Numerical shape updates

Florian Feppon

Spring 2022 – Seminar for Applied Mathematics

ETH zürich

Outline

Outline

Outline

Outline

Outline

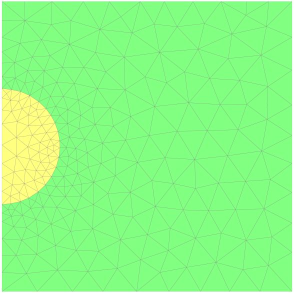
Outline

Outline

Definition 1

Let Ω be a polyhedral connected open set of $\mathbb{R}^N$. A triangular/tetrahedral mesh is a set $\mathcal{T}$ of non-degenerate N-simplices $(K_i)_{1 \leq i \leq n}$ which verify

1. $K_i \subset \bar{\Omega}$ and $\bar{\Omega} = \cup_{i=1}^{n} K_i$,

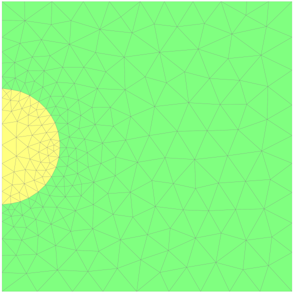


Finite Element Meshes

Definition 1

Let Ω be a polyhedral connected open set of $\mathbb{R}^N$. A triangular/tetrahedral mesh is a set $\mathcal{T}$ of non-degenerate N-simplices $(K_i)_{1\le i\le n}$ which verify

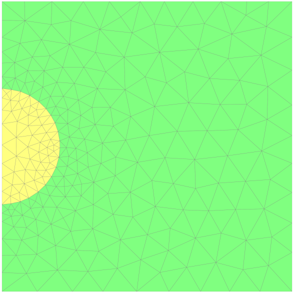
1. $K_i \subset \bar{\Omega}$ and $\bar{\Omega} = \cup_{i=1}^{n} K_i$,
2. for any $1 \le i, j \le n$, $K_i \cap K_j$ is a simplex whose vertices are also vertices of K_i and K_j.



Definition 1

Let Ω be a polyhedral connected open set of $\mathbb{R}^N$. A triangular/tetrahedral mesh is a set $\mathcal{T}$ of non-degenerate N-simplices $(K_i)_{1\leq i\leq n}$ which verify

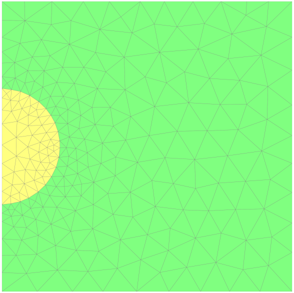
1. $K_i \subset \bar{\Omega}$ and $\bar{\Omega} = \cup_{i=1}^n K_i$,
2. for any $1 \leq i, j \leq n$, $K_i \cap K_j$ is a simplex whose vertices are also vertices of K_i and K_j.



Definition 1

Let Ω be a polyhedral connected open set of $\mathbb{R}^N$. A triangular/tetrahedral mesh is a set $\mathcal{T}$ of non-degenerate N-simplices $(K_i)_{1\leq i\leq n}$ which verify

1. $K_i \subset \bar{\Omega}$ and $\bar{\Omega} = \cup_{i=1}^{n} K_i$,
2. for any $1 \leq i, j \leq n$, $K_i \cap K_j$ is a simplex whose vertices are also vertices of K_i and K_j.



Good meshes leading to accurate FEM analysis are without “flat” triangles.

Definition 2

Given a mesh $\mathcal{T}$ with simplices $(K_i)_{1\leq i\leq n}$, we call the space of $\mathbb{P}_k$-finite elements the finite-dimensional set

$$\mathcal{V}_h := \{p \in \mathcal{C}(\bar{\Omega}) \,|\, p_{|K_i} \text{ is a polynomial of degree less than } k.\}$$

Remark 1

A $\mathbb{P}_1$ function is piecewise linear on each triangle/tetrahedron and is characterized by its values at the node of the mesh.

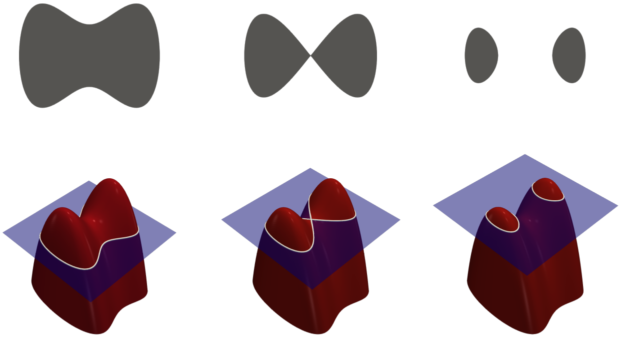
Finite Element Meshes

One of the most easiest way to generate meshes is to use **level set** functions and **remeshing**.

Level set functions

Given a computational domain D, we can represent a subset $\Omega \subset D$ as the negative subdomain of a scalar "level set" function $\phi : D \to \mathbb{R}$:

$$\begin{cases} \phi(x) < 0 & \text{If } x \in \Omega, \\ \phi(x) = 0 & \text{If } x \in \partial\Omega, \\ \phi(x) > 0 & \text{If } x \in D \backslash \Omega. \end{cases}$$



Remeshing

- It is rather difficult to mesh arbitrary domain.

Remeshing

- It is rather difficult to mesh arbitrary domain.
- It is possible to mesh domains with explicitly parameterized boundaries

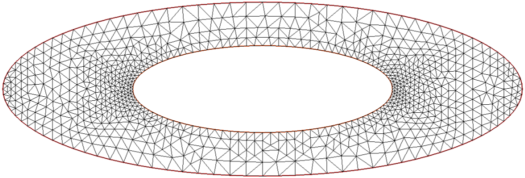


Figure: A triangular mesh with two elliptic boundaries

Remeshing

- It is rather difficult to mesh arbitrary domain.
- It is possible to mesh domains with explicitly parameterized boundaries

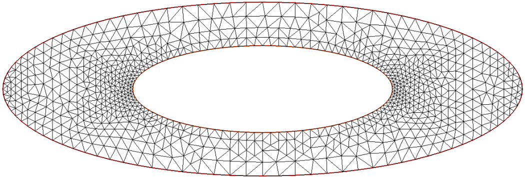


Figure: A triangular mesh with two elliptic boundaries

- It is possible to improve the “quality” of meshes using metric tensors.

