Unstructured mesh adaptation applied to CFD simulation.

Cécile Dobrzynski*

* IMB - Université de Bordeaux and
Team Cardamom - INRIA Bordeaux Sud-Ouest, Bordeaux, France

in collab. with R. Abgrall, H. Beaugendre, C. Dapogny,
P. Frey, A. Froehly, L. Nouveau, M. Ricchiuto

Guideline

Introduction

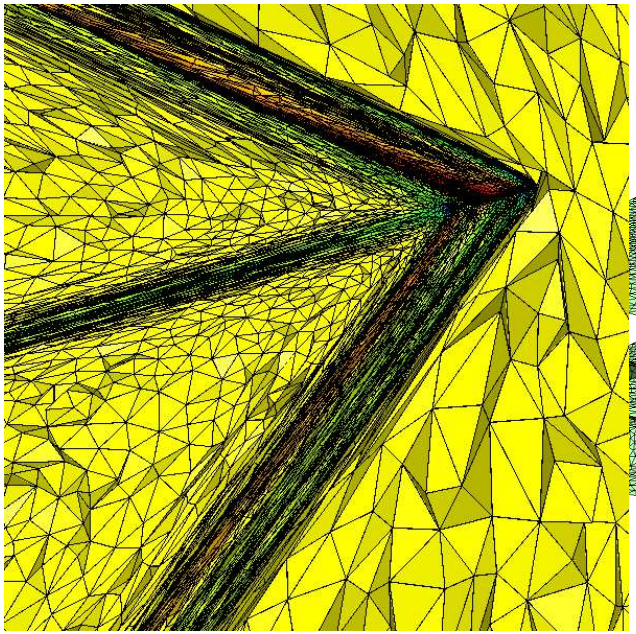
Useful tools for mesh adaptation

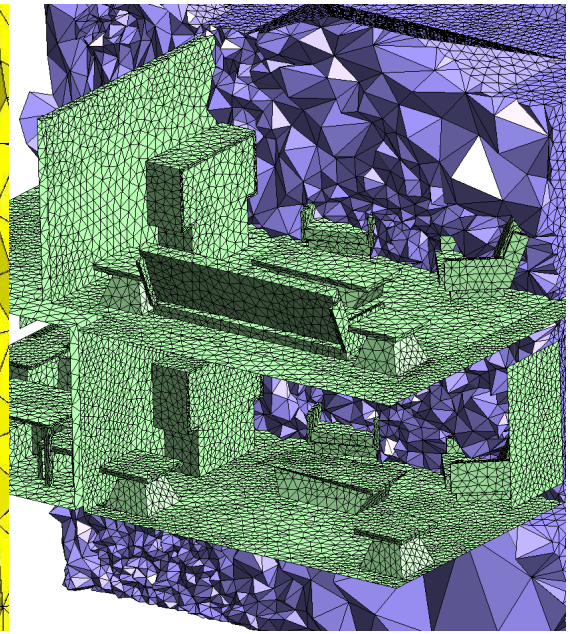
Local remeshing

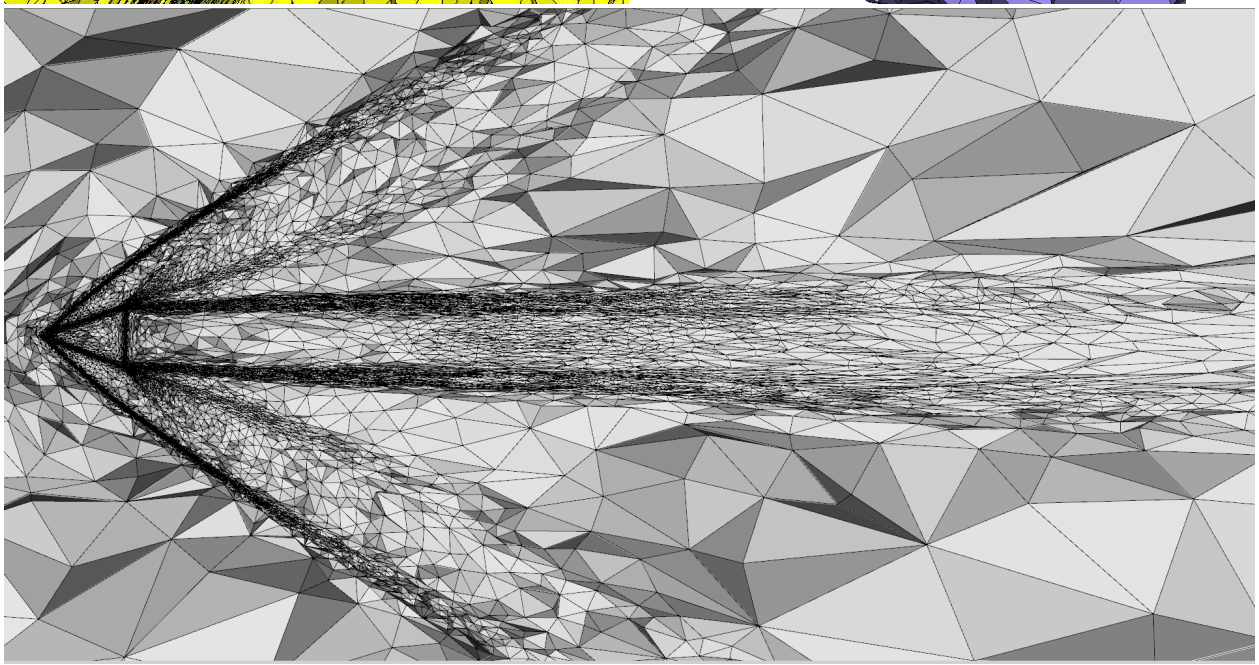
Generation of a computational mesh from an implicitly defined domain

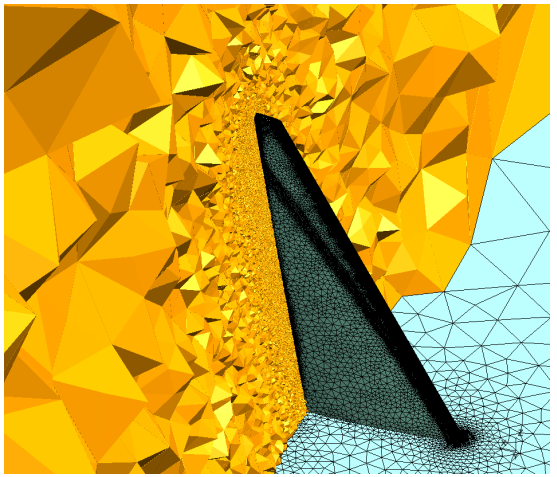
Anisotropic mesh adaptation for immersed boundary method

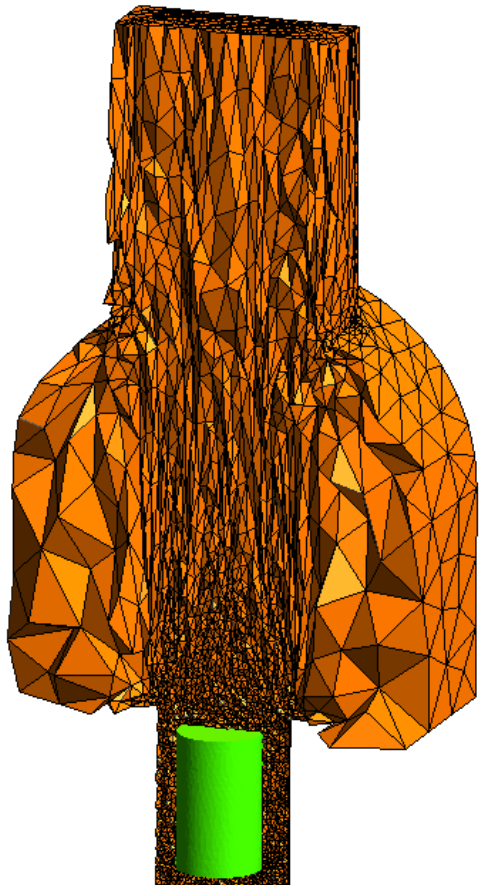
Mesh adaptation applications











MMG platform: overview

- **MMG2D** : 2d meshing/remeshing
- **MMGS** : surfacic remeshing
- **MMG3D** : 3d remeshing

Developped by

- C. Dobrzynski (Bordeaux), C. Dapopny (Grenoble), P. Frey (Paris), A. Froehly (Bordeaux)

- Distribution: https://github.com/MmgTools/mmg

Guideline

Introduction

Useful tools for mesh adaptation
- Drive the mesh process
- Evaluate a mesh

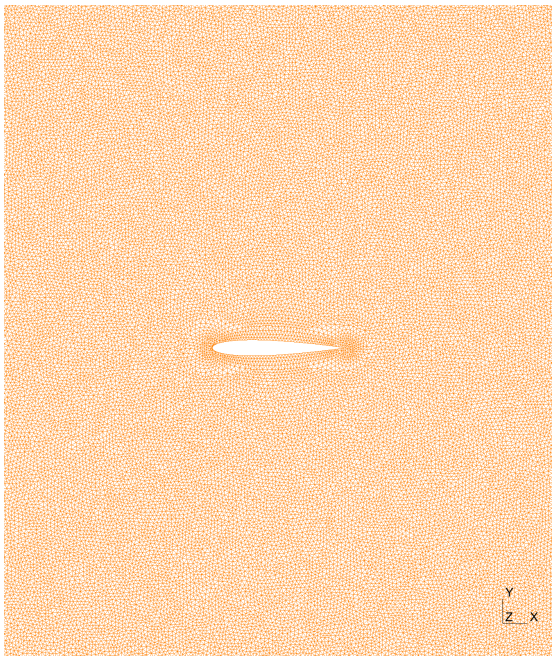
Local remeshing

Generation of a computational mesh from an implicitly defined domain

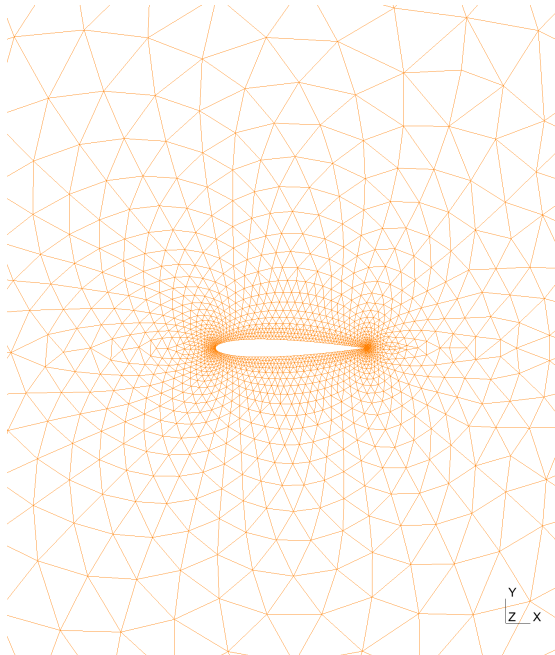
Anisotropic mesh adaptation for immersed boundary method

Drive the mesh process

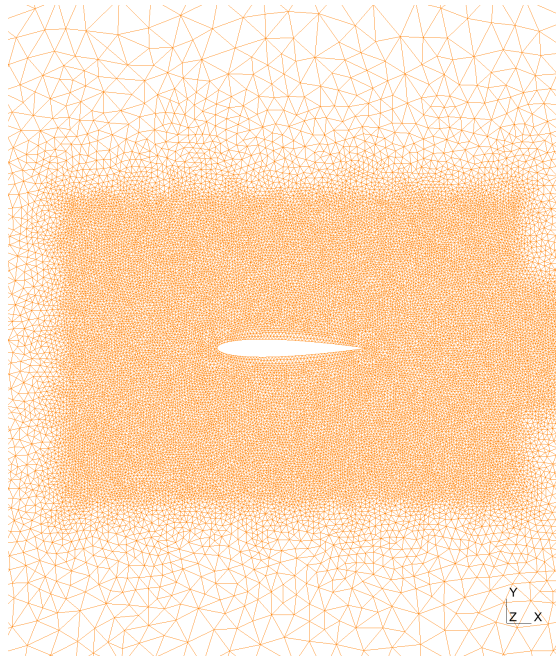
Generate a mesh suitable to numerical simulation



Uniform mesh



Non-uniform mesh

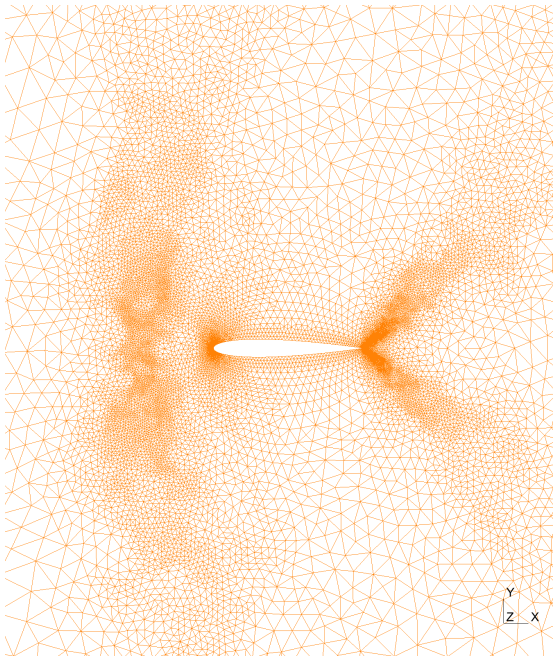


Non-uniform mesh

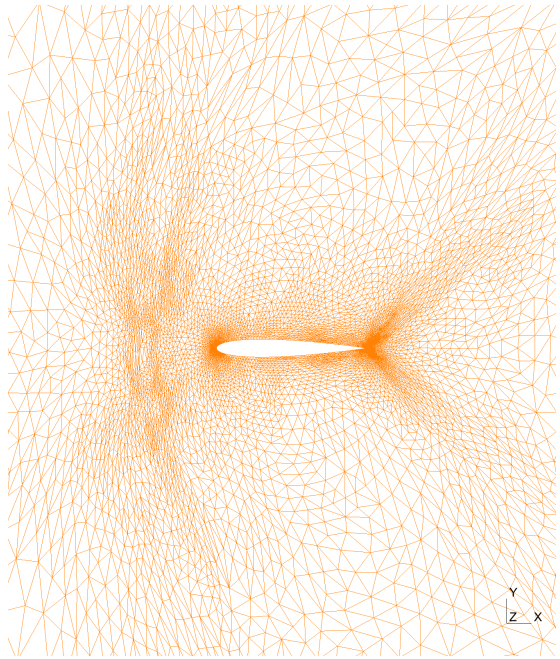
- Isotropic mesh : prescribed edge sizes
- At each point, give the prescribed sizes