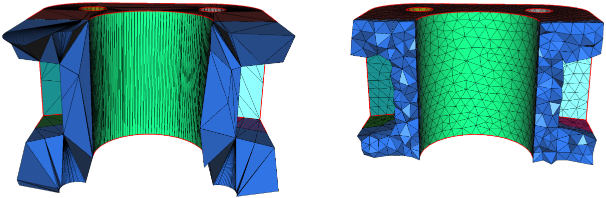


Figure: Improvement of the quality of a tetrahedral mesh (Figure from Dapogny, Dobrzynski and Frey, 2014).

Remeshing

- It is possible to mesh negative subdomains of level set functions.

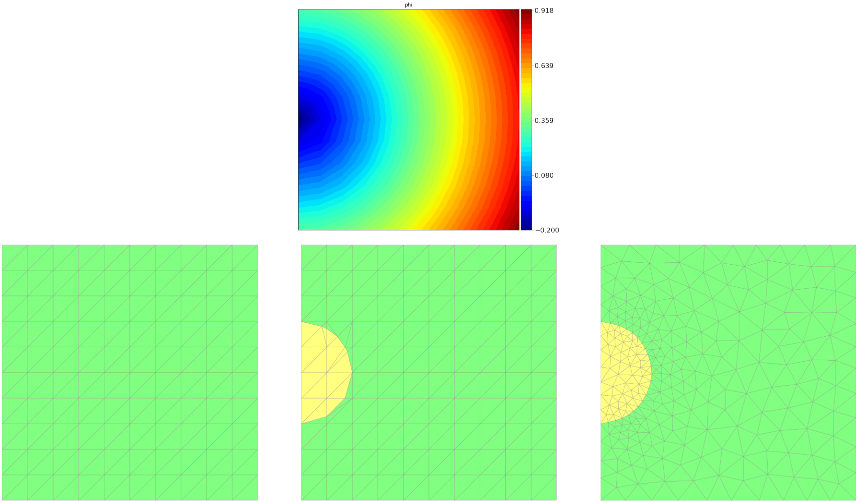


Figure: Meshing of a circular subdomain according to a level set function

Remeshing

- It is possible to mesh negative subdomains of level set functions.

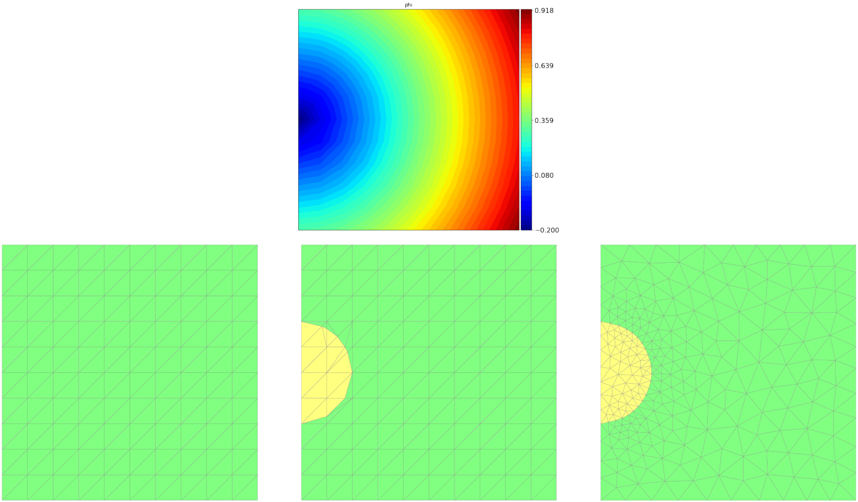


Figure: Meshing of a circular subdomain according to a level set function

We can do this using the library `Mmg`

Remeshing

Elementary operations on sets can be easily performed with level-set representations:

- $\Omega_1 \cup \Omega_2 \quad \longleftrightarrow \quad \min(\phi_1, \phi_2)$

Remeshing

Elementary operations on sets can be easily performed with level-set representations:

- $\Omega_1 \cup \Omega_2 \quad \longleftrightarrow \quad \min(\phi_1, \phi_2)$
- $\Omega_1 \cap \Omega_2 \quad \longleftrightarrow \quad \max(\phi_1, \phi_2)$

Remeshing

Elementary operations on sets can be easily performed with level-set representations:

- $\Omega_1 \cup \Omega_2 \quad \longleftrightarrow \quad \min(\phi_1, \phi_2)$
- $\Omega_1 \cap \Omega_2 \quad \longleftrightarrow \quad \max(\phi_1, \phi_2)$
- $D \backslash \Omega \quad \longleftrightarrow \quad -\phi$

Remeshing

Elementary operations on sets can be easily performed with level-set representations:

- $\Omega_1 \cup \Omega_2 \quad \longleftrightarrow \quad \min(\phi_1, \phi_2)$
- $\Omega_1 \cap \Omega_2 \quad \longleftrightarrow \quad \max(\phi_1, \phi_2)$
- $D \backslash \Omega \quad \longleftrightarrow \quad -\phi$

Remeshing

Elementary operations on sets can be easily performed with level-set representations:

- $\Omega_1 \cup \Omega_2 \quad \longleftrightarrow \quad \min(\phi_1, \phi_2)$
- $\Omega_1 \cap \Omega_2 \quad \longleftrightarrow \quad \max(\phi_1, \phi_2)$
- $D \backslash \Omega \quad \longleftrightarrow \quad -\phi$

Exercise: write the operation on level set which, given a material distribution $\Omega_f \subset D$ and a non optimizable region $\omega \subset D$, returns a new set Ω_f where the part $\Omega_f \cap \omega$ has been replaced with a new prescribed distribution $\mathcal{X}$.

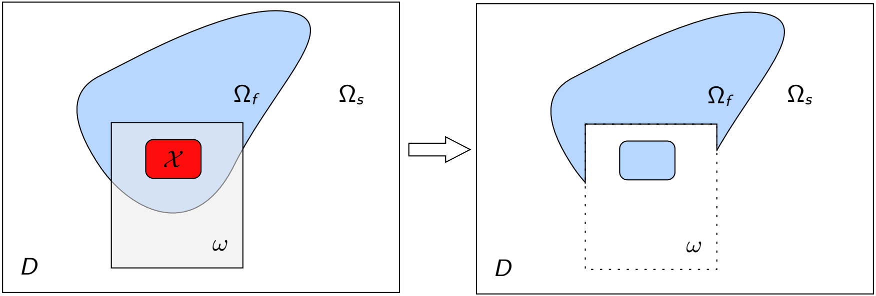


Figure: Enforcing non-optimizable regions ω: the distribution of material Ω inside the domain ω should match exactly the red set $\mathcal{X}$.

Outline

We are going to use FreeFEM to solve PDE problems.

- FreeFEM is a powerful PDE solver (developped by F. Hecht, F. Nataf, P.-H. Tournier).

We are going to use FreeFEM to solve PDE problems.

- FreeFEM is a powerful PDE solver (developped by F. Hecht, F. Nataf, P.-H. Tournier).
- It allows to solve PDEs with Finite Elements by writing their variational formulation in a spirit close to the mathematics

We are going to use FreeFEM to solve PDE problems.

- FreeFEM is a powerful PDE solver (developped by F. Hecht, F. Nataf, P.-H. Tournier).
- It allows to solve PDEs with Finite Elements by writing their variational formulation in a spirit close to the mathematics
- It allows to perform advanced operations on FEM structures

We are going to use FreeFEM to solve PDE problems.

- FreeFEM is a powerful PDE solver (developped by F. Hecht, F. Nataf, P.-H. Tournier).
- It allows to solve PDEs with Finite Elements by writing their variational formulation in a spirit close to the mathematics
- It allows to perform advanced operations on FEM structures
- It is interfaced with other powerful libraries, notably `PETSc` (linear algebra), `MPI` (parallel computing).

We are going to use FreeFEM to solve PDE problems.

- FreeFEM is a powerful PDE solver (developped by F. Hecht, F. Nataf, P.-H. Tournier).
- It allows to solve PDEs with Finite Elements by writing their variational formulation in a spirit close to the mathematics
- It allows to perform advanced operations on FEM structures
- It is interfaced with other powerful libraries, notably `PETSc` (linear algebra), `MPI` (parallel computing).
- there are regular fixes and updates

FREEFEM

We are going to use FreeFEM to solve PDE problems.

- FreeFEM is a powerful PDE solver (developped by F. Hecht, F. Nataf, P.-H. Tournier).
- It allows to solve PDEs with Finite Elements by writing their variational formulation in a spirit close to the mathematics
- It allows to perform advanced operations on FEM structures
- It is interfaced with other powerful libraries, notably `PETSc` (linear algebra), `MPI` (parallel computing).
- there are regular fixes and updates

We are going to use FreeFEM to solve PDE problems.

- FreeFEM is a powerful PDE solver (developped by F. Hecht, F. Nataf, P.-H. Tournier).
- It allows to solve PDEs with Finite Elements by writing their variational formulation in a spirit close to the mathematics
- It allows to perform advanced operations on FEM structures
- It is interfaced with other powerful libraries, notably `PETSc` (linear algebra), `MPI` (parallel computing).
- there are regular fixes and updates

FreeFEM allows in principle to solve large 3D FEM problems with millions of tetrahedral elements.

FreeFEM

Some drawbacks:

- FreeFEM might not be devoid of bugs
- the documentation is sometimes incomplete
- it is not interfaced with more communly used industrial software (using CAD descriptions)

FreeFEM installation

FreeFEM can be a bit delicate to install. Please refer to the page

`https://people.math.ethz.ch/~ffeppon/topopt_course/install_freefem.html`

for the installation !

Heat conduction problem

Let us write a solver for the heat conduction problem

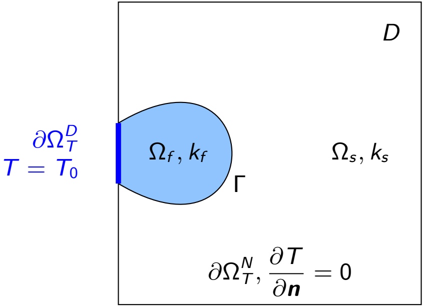


Figure: A bi-material distribution of two conductive media with conductivity k_s and k_v.

$$
\begin{cases}
-\mathrm{div}(k_f \nabla T_f) = Q_f & \text{in } \Omega_f \\
-\mathrm{div}(k_s \nabla T_s) = Q_s & \text{in } \Omega_s \\
T = T_0 & \text{on } \partial\Omega_T^D \\
-k_f \dfrac{\partial T_f}{\partial \boldsymbol{n}} = h & \text{on } \partial\Omega_T^N \cap \partial\Omega_f \\
-k_s \dfrac{\partial T_s}{\partial \boldsymbol{n}} = h & \text{on } \partial\Omega_T^N \cap \partial\Omega_s \\
T_f = T_s & \text{on } \Gamma \\
-k_f \dfrac{\partial T_f}{\partial \boldsymbol{n}} = -k_s \dfrac{\partial T_s}{\partial \boldsymbol{n}} & \text{on } \Gamma,
\end{cases}
$$

Heat conduction problem

The variational formulation reads find $T \in T_0 + V_T(\Gamma)$ such that, for any $S \in V_T(\Gamma)$,

$$\int_{\Omega_s} k_s \nabla T \cdot \nabla S \mathrm{d}x + \int_{\Omega_f} k_f \nabla T \cdot \nabla S \mathrm{d}x = \int_{\Omega_s} Q_s S \mathrm{d}x + \int_{\Omega_f} Q_f S \mathrm{d}x + \int_{\partial\Omega_T^N} hS \mathrm{d}s.$$

where

$$V_T(\Gamma) = \{S \in H^1(D) \,|\, S = 0 \text{ on } \partial\Omega_T^D\},$$

Outline

Outline

Outline

Outline

Numerical shape updates

Suppose we want to solve a shape optimization problem

$$\min_{\Omega} J(\Omega)$$

and that we need to do a Hadamard's shape update:

$$\Omega_{n+1} = (I + \boldsymbol{\theta}_n)\Omega_n$$

for a current shape Ω_n and vector field $\boldsymbol{\theta}_n \in W^{1,\infty}(D, \mathbb{R}^d)$.

Numerical shape updates

Suppose we want to solve a shape optimization problem

$$\min_{\Omega} J(\Omega)$$

and that we need to do a Hadamard's shape update:

$$\Omega_{n+1} = (I + \boldsymbol{\theta}_n)\Omega_n$$

for a current shape Ω_n and vector field $\boldsymbol{\theta}_n \in W^{1,\infty}(D, \mathbb{R}^d)$.

- If Ω_n has a mesh discretization, one can try **nodal displacements**

Numerical shape updates

Suppose we want to solve a shape optimization problem

$$\min_{\Omega} J(\Omega)$$

and that we need to do a Hadamard's shape update:

$$\Omega_{n+1} = (I + \boldsymbol{\theta}_n)\Omega_n$$

for a current shape Ω_n and vector field $\boldsymbol{\theta}_n \in W^{1,\infty}(D, \mathbb{R}^d)$.