Drive the mesh process

Adapt the mesh to the numerical solution



Isotropic mesh



Anisotropic mesh

- According to an error estimator, prescribe a size map
- Anisotropic case: sizes + directions

Mesh adaptation background: metric specification

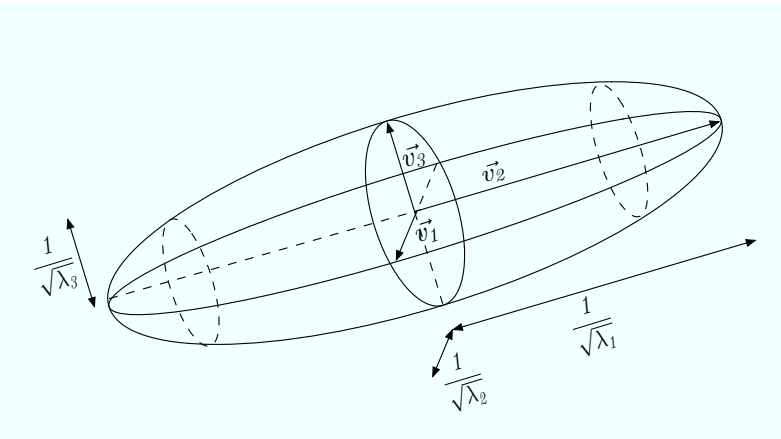
Metric definition

$d \times d$ symetric positive definite matrix:

$$M = \mathcal{R}\ \Lambda\ \mathcal{R}^{-1}$$

Edges orientation:
$\mathcal{R} = (\vec{v_1}\ \vec{v_2}\ \vec{v_3})$

Sizes prescription:
$\Lambda = diag(\lambda_1\ ,\ \lambda_2\ ,\ \lambda_3)$



Isotropic case

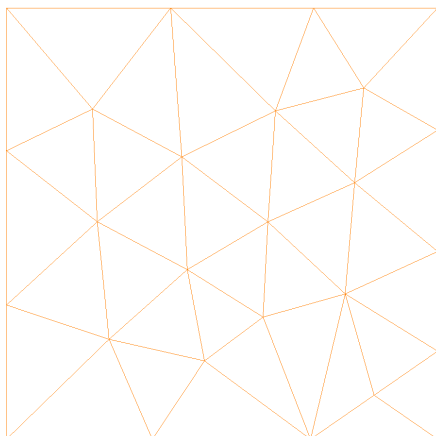
Ex: size prescription $= 0.1$

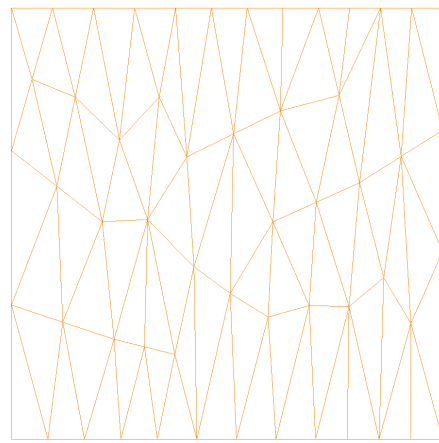
- $\lambda_1 = \lambda_2 = \lambda_3$
- ellipsoide $\Rightarrow$ sphere

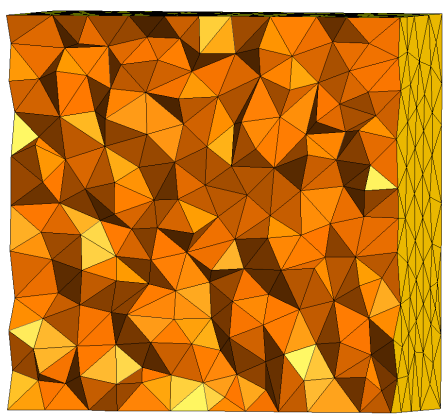
$$M = \begin{pmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{pmatrix}$$

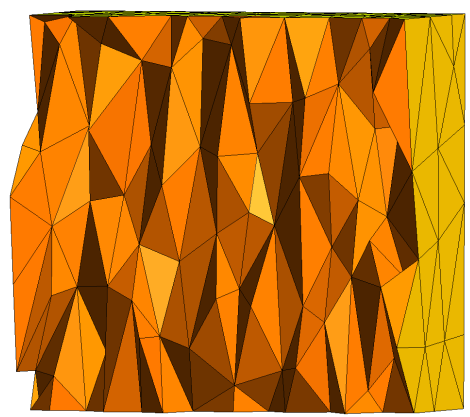
Anisotropic metric example

$$M = \begin{pmatrix} 100 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{pmatrix}$$





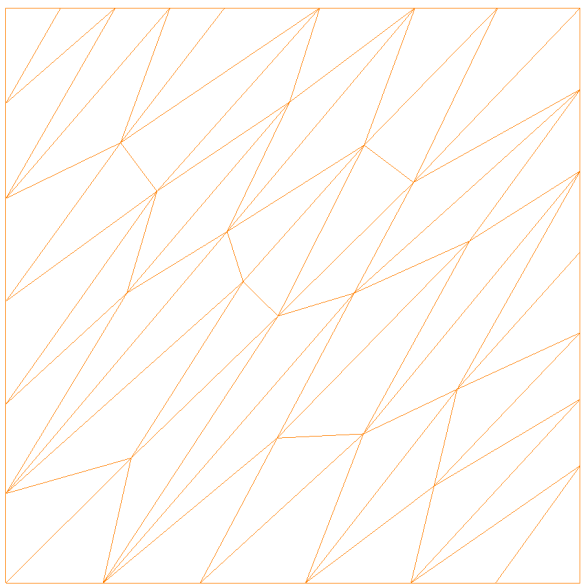


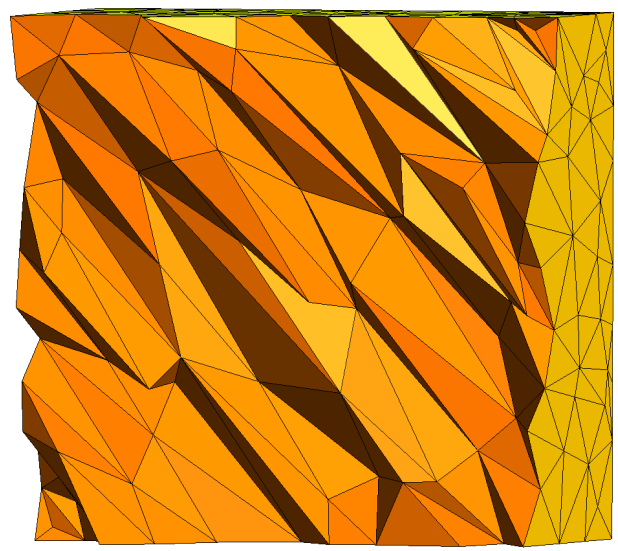


Anisotropic metric example

Rotation of $\frac{\pi}{4}$:

$$M = \begin{pmatrix} 55 & 0 & -45 \\ 0 & 10 & 0 \\ 0 & 0 & 45 \end{pmatrix}$$





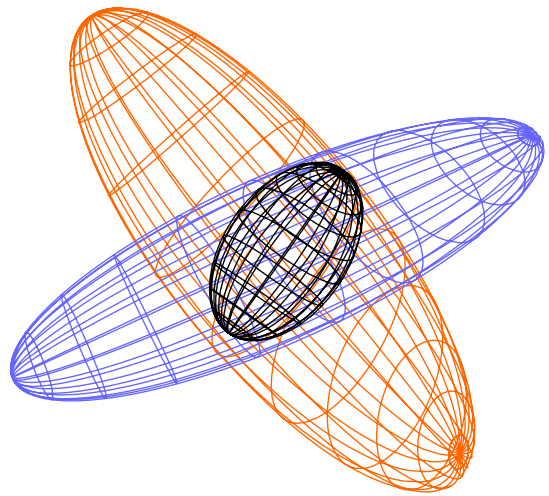
Mesh adaptation background

- Length definition for an edge e:

$$l_M(e) = \int_0^1 \sqrt{e^t M(t) e}\, dt$$

- Metric intersection:

$$\mathcal{M} = \mathcal{M}_1 \cap \mathcal{M}_2$$



- Metric interpolation:

$$\mathcal{M}(P) = \left(t\, \mathcal{M}(A)^{-\frac{1}{k}} + (1-t)\, \mathcal{M}(B)^{-\frac{1}{k}} \right)^{-k}$$

Error estimators examples

Isotropic

1. Have a sensor (based on gradient variations of a quantity, hessian variations...)
2. Impose cell size or divide cell size in 2 if needed

Anisotropic

Interpolation error majoration on an element K [1]:

$$\| u - \Pi_h u \|_{\infty,K} \leq \frac{9}{32} \max_{e \in E_K} \langle \vec{e}, \mathcal{M}(K)\, \vec{e} \rangle ,$$

where $\mathcal{M}(K)$ computed with the Hessian of u
and $\vec{e}$ an edge.

[1] P. Frey and al., Comput. Methods Appl. Mech. Engrg., Vol. 194, Issues 48-49, 2005

Guideline

Introduction

Useful tools for mesh adaptation

Local remeshing
- Geometric mesh
- Volumic remeshing
- Coupling surface/volume mesh generation

Generation of a computational mesh from an implicitly defined domain

Anisotropic mesh adaptation for immersed boundary method

Local remeshing

Principles

- input : a mesh + (not mandatory) a size map
- modify iteratively the mesh

2 main requirements

- well describe the underlying geometry
- be in agreement with the user-defined size map

Advantages

- only one mesh in memory,
- mesh always valid,
- in mesh adaptation process, very quick process.

Control the underlying geometry

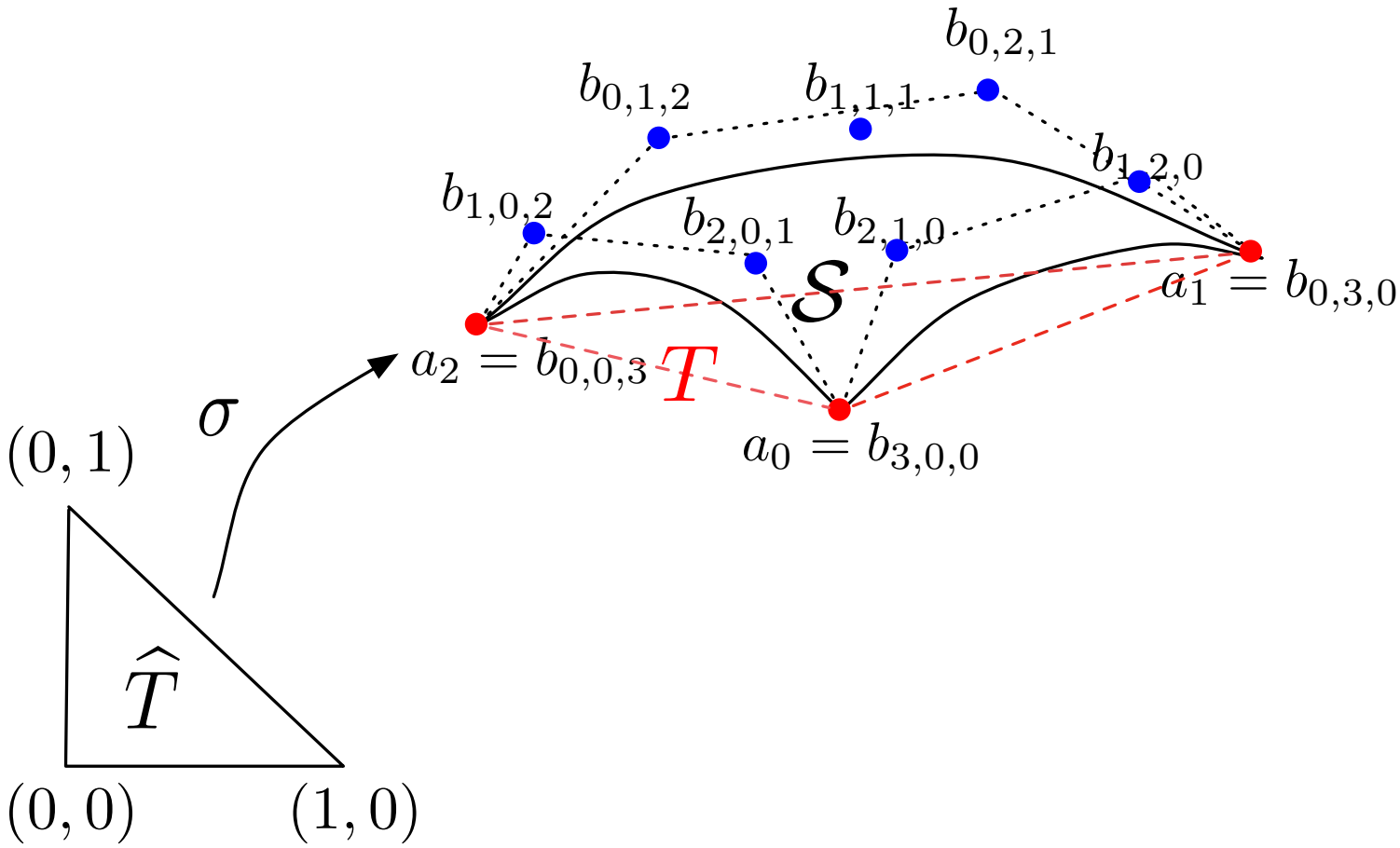
Principles

- input : a triangulated surface (no CAD)
- approximate the underlying geometry
- modify the mesh to control the geometric approximation