- If Ω_n has a mesh discretization, one can try **nodal displacements**
- If Ω_n is described by a level set, one can solve an **advection equation**

Numerical shape updates

Suppose we want to solve a shape optimization problem

$$\min_{\Omega} J(\Omega)$$

and that we need to do a Hadamard's shape update:

$$\Omega_{n+1} = (I + \boldsymbol{\theta}_n)\Omega_n$$

for a current shape Ω_n and vector field $\boldsymbol{\theta}_n \in W^{1,\infty}(D, \mathbb{R}^d)$.

- If Ω_n has a mesh discretization, one can try **nodal displacements**
- If Ω_n is described by a level set, one can solve an **advection equation**
- If Ω_n is described as a meshed subdomain, then one can use a hybrid method coupling **the level set method** and **remeshing**.

Outline

Very simple algorithm:

$$x_i \leftarrow x_i + \boldsymbol{\theta}(x_i) \text{ for all nodes } x_i$$

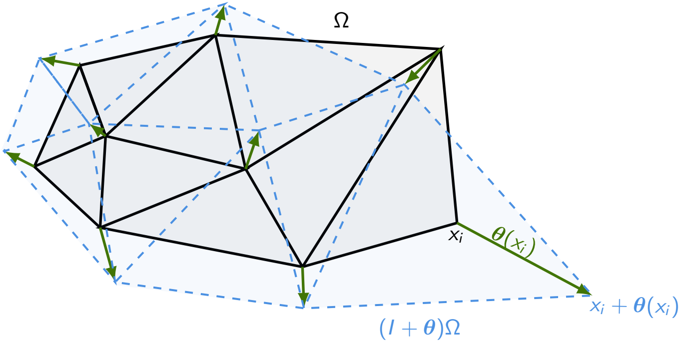


Figure: Discretization of a 2-d domain Ω in a simplicial mesh and its deformation by application of a displacement vector field $\boldsymbol{\theta}$.

Moving mesh method

- In general yields poor quality meshes.
- A refinement of the method: construct an extension $\widetilde{\boldsymbol{\theta}}$ such that

$$\begin{cases} -\mathrm{div}(A\nabla\widetilde{\boldsymbol{\theta}}) = 0 \text{ in } D, \\ \widetilde{\boldsymbol{\theta}} = \boldsymbol{\theta} \text{ on } \Gamma, \\ \widetilde{\boldsymbol{\theta}} = 0 \text{ on } \partial\Omega\backslash\Gamma \end{cases}$$

  for some positive definite matrix A, where Γ is the deformed interface and $\partial\Omega\backslash\Gamma$ are fixed interfaces.

- In general yields poor quality meshes.
- A refinement of the method: construct an extension $\widetilde{\boldsymbol{\theta}}$ such that

$$\begin{cases} -\mathrm{div}(A\nabla\widetilde{\boldsymbol{\theta}}) = 0 \text{ in } D, \\ \widetilde{\boldsymbol{\theta}} = \boldsymbol{\theta} \text{ on } \Gamma, \\ \widetilde{\boldsymbol{\theta}} = 0 \text{ on } \partial\Omega \backslash \Gamma \end{cases}$$

for some positive definite matrix A, where Γ is the deformed interface and $\partial\Omega\backslash\Gamma$ are fixed interfaces.
Then do

$$x_i \leftarrow x_i + \widetilde{\boldsymbol{\theta}}(x_i).$$

Moving mesh method

- In general yields poor quality meshes.
- A refinement of the method: construct an extension $\widetilde{\boldsymbol{\theta}}$ such that

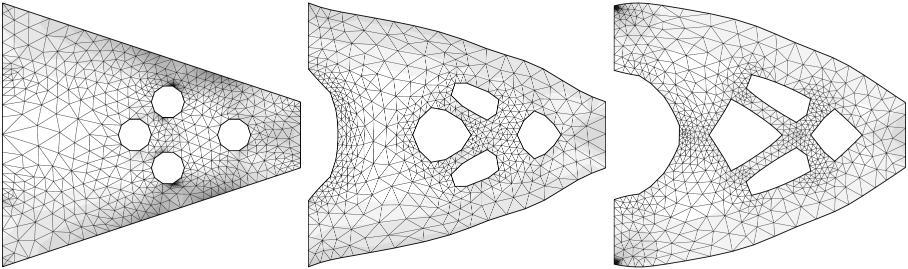
$$\begin{cases} -\mathrm{div}(A\nabla\widetilde{\boldsymbol{\theta}}) = 0 \text{ in } D, \\ \widetilde{\boldsymbol{\theta}} = \boldsymbol{\theta} \text{ on } \Gamma, \\ \widetilde{\boldsymbol{\theta}} = 0 \text{ on } \partial\Omega\backslash\Gamma \end{cases}$$

for some positive definite matrix A, where Γ is the deformed interface and $\partial\Omega\backslash\Gamma$ are fixed interfaces.
Then do

$$x_i \leftarrow x_i + \widetilde{\boldsymbol{\theta}}(x_i).$$

This method does not allow to treat topological changes and yield very poor quality meshes after a few iterations.

Moving mesh method



(a) Initial design (b) Intermediate design (c) Final design

Figure: Moving mesh method, figures from Allaire 2007.

Outline

The level set method

If the domain Ω is represented by a level set function:

$$\begin{cases} \phi(x) < 0 & \text{If } x \in \Omega, \\ \phi(x) = 0 & \text{If } x \in \partial\Omega, \\ \phi(x) > 0 & \text{If } x \in D \backslash \Omega. \end{cases}$$

then the motion of a domain $\Omega(t)$ in D according to a vector field $\boldsymbol{\theta}(x)$ can be captured by the solution $\phi(t,x)$ of the **advection equation**:

$$\begin{cases} \dfrac{\partial \phi}{\partial t}(t,x) + \boldsymbol{\theta}(x) \cdot \nabla \phi(t,x) = 0, & x \in D, \\ \phi(0,x) = \phi_0(x), & x \in D. \end{cases}$$

The level set method

If the domain Ω is represented by a level set function:

$$\begin{cases} \phi(x) < 0 & \text{If } x \in \Omega, \\ \phi(x) = 0 & \text{If } x \in \partial\Omega, \\ \phi(x) > 0 & \text{If } x \in D \backslash \Omega. \end{cases}$$

then the motion of a domain $\Omega(t)$ in D according to a vector field $\boldsymbol{\theta}(x)$ can be captured by the solution $\phi(t, x)$ of the **advection equation**:

$$\begin{cases} \frac{\partial \phi}{\partial t}(t, x) + \boldsymbol{\theta}(x) \cdot \nabla \phi(t, x) = 0, & x \in D, \\ \phi(0, x) = \phi_0(x), & x \in D. \end{cases}$$

Indeed one can show that $\phi(x(t)) = \phi_0(x_0)$ for any trajectory $x(t)$ satisfying

$$\begin{cases} \dot{x} = \boldsymbol{\theta}(x) \\ x(0) = x_0. \end{cases} \tag{1}$$

The level set method

If the domain Ω is represented by a level set function:

$$\begin{cases} \phi(x) < 0 & \text{If } x \in \Omega, \\ \phi(x) = 0 & \text{If } x \in \partial\Omega, \\ \phi(x) > 0 & \text{If } x \in D \backslash \Omega. \end{cases}$$

then the motion of a domain $\Omega(t)$ in D according to a vector field $\boldsymbol{\theta}(x)$ can be captured by the solution $\phi(t, x)$ of the **advection equation**:

$$\begin{cases} \dfrac{\partial \phi}{\partial t}(t, x) + \boldsymbol{\theta}(x) \cdot \nabla \phi(t, x) = 0, & x \in D, \\ \phi(0, x) = \phi_0(x), & x \in D. \end{cases}$$

Indeed one can show that $\phi(x(t)) = \phi_0(x_0)$ for any trajectory $x(t)$ satisfying

$$\begin{cases} \dot{x} = \boldsymbol{\theta}(x) \\ x(0) = x_0. \end{cases} \tag{1}$$

The new domain is then $\Omega(1) = \{x \,|\, \phi(1, x) < 0\} = \Phi^{\boldsymbol{\theta}}(\Omega)$ where $\Phi^{\boldsymbol{\theta}}$ is the flow map diffeomorphism associated with eq. (1).

The level set method

Remark: we don't have $\Phi^{\boldsymbol{\theta}}(\Omega) = (\boldsymbol{I} + \boldsymbol{\theta})\Omega$

Remark: we don't have $\Phi^{\boldsymbol{\theta}}(\Omega) = (I + \boldsymbol{\theta})\Omega$
However one can show that for small $\boldsymbol{\theta}$,

$$\Phi^{\boldsymbol{\theta}} = I + \boldsymbol{\theta} + O(||\boldsymbol{\theta}||^2),$$

Remark: we don't have $\Phi^{\boldsymbol{\theta}}(\Omega) = (I + \boldsymbol{\theta})\Omega$
However one can show that for small $\boldsymbol{\theta}$,

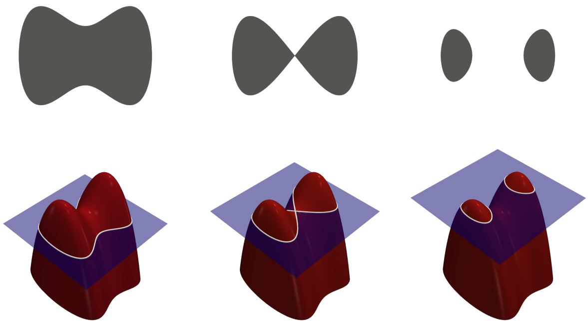
$$\Phi^{\boldsymbol{\theta}} = I + \boldsymbol{\theta} + O(||\boldsymbol{\theta}||^2),$$

hence the first order asymptotic shape calculus works with $\Phi^{\boldsymbol{\theta}}$.

The level set method

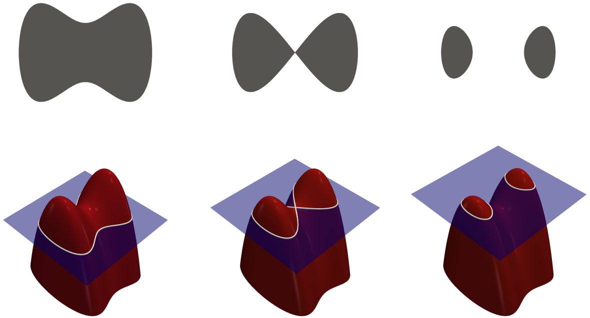
The power of the level set method:

- capture easily topological changes on **fixed meshes**:



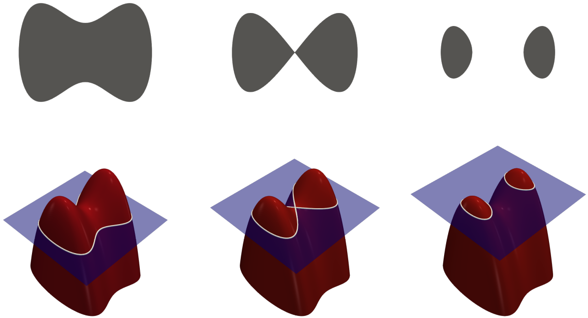
The power of the level set method:

- capture easily topological changes on **fixed meshes**:



The power of the level set method:

- capture easily topological changes on **fixed meshes**:



Its main drawback:

- one needs to use an interpolating physical model if one relies only on a fixed meshes where the physical interfaces are only captured implicitly

For instance:

- the linear elasticity equations are interpolated by using an "ersatz material" for representing void:

$$\begin{cases} -\mathrm{div}(Ae(\boldsymbol{u})) = \boldsymbol{f}_s \text{ in } \Omega_s \\ Ae(\boldsymbol{u}) \cdot \boldsymbol{n} = 0 \text{ on } \Gamma \\ \boldsymbol{u} = \boldsymbol{u}_0 \text{ on } \partial\Omega_s^D \\ Ae(\boldsymbol{u}) \cdot \boldsymbol{n} = \boldsymbol{g} \text{ on } \partial\Omega_s^N \end{cases} \longrightarrow \begin{cases} -\mathrm{div}(A(\Omega_s)e(\boldsymbol{u})) = \boldsymbol{f} \text{ in } D \\ \boldsymbol{u} = \boldsymbol{u}_0 \text{ on } \partial\Omega_s^D \\ Ae(\boldsymbol{u}) \cdot \boldsymbol{n} = \boldsymbol{g} \text{ on } \partial\Omega_s^N \end{cases}$$

with $A(\Omega_s) = A1_{\Omega_s} + \epsilon I 1_{D\setminus\Omega_s}$.