Tools

- surface models based on Bézier surface
- Hausdorff distance evaluation

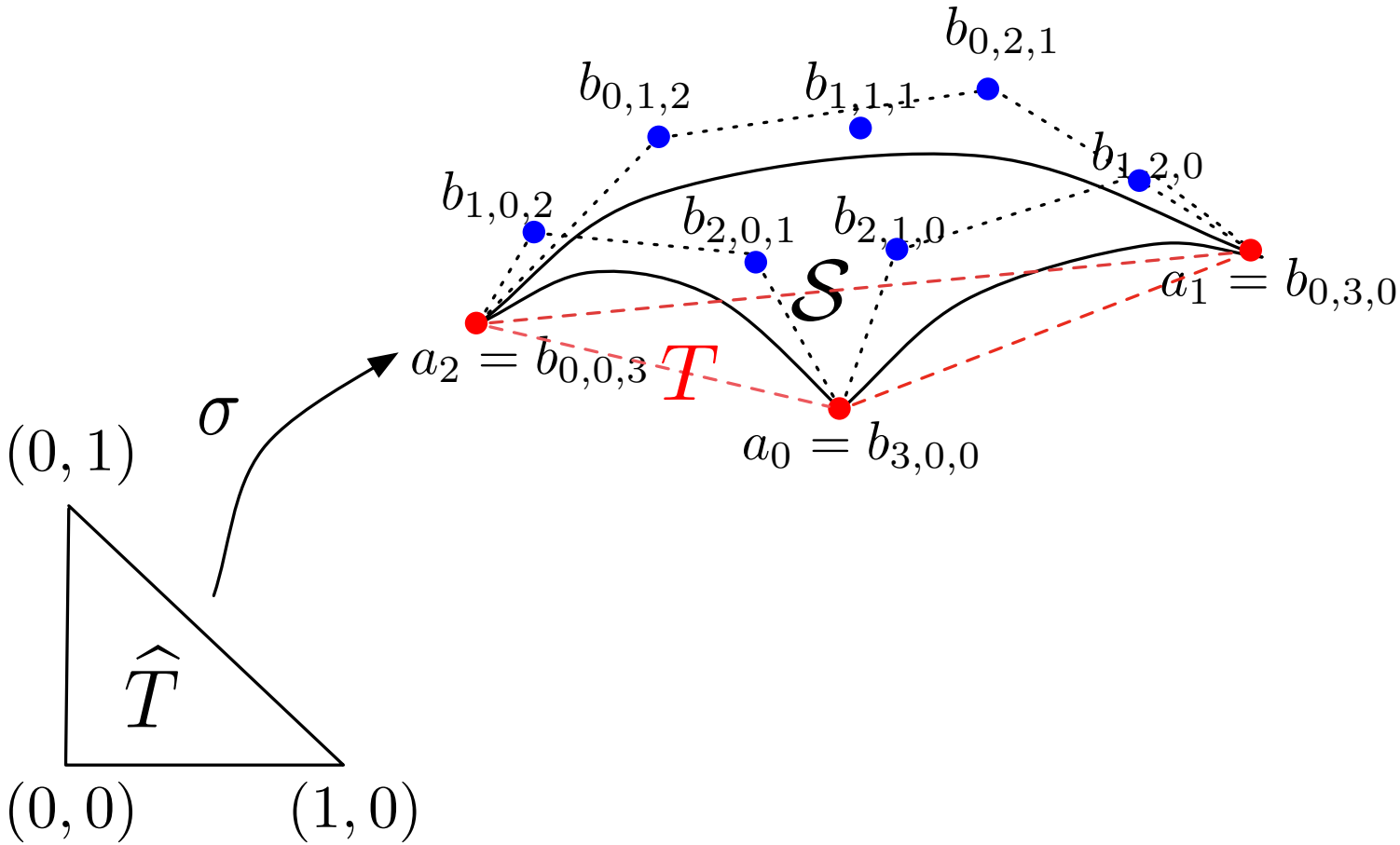
Surface model: cubic surface



Parametric Bézier cubic surface

$b_{i,j,k}$ control points

Surface model: cubic surface



- 3-order Bézier surface
- $\forall (u,v) \in \hat{T}$,

$$\sigma(u,v) = \sum_{i,j \in 0..3} \frac{3!}{i!j!k!} (1-u-v)^i u^j v^{1-i-j} b_{i,j,k}$$

Surface model: control points

Triangle vertex (interpolate by the Bézier surface)

$a0 = b_{3,0,0} \quad a1 = b_{0,3,0} \quad a2 = b_{0,0,3}$

Edge vertex

T_{a_i}: tangent plane at vertex a_i defined by $\vec{n}_i$ and a_i

- Hypothesis
  1. Tangents at a_i are onto T_{a_i}
  2. Edges are curves with constant speed
- Example for $b_{2,1,0}$
  1. projection of a_1 onto T_{a_0}
  2. $\vec{a_0 b_{2,1,0}} = \frac{1}{3}\vec{a_0 a_1}$

Last control point

- Such as if a quadratic surface exists, it coincides

Surface model

1. Geometric elements identification (corners, edges...)
2. Normal computation on each vertex P of the discrete surface

$$n(P) = \frac{\sum_{T \supset P} \alpha_T \times n_T}{||\sum_{T \supset P} \alpha_T \times n_T||} \text{ avec } \sum_{T \supset P} \alpha_T = 1$$

3. Construction of a local geometry :
   - 3-order Bézier surface
   - $\forall (u, v) \in \hat{T}$,

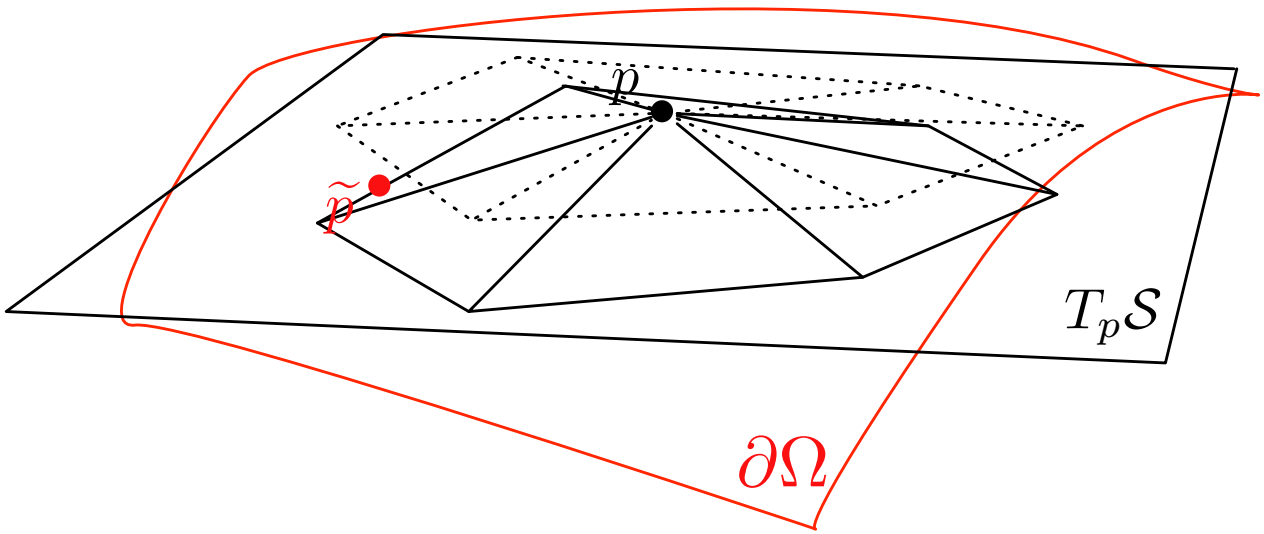
$$\sigma(u, v) = \sum_{i,j \in 0..3} \frac{3!}{i!j!k!} (1 - u - v)^i u^j v^{1-i-j} b_{i,j,k}$$

Geometric mesh algorithm

Main steps

- Geometric elements identification (corners, edges...)
- Construction of a local geometry
- Evaluation of the Haussdorff distance
- Nodes insertions/deletions to control the geometry
- Nodes relocations/edge swaps to improve the triangle quality

Surfacic node relocation



Point relocation

Volumic remeshing

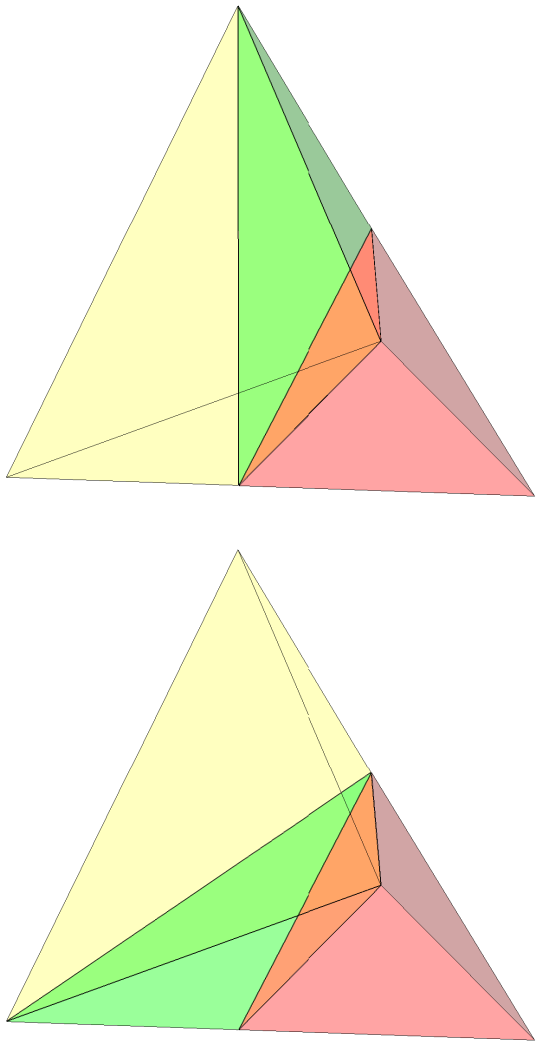
Principles

- drive by a user-defined size map
- no geometric constraints

Tools

- Nodes insertions
- Nodes deletions
- Edge swaps
- Nodes relocation

Node insertion by pattern



Point insertion by pattern.

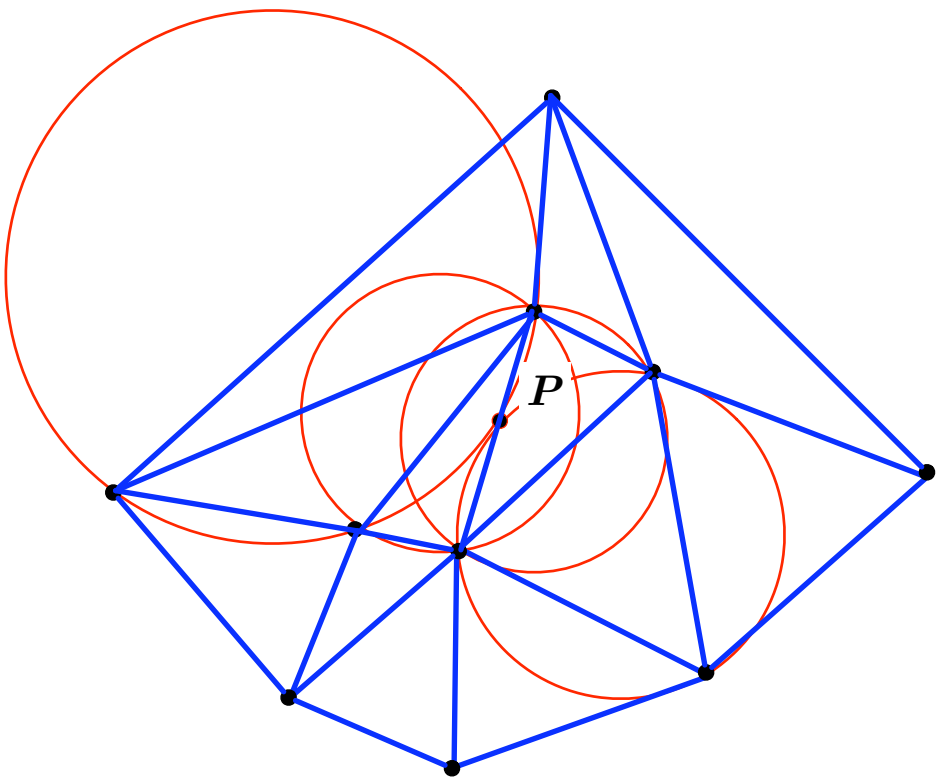
Delaunay triangulation:

- Delaunay measurement:

$$\alpha(K, P) = \frac{d(P, O_K)}{r_K}$$

- Cavity characterization:

$$K \in \mathcal{C}_P \text{ iff } \alpha(K, P) \leq 1.$$



Triangulation $\mathcal{T}_n$

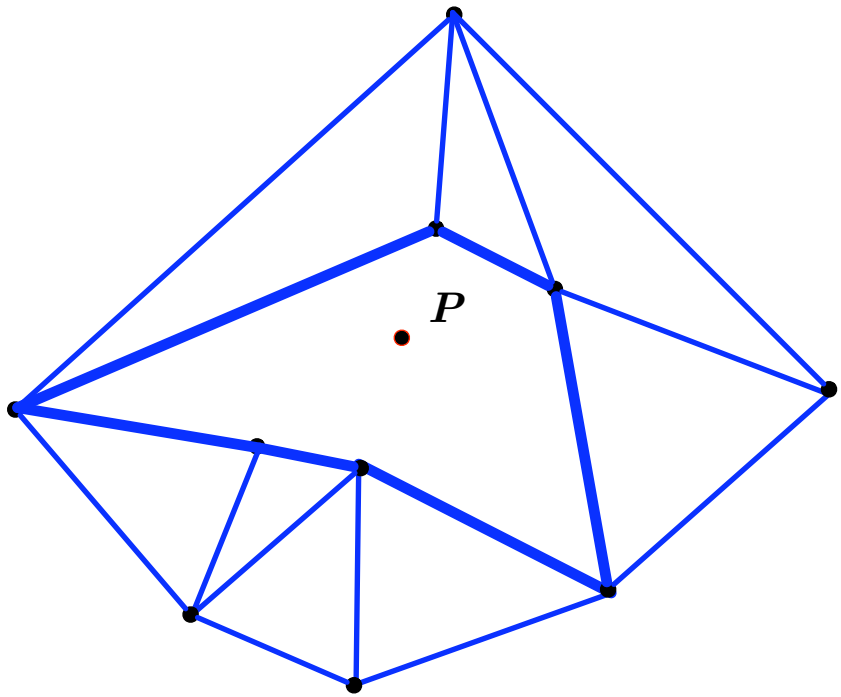
Delaunay triangulation:

- Delaunay measurement:

$$\alpha(K, P) = \frac{d(P, O_K)}{r_K}$$

- Cavity characterization:

$$K \in \mathcal{C}_P \text{ iff } \alpha(K, P) \leq 1.$$



$\mathcal{T}_n - \mathcal{C}_n$

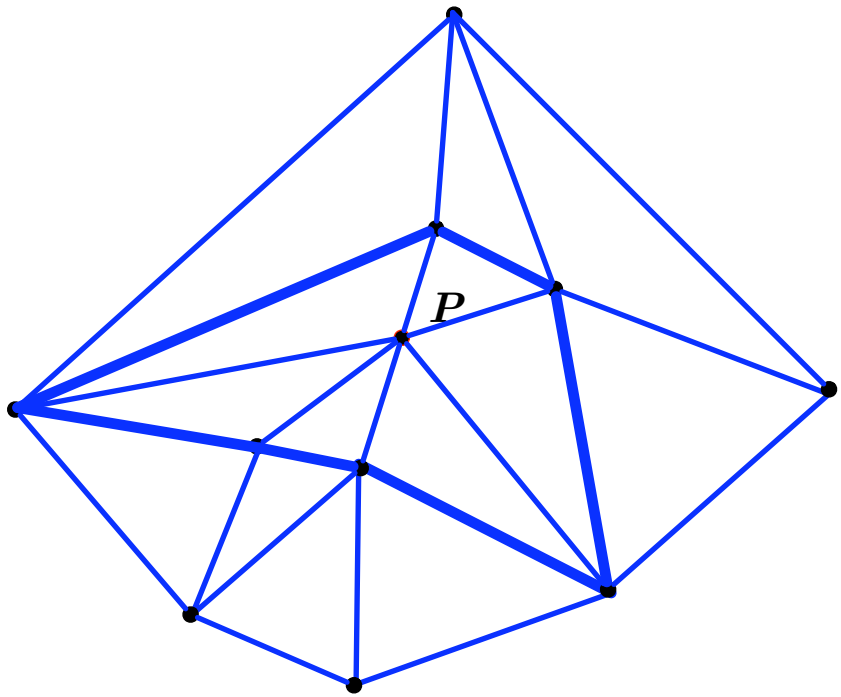
Delaunay triangulation:

- Delaunay measurement:

$$\alpha(K, P) = \frac{d(P, O_K)}{r_K}$$

- Cavity characterization:

$$K \in \mathcal{C}_P \text{ iff } \alpha(K, P) \leq 1.$$



$$\mathcal{T}_{n+1} = \mathcal{T}_n - \mathcal{C}_n \cup \mathcal{B}_{n+1}$$

Delaunay triangulation:

- Delaunay measurement:

$$\alpha(K,P) = \frac{d(P,O_K)}{r_K}$$

- Cavity characterization:

$$K \in \mathcal{C}_P \text{ iff } \alpha(K,P) \leq 1.$$

- Anisotropic extension:

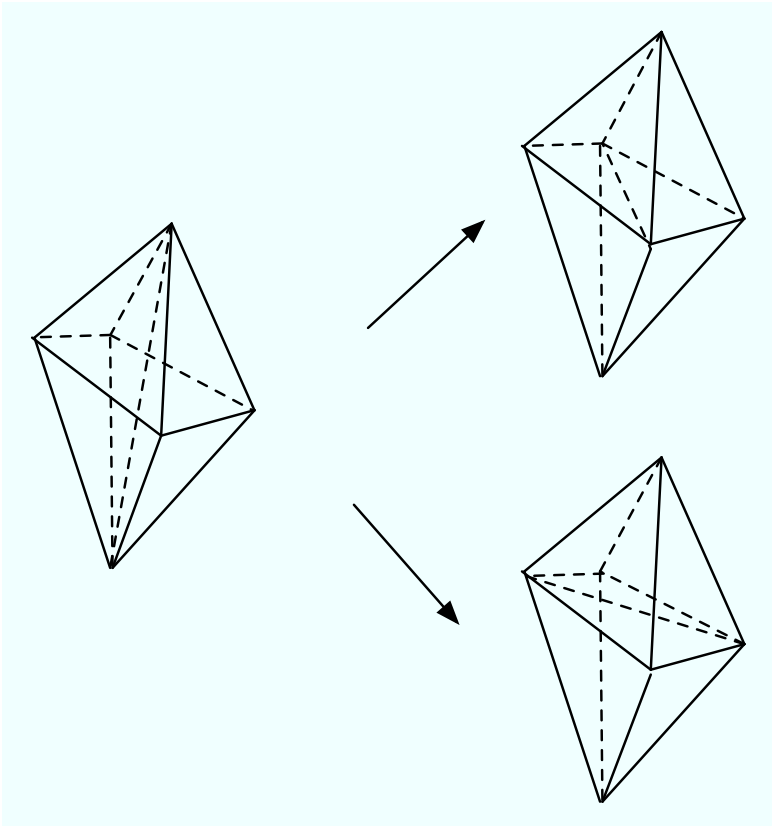
$$\alpha(K,P)_{\mathcal{M}} = \frac{\ell_{\mathcal{M}}(P,O_K)}{r_K}\,.$$

Node suppression

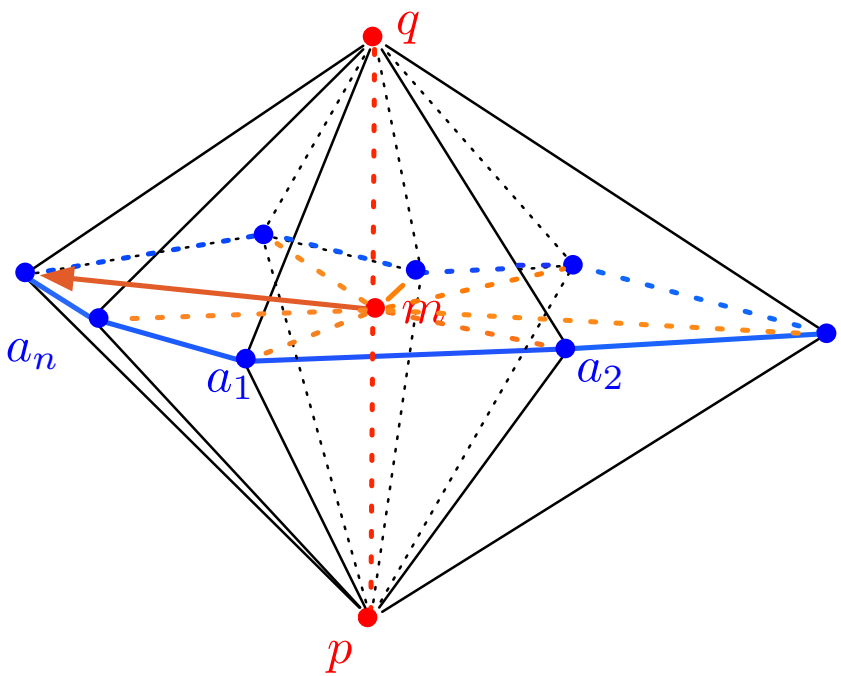
- Transform edge AB into vertex C.
  Three possibilities:
  1. Take $C = A$,
  2. Take $C = B$,
  3. Find a vertex C between A and B.

- Apply this operator if:
  1. all tetra containing C are valid (positive volume and admissible quality)
  2. new configuration has no big edges.

Edge swap



Edge swap



Node relocation

- Find a new position for P such as:
  - all tet containing P' have a better quality as the worst containing P
  - all edges from P' have an admissible lenght
- Optimal position:
  For all tet i in the ball of P, the optimal position P_i^{opt} is:

$$P_i^{opt} = \frac{1}{3}\sum_{j=1}^{3}(P + \frac{\overrightarrow{PP_j}}{l(PP_j)}) \tag{1}$$

  P' is found *via* a relaxation method as the barycenter of all the computed P_i^{opt}:

$$P' = (1-\omega)P + \omega(\frac{1}{n_b}\sum_{i=1,..,n_b} P_i^{opt}). \tag{2}$$

  where ω is the relaxation parameter (between 0 et 1).

General algorithm

1. mesh surface analysis
2. geometric remeshing (control of the Hausdorff distance)

3. Edge length analysis (both internal and surfacic)
4. Mesh optimisation (edge swap, node relocation)

At the end : adapted mesh to a prescribed size map

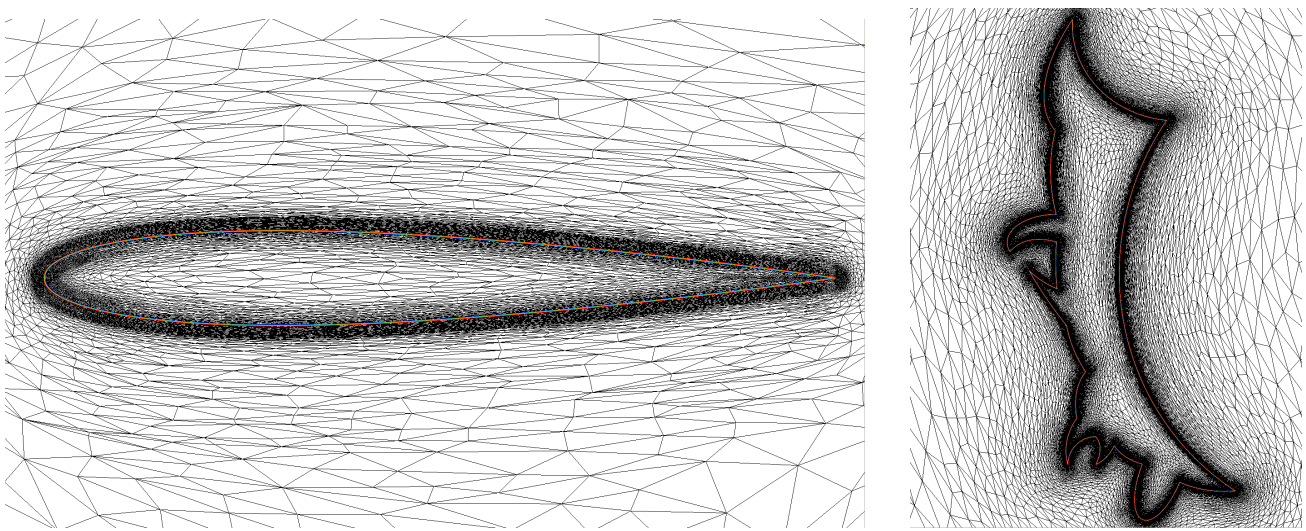
Motivation III: our choices

Combine:

- simplicity of embedded techniques
- strength of mesh adaptation

Tools:

- level-set description of solid bodies : Sign Distance Function (SDF)
- anisotropic mesh adaptation



Mesh for IBM. Right : Naca0012 airfoil - Left : 2D Complex Ice Shape

Outline

- Introduction
- Useful tools for mesh adaptation
  - Drive the mesh process
  - Evaluate a mesh
- Local remeshing
  - Geometric mesh
  - Volumic remeshing
  - Coupling surface/volume mesh generation
- Generation of a computational mesh from an implicitly defined domain
- Anisotropic mesh adaptation for immersed boundary method
  - General Ideas
  - Numerical methods
  - Accuracy and mesh adaptation
  - Numerical results
  - Moving bodies (ongoing work)

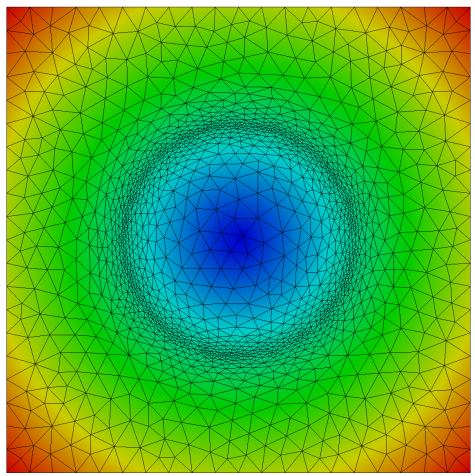
How to locate the objects ?

- Domain does not fit the obstacles
  $\Rightarrow$ need to know where is the inside and the outside

Definition: signed distance fonction

Considering a domain $\Omega_2 \subset \Omega_1$, delimited by a surface Γ :

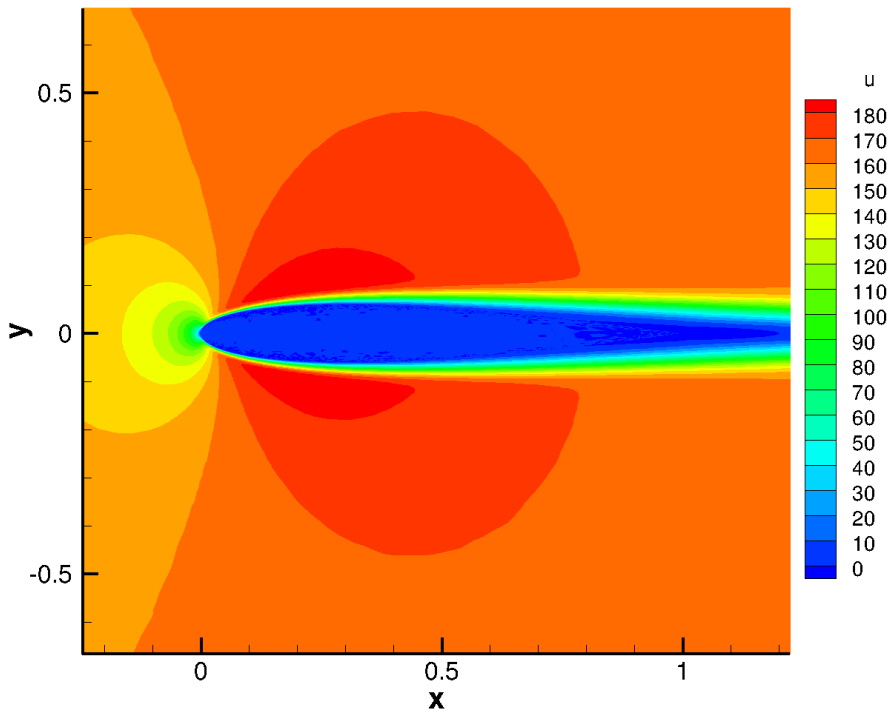
$$\phi(x,t) = \begin{cases} d(x,\Gamma) \text{ if } x \in \Omega_1 \setminus \Omega_2 \\ 0 \text{ if } x \in \Gamma \\ -d(x,\Gamma) \text{ if } x \in \Omega_2 \end{cases} \tag{3}$$



SDF on a mesh for a circle

How to impose boundary conditions ?

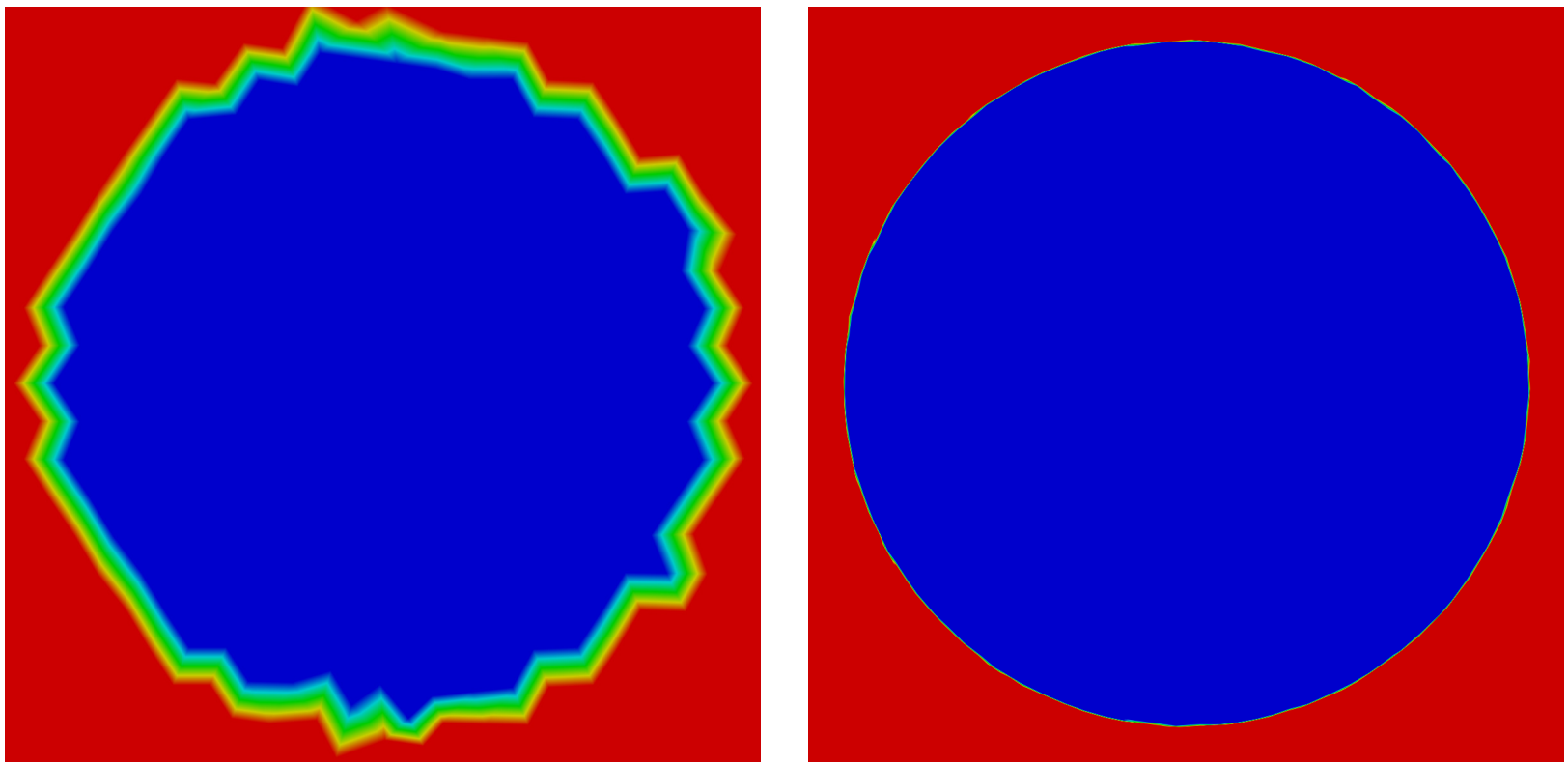
- With IBM solid wall BCs are taken into account differently
- Penalization : account for the rigid solid (through the governing equations) using a penalty term
  - Idea: extend the velocity field inside the solid



u-velocity for NACA0012

About accuracy

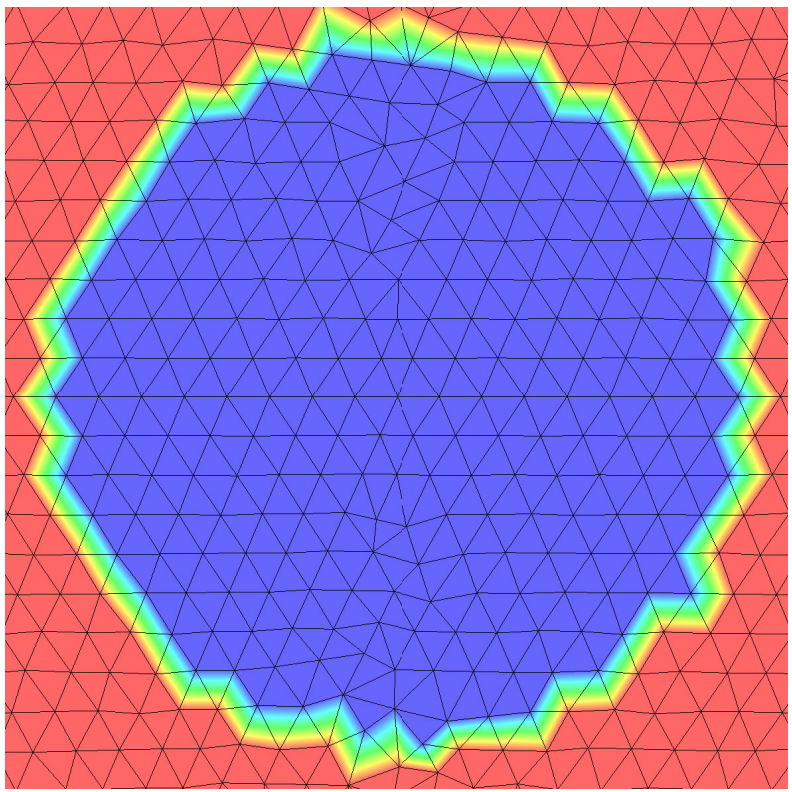
- Penalty term active inside the solid only
  $\Rightarrow$ accuracy depends on the capture of the interface
- Our proposition : mesh adaptation to improve accuracy of the SDF

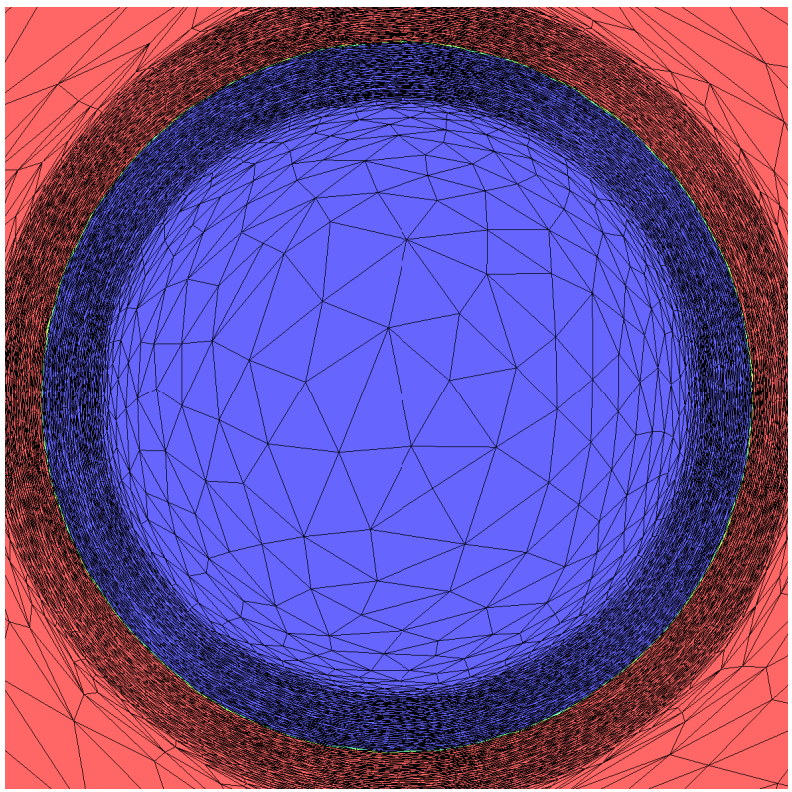


Characteristic function for a circle

About accuracy

- Penalty term active inside the solid only
  $\Rightarrow$ accuracy depends on the capture of the interface
- Our proposition : mesh adaptation to improve accuracy of the SDF





Characteristic function for a circle

Physical problem

Full compressible Navier-Stokes equations:

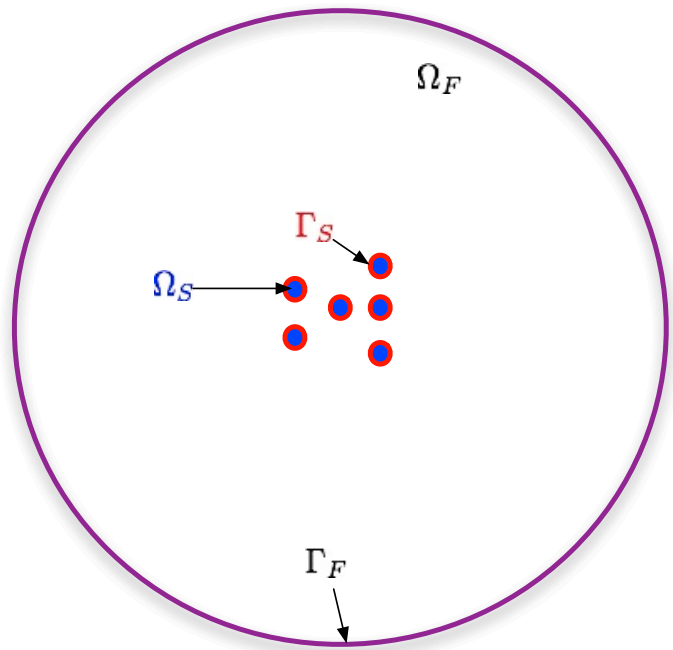
$$\left\{ \partial_t \boldsymbol{U} + \partial_{\boldsymbol{x}} \cdot \underline{\boldsymbol{F}} = \partial_{\boldsymbol{x}} \cdot \underline{\boldsymbol{G}} \right. \tag{4}$$

$$\boldsymbol{U} = \begin{pmatrix} \rho \\ \rho \boldsymbol{u} \\ \rho e \end{pmatrix}, \quad \underline{\boldsymbol{F}} = \begin{pmatrix} \rho \boldsymbol{u} \\ \rho \boldsymbol{u} \otimes \boldsymbol{u} + p \underline{\mathbf{Id}} \\ (\rho e + p) \boldsymbol{u} \end{pmatrix} \quad \text{and} \quad \underline{\boldsymbol{G}} = \begin{pmatrix} 0 \\ \underline{\boldsymbol{\pi}} \\ \underline{\boldsymbol{\pi}} \boldsymbol{u} + \boldsymbol{q} \end{pmatrix}$$

with $\underline{\boldsymbol{\pi}} = \mu \left(\left[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}} \right] + \left[\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}} \right]^T - \frac{2}{3} \left[\frac{\partial}{\partial \boldsymbol{x}} \cdot \boldsymbol{u} \right] \underline{\mathbf{Id}} \right)$ the stress tensor.

Boundary conditions:

- inflow/outflow on the outer boundary
- no-slip boundary on the obstacles Γ_{S^i}:



$$\left\{ \begin{array}{c} u_{\Gamma_{S^i}} = v_{\Gamma_{S^i}} = w_{\Gamma_{S^i}} = 0 \\ T_{\Gamma_{S^i}} = cte \end{array} \right.$$

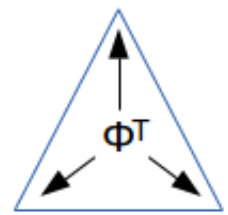
Residual distribution schemes

overview

- 1 - Compute Φ^T
- 2 - Distribute Φ^T to each DoF of the triangle

Nodal Residual

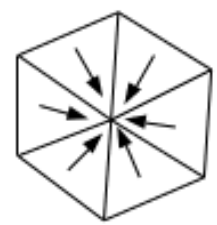
$$\Phi_i^T = \beta_i^T \Phi^T$$



- 3 - Gathering all the contributions of each triangle where i belongs

Residual Scheme

$$\sum_{T \ni i} \Phi_i^T(u_h) = 0$$



Resolution of the RDS scheme

- The obtained RDS scheme

$$\sum_{T \ni i} \Phi_i^T = 0$$

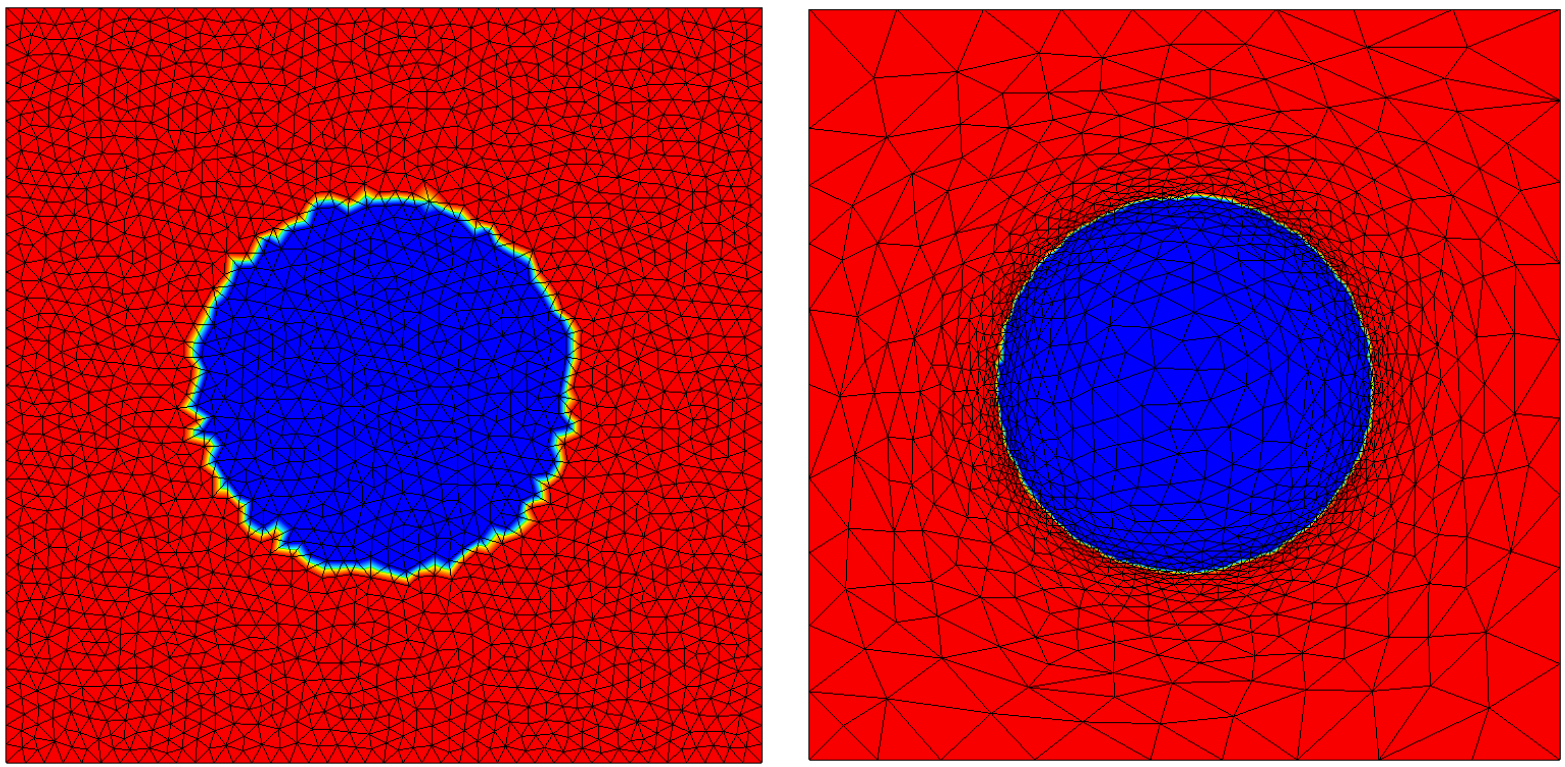
- is solved using a pseudo-iterative scheme

$$\begin{cases} \dfrac{u_i^{n+1} - u_i^n}{\Delta t} + \dfrac{1}{|C_i|} \displaystyle\sum_{T \ni i} \Phi_i^T = 0 \\ \\ u_i^0 \ \ given \end{cases}$$

- Implicit scheme is used such that $1/\eta >> 1$.
- $\eta = 10^{-12}$ in our steady simulations.