The level set method

For instance:

- the linear elasticity equations are interpolated by using an "ersatz material" for representing void:

$$\begin{cases} -\mathrm{div}(Ae(\boldsymbol{u})) = \boldsymbol{f}_s \text{ in } \Omega_s \\ Ae(\boldsymbol{u}) \cdot \boldsymbol{n} = 0 \text{ on } \Gamma \\ \boldsymbol{u} = \boldsymbol{u}_0 \text{ on } \partial\Omega_s^D \\ Ae(\boldsymbol{u}) \cdot \boldsymbol{n} = \boldsymbol{g} \text{ on } \partial\Omega_s^N \end{cases} \longrightarrow \begin{cases} -\mathrm{div}(A(\Omega_s)e(\boldsymbol{u})) = \boldsymbol{f} \text{ in } D \\ \boldsymbol{u} = \boldsymbol{u}_0 \text{ on } \partial\Omega_s^D \\ Ae(\boldsymbol{u}) \cdot \boldsymbol{n} = \boldsymbol{g} \text{ on } \partial\Omega_s^N \end{cases}$$

  with $A(\Omega_s) = A 1_{\Omega_s} + \epsilon I 1_{D \setminus \Omega_s}$.
- Physically, "void" is replaced with a very soft material.

For instance:

- the linear elasticity equations are interpolated by using an "ersatz material" for representing void:

$$\begin{cases} -\mathrm{div}(Ae(\boldsymbol{u})) = \boldsymbol{f}_s \text{ in } \Omega_s \\ Ae(\boldsymbol{u})\cdot\boldsymbol{n} = 0 \text{ on } \Gamma \\ \boldsymbol{u} = \boldsymbol{u}_0 \text{ on } \partial\Omega_s^D \\ Ae(\boldsymbol{u})\cdot\boldsymbol{n} = \boldsymbol{g} \text{ on } \partial\Omega_s^N \end{cases} \longrightarrow \begin{cases} -\mathrm{div}(A(\Omega_s)e(\boldsymbol{u})) = \boldsymbol{f} \text{ in } D \\ \boldsymbol{u} = \boldsymbol{u}_0 \text{ on } \partial\Omega_s^D \\ Ae(\boldsymbol{u})\cdot\boldsymbol{n} = \boldsymbol{g} \text{ on } \partial\Omega_s^N \end{cases}$$

  with $A(\Omega_s) = A 1_{\Omega_s} + \epsilon I 1_{D\setminus\Omega_s}$.
- Physically, "void" is replaced with a very soft material.
- Since the physical model is different, different formulas for the shape derivative must be implemented.

For instance:

- the Navier Stokes equations are interpolated by using the Brinkmann porous flow model:

$$\begin{cases} -\mathrm{div}(\sigma_f(\boldsymbol{v}, p)) + \rho \nabla \boldsymbol{v}\, \boldsymbol{v} = \boldsymbol{f}_f & \text{in } \Omega_f \\ \mathrm{div}(\boldsymbol{v}) = 0 & \text{in } \Omega_f \\ \boldsymbol{v} = \boldsymbol{v}_0 & \text{on } \partial\Omega_f^D \\ \sigma_f(\boldsymbol{v}, p)\boldsymbol{n} = 0 & \text{on } \partial\Omega_f^N \\ \boldsymbol{v} = 0 & \text{on } \Gamma, \end{cases}$$

The level set method

For instance:

- the Navier Stokes equations are interpolated by using the Brinkmann porous flow model:

$$\begin{cases} -\mathrm{div}(\sigma_f(\boldsymbol{v}, p)) + \rho \nabla \boldsymbol{v}\, \boldsymbol{v} + \alpha(\Omega_f)\boldsymbol{v} = \boldsymbol{f}_f & \text{in } D \\ \mathrm{div}(\boldsymbol{v}) = 0 & \text{in } D \\ \boldsymbol{v} = \boldsymbol{v}_0 & \text{on } \partial\Omega_f^D \\ \sigma_f(\boldsymbol{v}, p)\boldsymbol{n} = 0 & \text{on } \partial\Omega_f^N \end{cases}$$

  with $\alpha(\Omega_f) = \mathtt{tgv} 1_{D \setminus \Omega_f}$ for some large value $\mathtt{tgv}$.
- Physically, the solid material is replaced with a slightly porous material.

Further remarks:

- when using the advection equation, topological changes are the outcome of some **numerical diffusion**.

Further remarks:

- when using the advection equation, topological changes are the outcome of some **numerical diffusion**. These are not mathematically captured by the notion of shape derivatives and happen "fortunately" in this method

Further remarks:

- when using the advection equation, topological changes are the outcome of some **numerical diffusion**. These are not mathematically captured by the notion of shape derivatives and happen "fortunately" in this method
- "true topological changes" are permitted when solving the **Hamilton-Jacobi** equation:

$$\frac{\partial \phi}{\partial t} + v|\nabla \phi| = 0$$

  and considering **viscosity solutions**.

The level set method

Further remarks:

- when using the advection equation, topological changes are the outcome of some **numerical diffusion**. These are not mathematically captured by the notion of shape derivatives and happen "fortunately" in this method
- "true topological changes" are permitted when solving the **Hamilton-Jacobi** equation:

$$\frac{\partial \phi}{\partial t} + v|\nabla \phi| = 0$$

  and considering **viscosity solutions**.
  Appropriate numerical schemes are required, which are quite delicate to implement on triangular meshes.

Further remarks:

- when using the advection equation, topological changes are the outcome of some **numerical diffusion**. These are not mathematically captured by the notion of shape derivatives and happen "fortunately" in this method
- "true topological changes" are permitted when solving the **Hamilton-Jacobi** equation:

$$\frac{\partial \phi}{\partial t} + v|\nabla \phi| = 0$$

  and considering **viscosity solutions**.
  Appropriate numerical schemes are required, which are quite delicate to implement on triangular meshes.
- The **advection equation** can be solved on triangular/tetrahedral meshes with the **method of characteristics**.

The level set method

Further remarks:

- when using the advection equation, topological changes are the outcome of some **numerical diffusion**. These are not mathematically captured by the notion of shape derivatives and happen "fortunately" in this method
- "true topological changes" are permitted when solving the **Hamilton-Jacobi** equation:

$$\frac{\partial \phi}{\partial t} + v|\nabla \phi| = 0$$

  and considering **viscosity solutions**.
  Appropriate numerical schemes are required, which are quite delicate to implement on triangular meshes.
- The **advection equation** can be solved on triangular/tetrahedral meshes with the **method of characteristics**.
- A common practice is to initialize the level-set to the **signed distance function** to the domain.