BCs inside penalty methods

- Penalty source term to impose BCs
- Accuracy depends on the capture of the interface
- Our proposition : use mesh adaptation to improve accuracy of the SDF



Characteristic function for a circle

Mesh adaptation background: metric specification

- $d \times d$ positive definite symmetric matrix:

$$M = \mathcal{R}\ \Lambda\ \mathcal{R}^{-1}$$

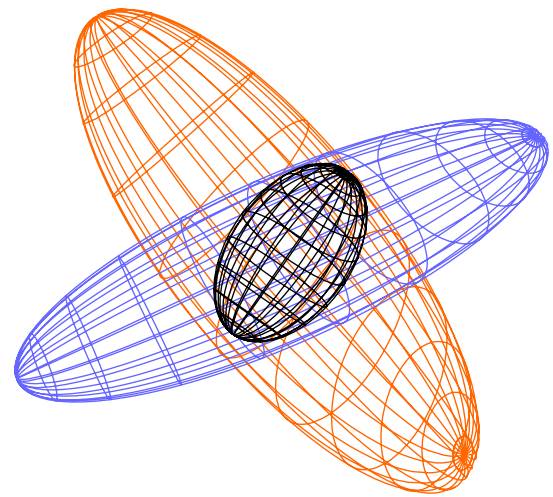
  $\mathcal{R}$ prescribes the orientation of the edges
  Λ prescribes the size

- Length definition for an edge e:

$$l_M(e) = \int_0^1 \sqrt{e^t M(t) e}\, dt$$

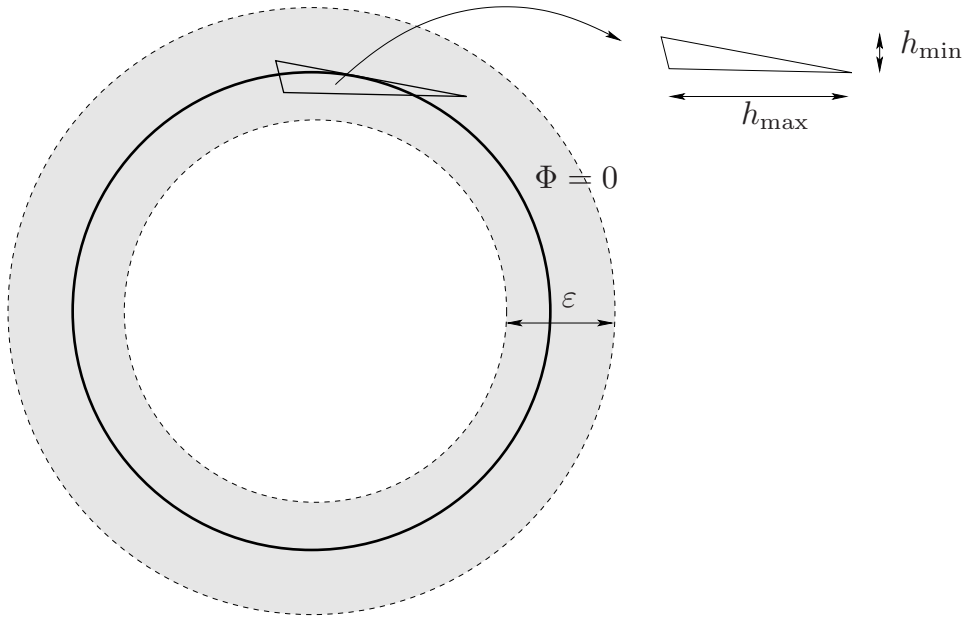
- Metric intersection:

$$M = M_1 \cap M_2$$



Mesh Adaptation : two criteria

1. Accurate representation of level-set function,



2. Accurate capture of flow features.

Metric definition for good accuracy of level-set [Frey and al.]

- Let's ε an error, h_{min} (resp. h_{max}) the minimal (resp. max.) length edge.

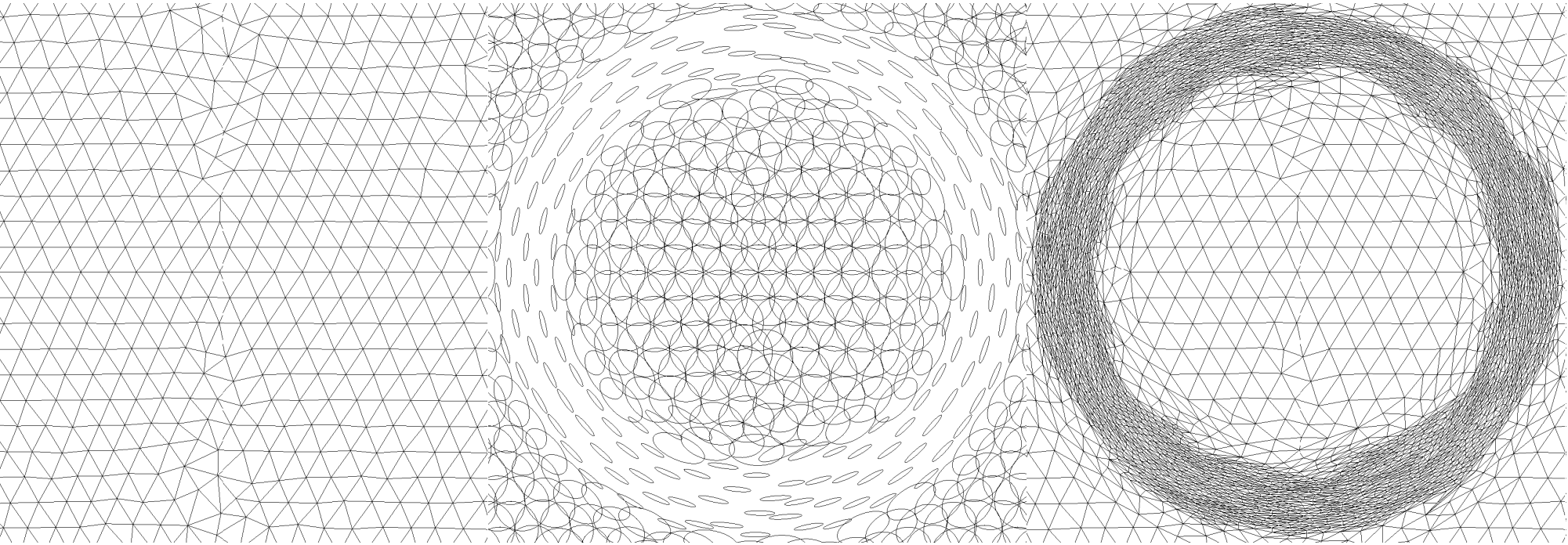
- The following metric allows to control the error of an isovalue:

$$M = R\ diag\left(\frac{1}{\varepsilon^2}, \frac{|\lambda_1|}{\varepsilon}, \frac{|\lambda_2|}{\varepsilon}\right) R^T \tag{8}$$

  with $R = (\nabla\Phi, v_1, v_2)$, (v_1, v_2) a basis of the tangent plane to the boundary and λ_i eigenvalues of the Hessian of Φ.

- In order to control the 0 isovalue,
  - $\forall$ nodes close to Γ_S, prescribe M
  - $\forall$ other nodes, increase linearly h_{min} and ε until h_{max}.

Example of level-set mesh adaptation

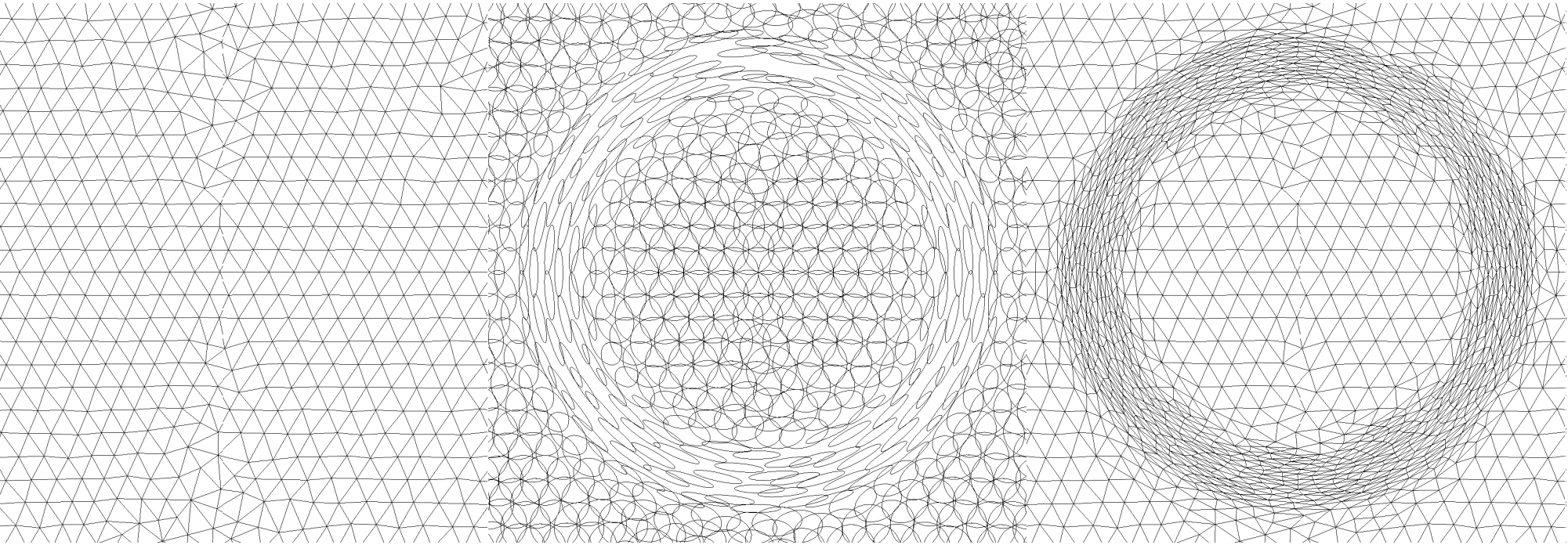


- near the 0-level-set :

  -hmax 0.06 -hmin 0.005 -eps 0.005

- elsewhere : isotropic mesh

Example of level-set mesh adaptation



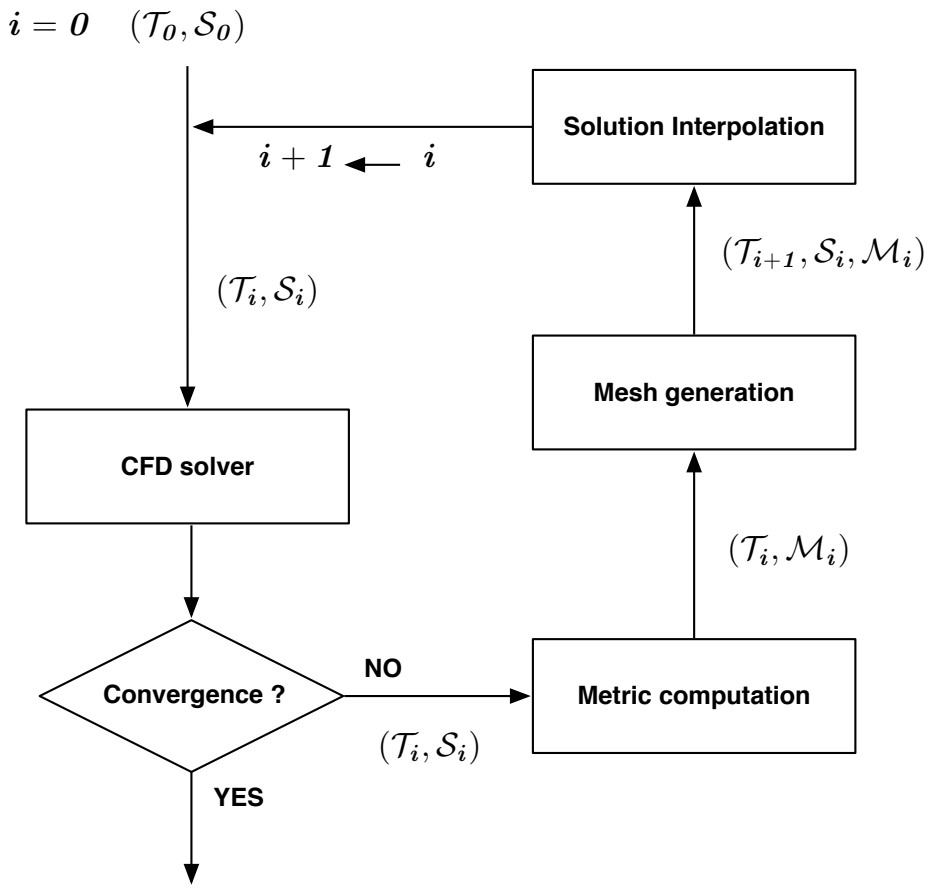
- near the 0-level-set :

  -hmax 0.06 -hmin 0.01 -eps 0.01

- elsewhere : isotropic mesh

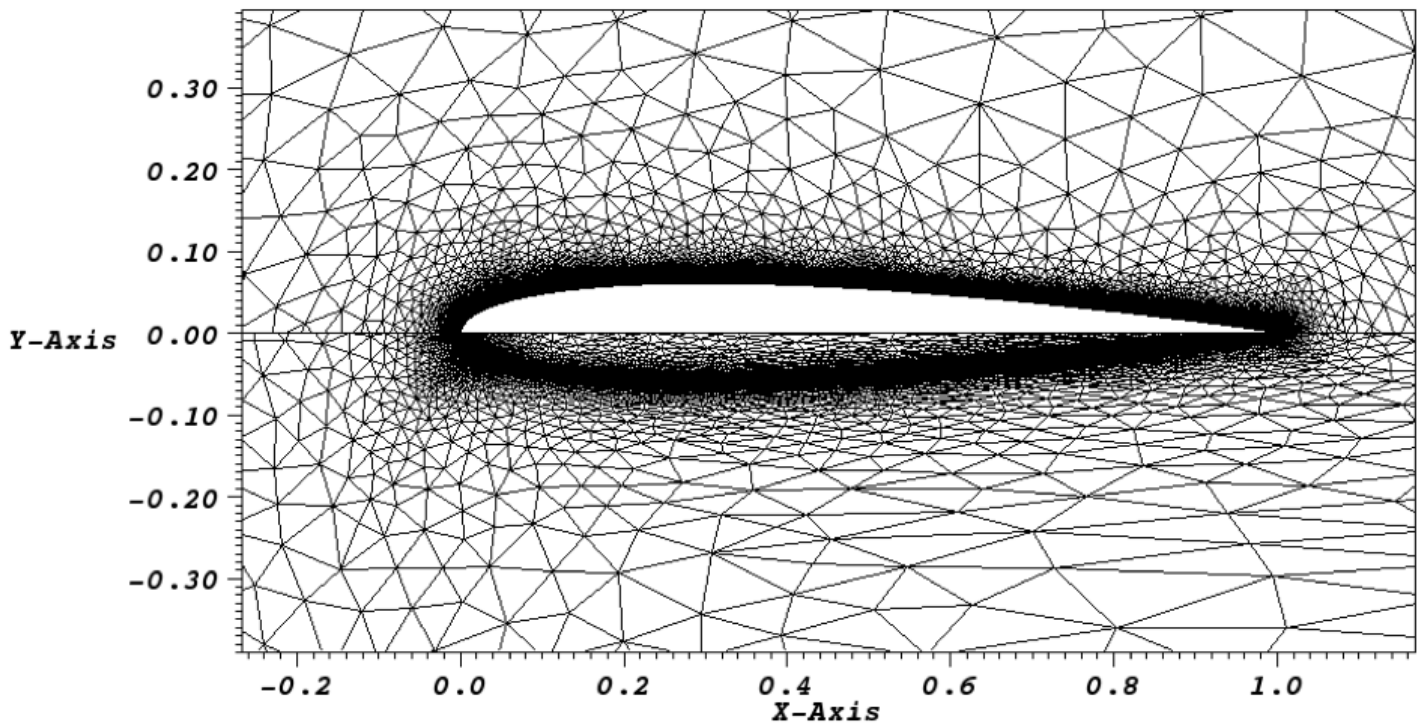
Mesh adaptation

Goal : accurate solution with minimum degrees of freedom ($\Rightarrow$ decreasing CPU time).



Mesh adaptation with penalization

- Laminar subsonic flow around Naca0012.
- Reynolds 5000 ; Mach 0.5 ; no angle of attack



Initial mesh : embedded (49 000 pts) and fitted (45 000 pts)

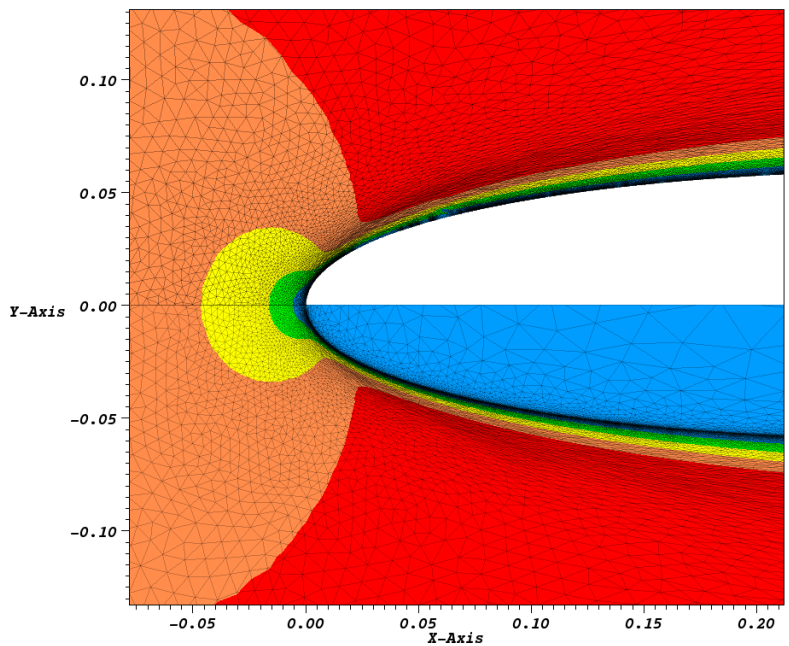
Mesh adaptation with penalization: leading and trailing edge

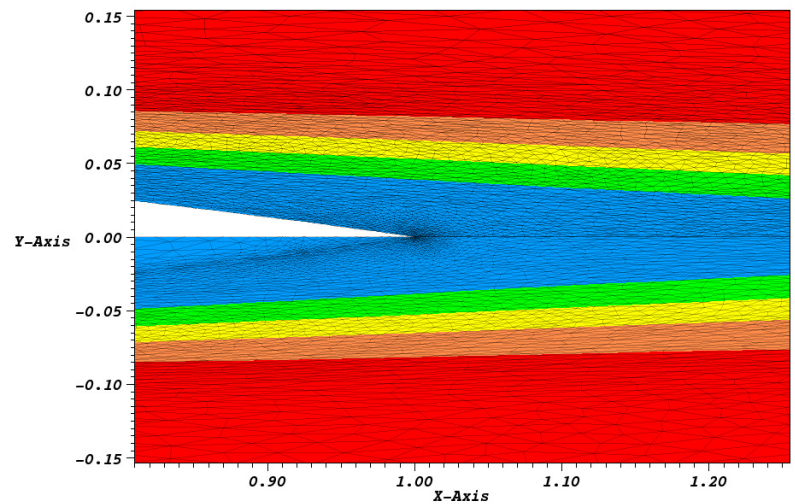
- physical adap parameter:

$$\varepsilon = 5e-4;\ h_{min} = .10^{-4};\ h_{max} = 2$$

- interface adap parameter:

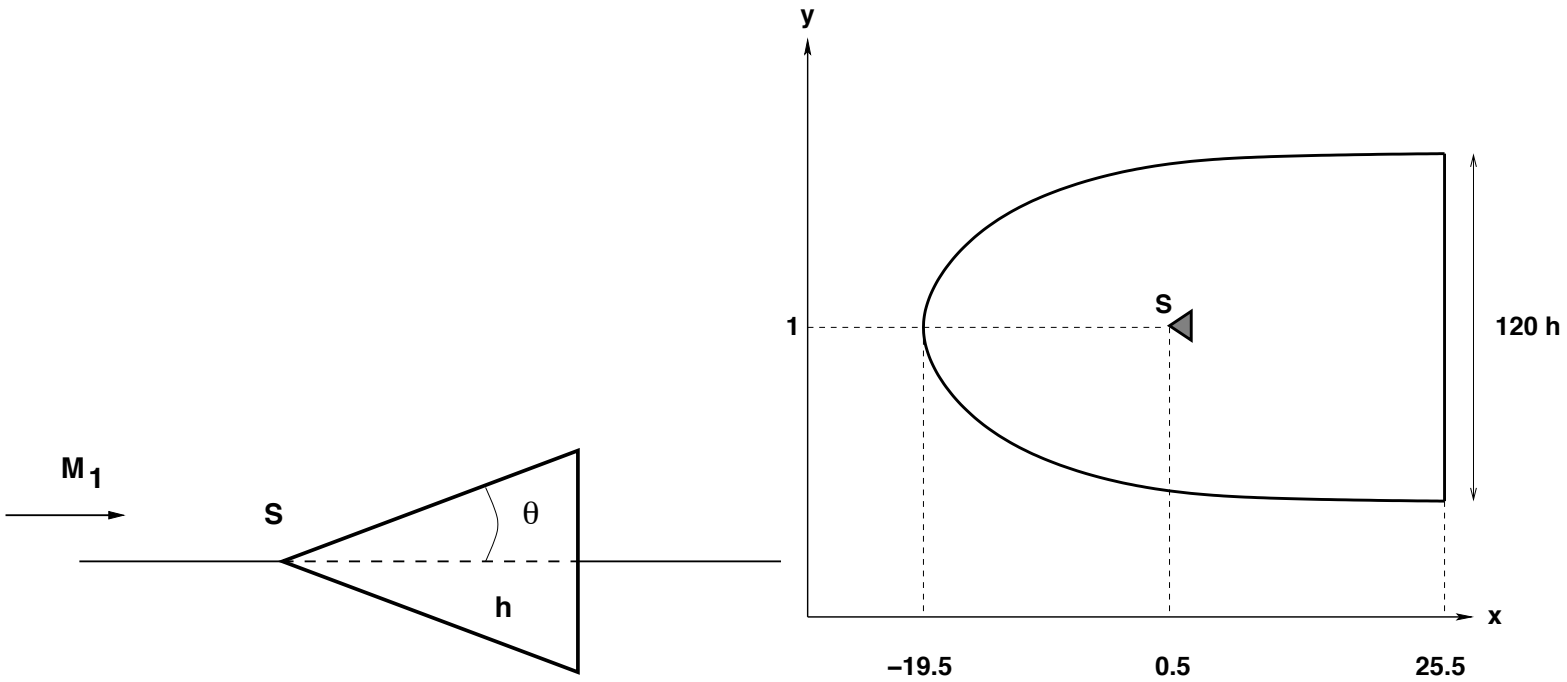
$$\varepsilon = h_{min} = 10^{-4};\ h_{max} = 2$$





Adapted mesh : embedded (101 000 pts) and fitted (85 000 pts)

Numerical results: Supersonic flow around a triangle [4]



$$h = 0.5, \quad \theta = 20 \text{ deg}, \quad S = (0.5\,,\, 1),$$

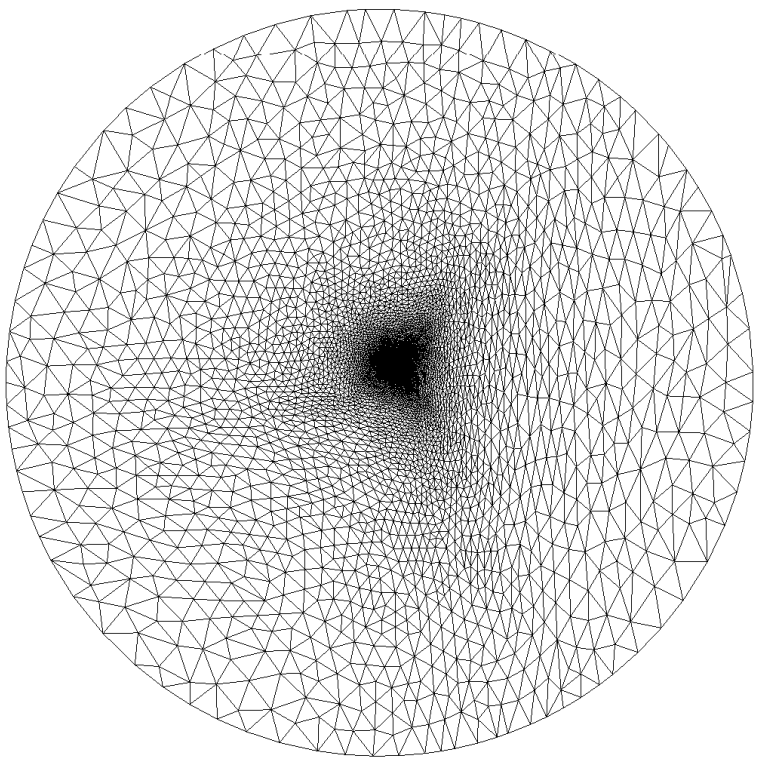
$$Re = 5 \times 10^4, \quad \text{Prandtl nb} = 0.72, \quad M_1 = 2, \quad T_s = 3.$$

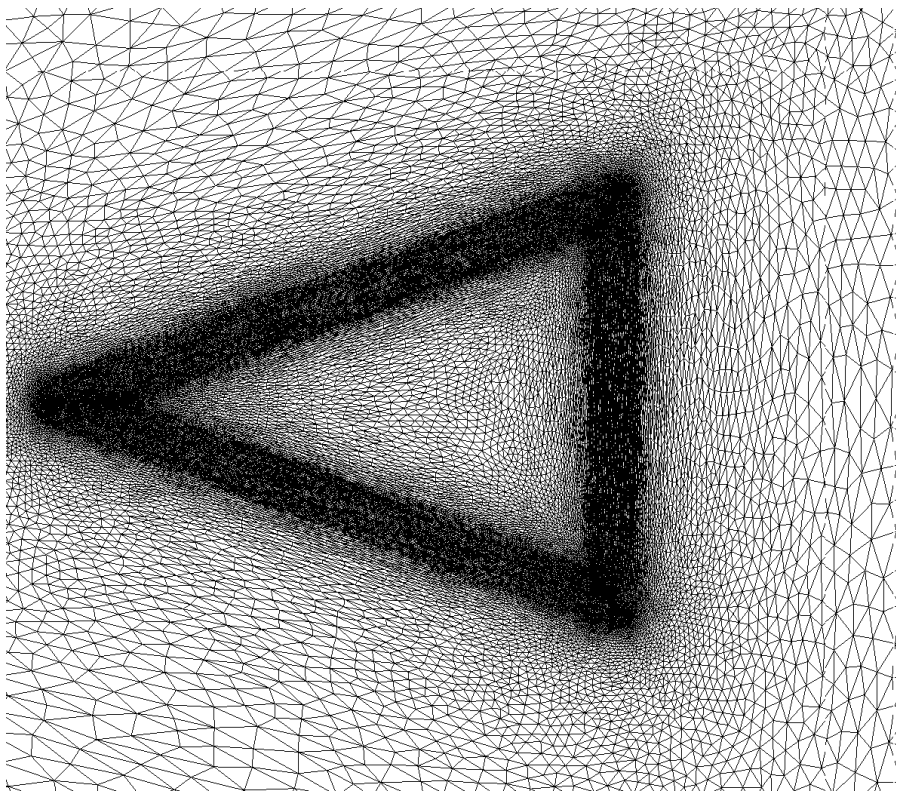
Penalized parameters inside the triangle $\boldsymbol{u} = 0$, $T = 3$

[4] O. Boiron, G. Chiavassa, and R. Donat. *Computers and Fluids*, 2009.