The signed distance function

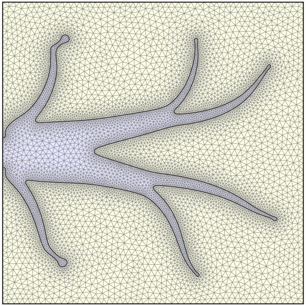
The signed distance function to a bounded open domain $\Omega \subset \mathbb{R}^d$ is the function

$$d_\Omega : \mathbb{R}^d \to \mathbb{R}$$

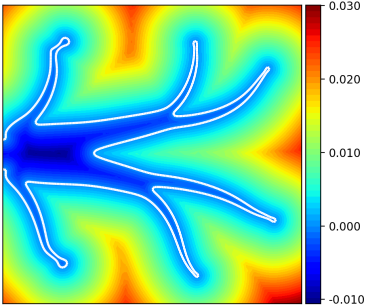
defined for any $x \in \mathbb{R}^d$ by

$$d_\Omega(x) = \begin{cases} -\inf\limits_{y \in \partial\Omega} ||x - y|| & \text{if } x \in \Omega, \\ \inf\limits_{y \in \partial\Omega} ||x - y|| & \text{if } x \notin \Omega. \end{cases}$$

The signed distance function



(a) Meshed subdomain $\Omega \subset D$ (in blue) of a computational domain D.



(b) Isocontours of the signed distance function d_Ω.

Figure: Example of signed distance function d_Ω numerically computed on a meshed domain.

The signed distance function

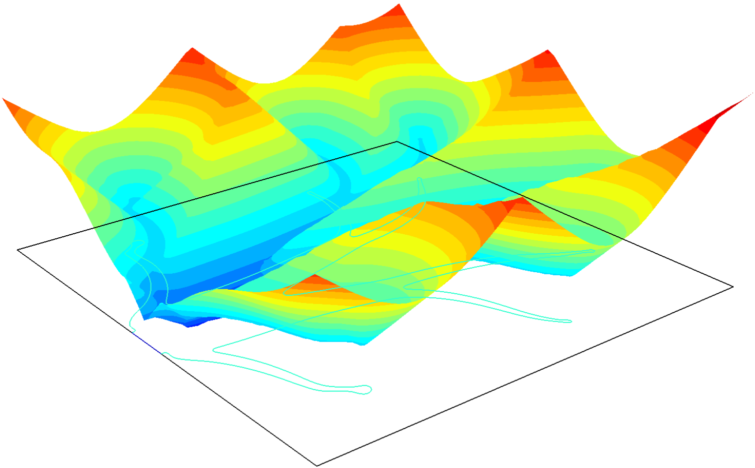


Figure: 3-d plot of d_Ω.

The signed distance function

A fundamental property: ∇d_Ω is an extension of the outward normal to d_Ω constant along the rays, in particular:

- $|\nabla d_\Omega(x)| = 1$ for any $x \in D$ where d_Ω is differentiable;

The signed distance function

A fundamental property: ∇d_Ω is an extension of the outward normal to d_Ω constant along the rays, in particular:

- $|\nabla d_\Omega(x)| = 1$ for any $x \in D$ where d_Ω is differentiable;
- $\nabla d_\Omega(y) = \boldsymbol{n}(y)$ for any $y \in \partial\Omega$.

The signed distance function

A fundamental property: ∇d_Ω is an extension of the outward normal to d_Ω constant along the rays, in particular:

- $|\nabla d_\Omega(x)| = 1$ for any $x \in D$ where d_Ω is differentiable;
- $\nabla d_\Omega(y) = \boldsymbol{n}(y)$ for any $y \in \partial\Omega$.

The signed distance function

A fundamental property: ∇d_Ω is an extension of the outward normal to d_Ω constant along the rays, in particular:

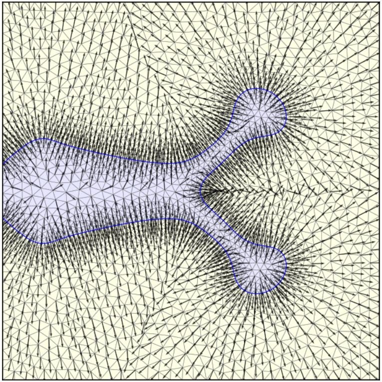
- $|\nabla d_\Omega(x)| = 1$ for any $x \in D$ where d_Ω is differentiable;
- $\nabla d_\Omega(y) = \boldsymbol{n}(y)$ for any $y \in \partial\Omega$.

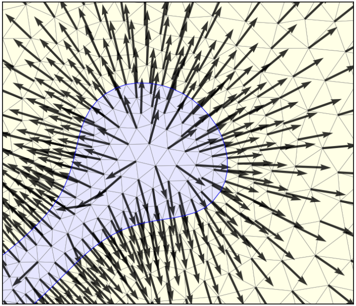
The signed distance function

A fundamental property: ∇d_Ω is an extension of the outward normal to d_Ω constant along the rays, in particular:

- $|\nabla d_\Omega(x)| = 1$ for any $x \in D$ where d_Ω is differentiable;
- $\nabla d_\Omega(y) = \boldsymbol{n}(y)$ for any $y \in \partial\Omega$.

It also holds that $\Delta d_\Omega = \kappa$ on $\partial\Omega$ where κ is the mean curvature of $\partial\Omega$.





The signed distance function

- There exist a number of numerical algorithms for computing d_Ω of a mesh subdomain (notably, the Fast Marching Method)

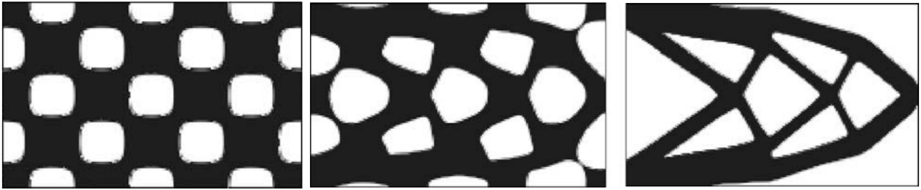
The signed distance function

- There exist a number of numerical algorithms for computing d_Ω of a mesh subdomain (notably, the Fast Marching Method)
- From a given level set function $\phi_0(x)$, solving the **reinitialization** equation

$$\begin{cases} \partial_t \phi + \mathrm{sign}(\phi)(|\nabla \phi| - 1) = 0 \\ \phi(0, x) = \phi_0(x). \end{cases}$$

allows to transform ϕ_0 into the signed distance function to the domain $\Omega = \{x \mid \phi_0(x) < 0\}$.

The signed distance function



(a) Initial design (b) Intermediate design (c) Final design

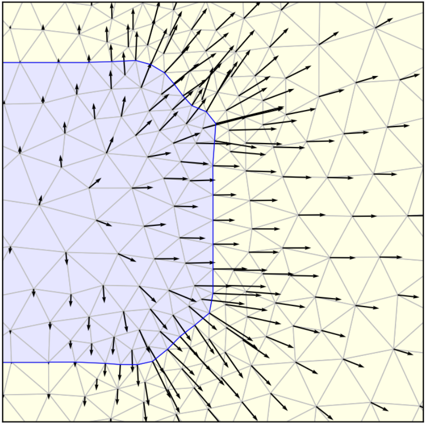
Figure: Topology optimization with the level set method

Outline

1. Finite Element Meshes – `MMg`
2. Introduction to FreeFEM
3. Numerical shape updates
   3.1 Moving mesh method
   3.2 The level set method
   3.3 Body-fitted meshes

Body fitted meshes

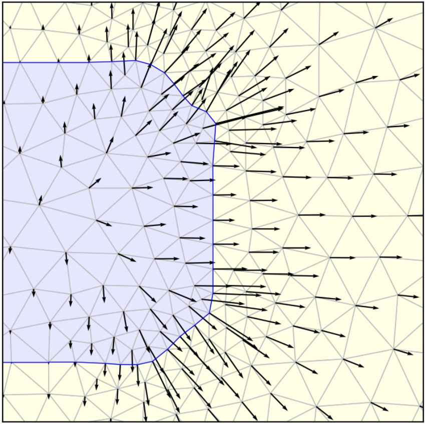
A more recent trend is to combine remeshing with the level-set method to evolve **body-fitted** meshes (Allaire et. al. 2014).



Body fitted meshes

A more recent trend is to combine remeshing with the level-set method to evolve **body-fitted** meshes (Allaire et. al. 2014).

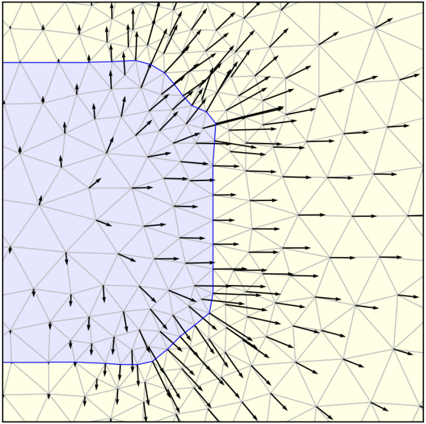
- Domain interface Γ exactly captured.



Body fitted meshes

A more recent trend is to combine remeshing with the level-set method to evolve **body-fitted** meshes (Allaire et. al. 2014).

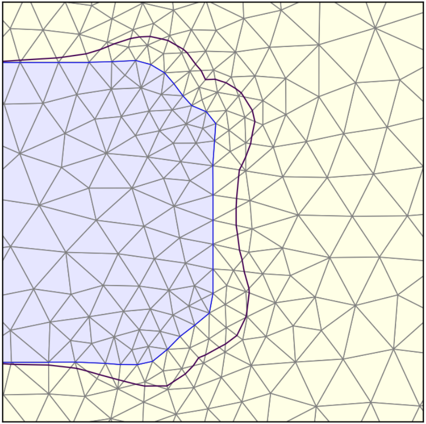
- Domain interface Γ exactly captured.
- Mesh size control is possible.



Body fitted meshes

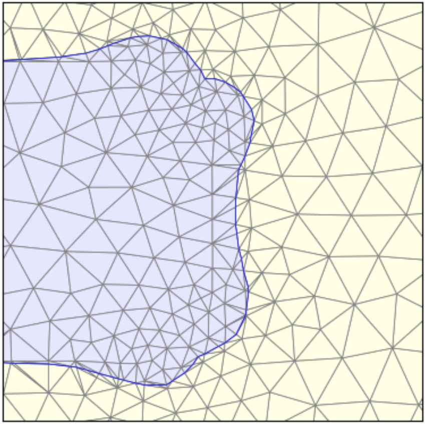
A more recent trend is to combine remeshing with the level-set method to evolve **body-fitted** meshes (Allaire et. al. 2014).

- Domain interface Γ exactly captured.
- Mesh size control is possible.



Body fitted meshes

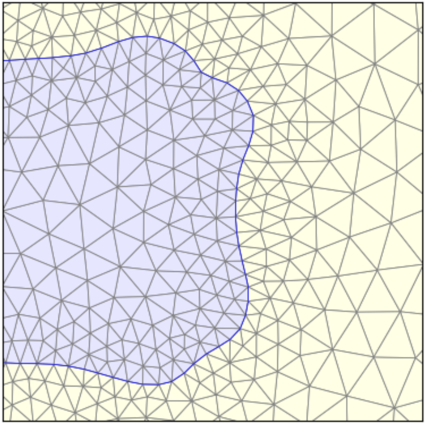
A more recent trend is to combine remeshing with the level-set method to evolve **body-fitted** meshes (Allaire et. al. 2014).



- Domain interface Γ exactly captured.
- Mesh size control is possible.

Body fitted meshes

A more recent trend is to combine remeshing with the level-set method to evolve **body-fitted** meshes (Allaire et. al. 2014).

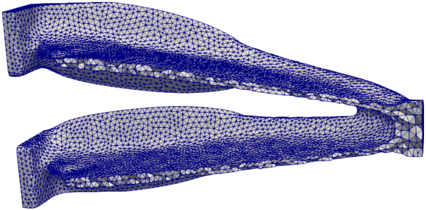


- Domain interface Γ exactly captured.
- Mesh size control is possible.

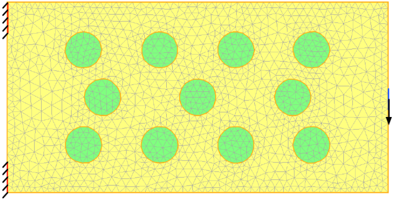
Body fitted meshes

A more recent trend is to combine remeshing with the level-set method to evolve **body-fitted** meshes (Allaire et. al. 2014).

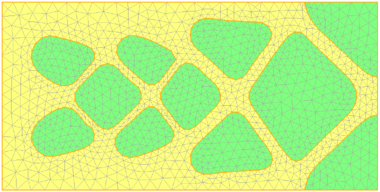
- Domain interface Γ exactly captured.
- Mesh size control is possible.
- Less prone to numerical diffusion, allows to capture fine details.



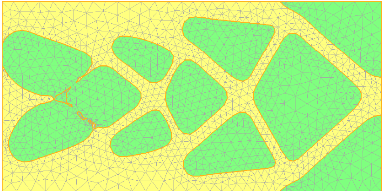
Body fitted meshes



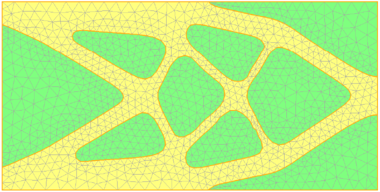
(a) Initial design



(b) Intermediate design



(c) Intermediate design featuring a topological change



(d) Final design

Figure: Level-set based mesh evolution method (figures from Dapogny et. al., 2013).