Numerical results: Supersonic flow around a triangle

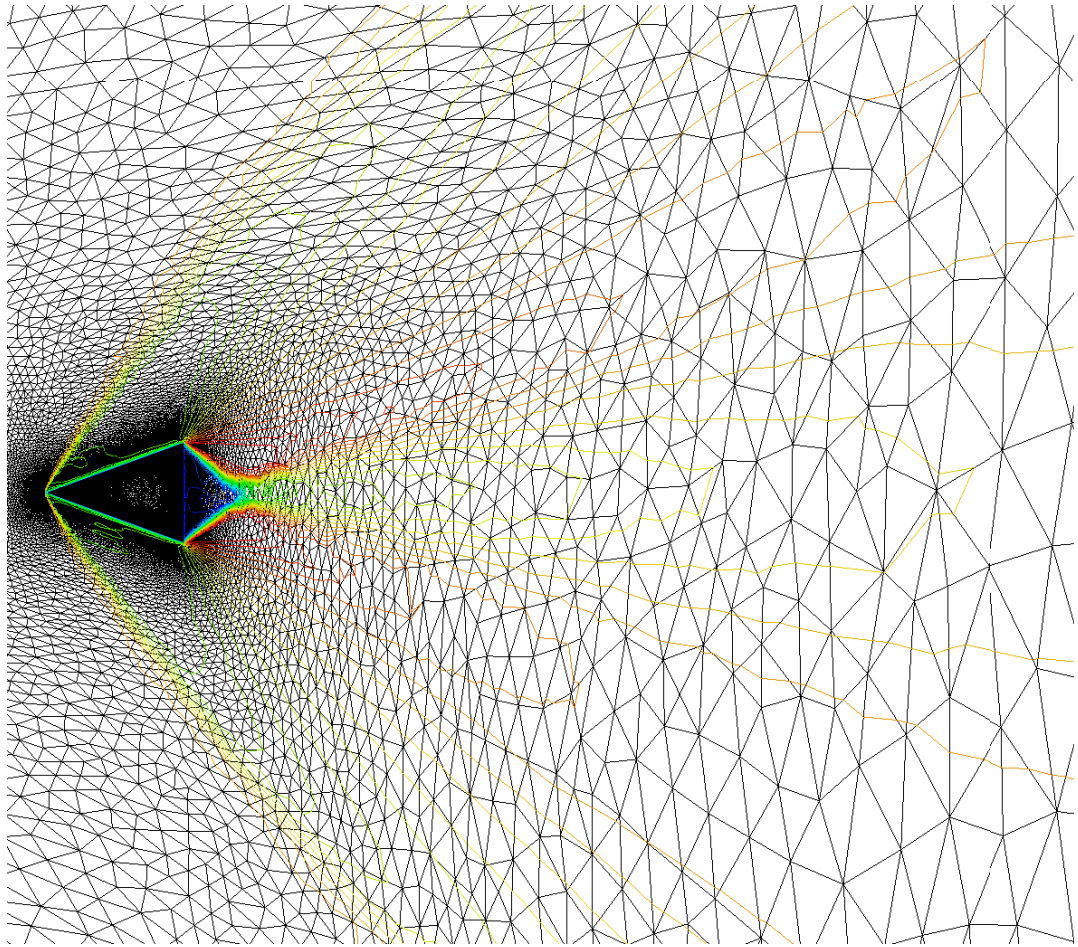
- Initial mesh : 30407 nodes and 60730 triangles
- Interface adaptation parameters :
  $\varepsilon = h_{min} = 1.10^{-4}$; $h_{max} = 2$





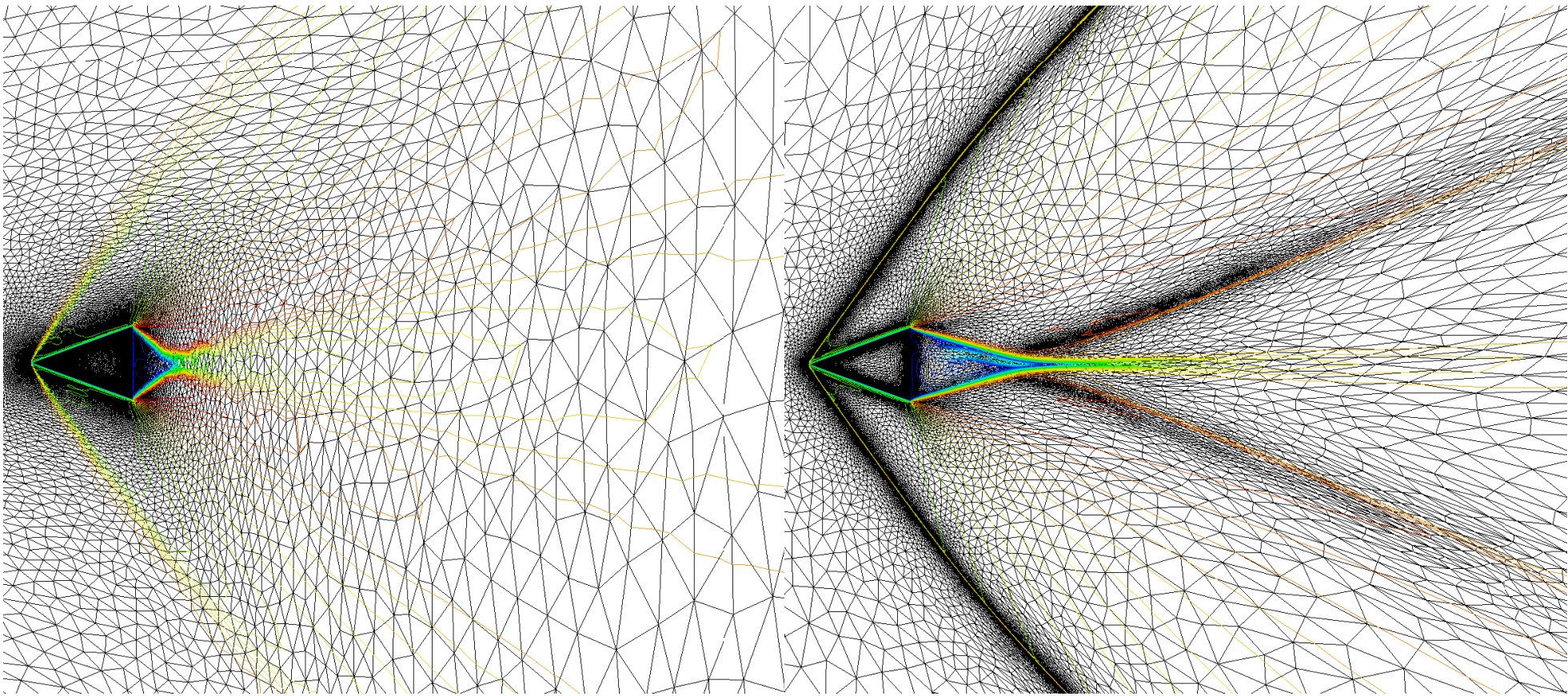
Numerical results: Supersonic flow around a triangle

- Initial mesh : 30407 nodes and 60730 triangles
- Adaptation parameters : $\varepsilon = h_{min} = 1.10^{-4}$; $h_{max} = 2$

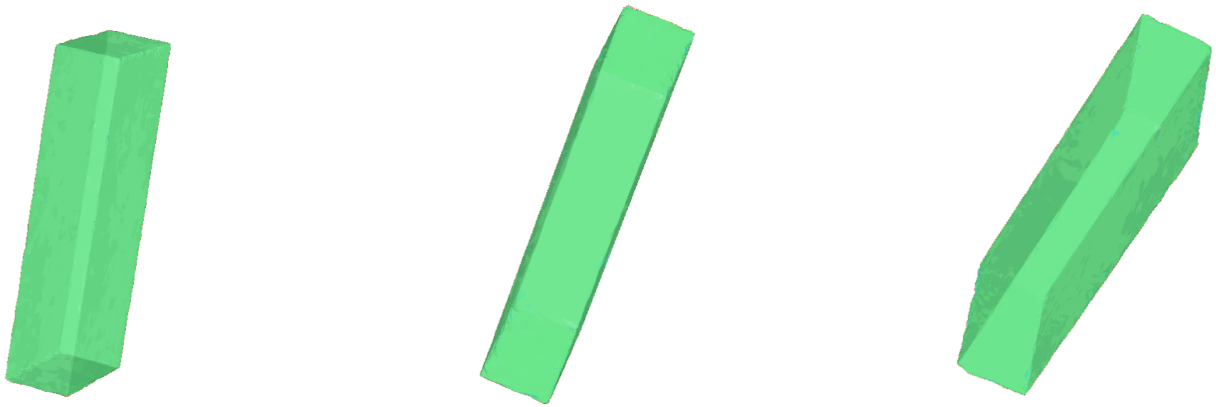


Numerical results: Supersonic flow around a triangle

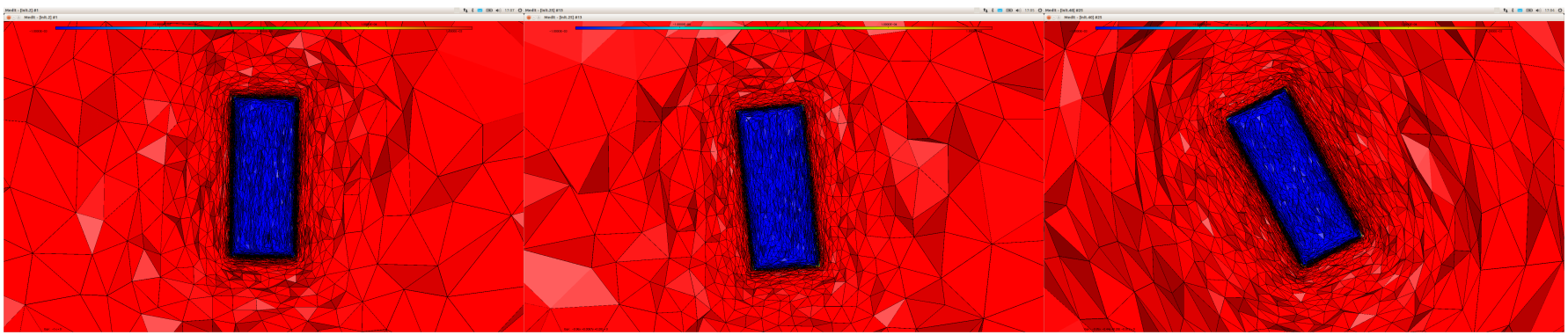
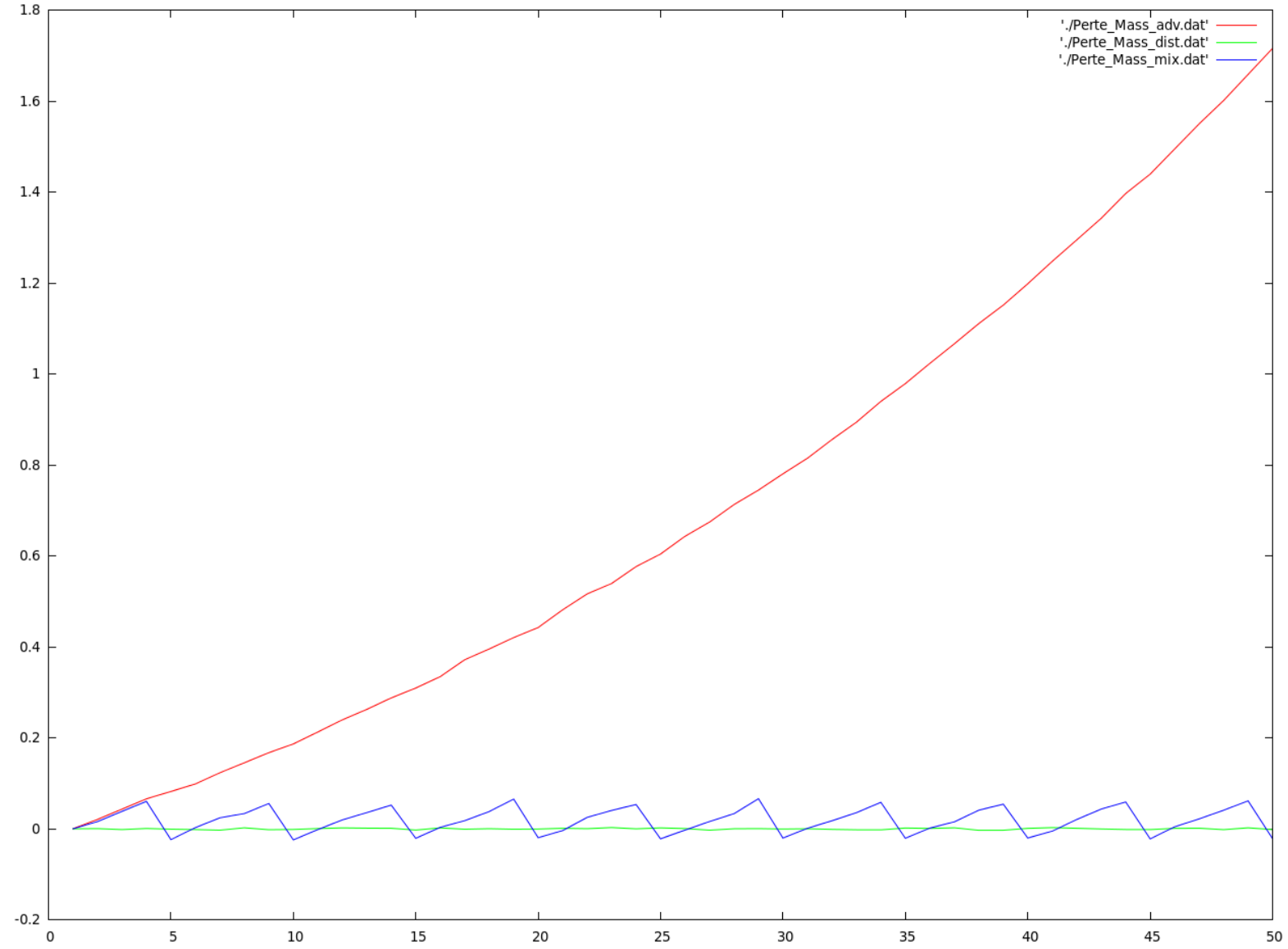
- Initial mesh : 30407 nodes and 60730 triangles
- Adaptation parameters : $\varepsilon = h_{min} = 1.10^{-4}$; $h_{max} = 2$
- after 3 cycles of adaptation : 49648 nodes and 99184 triangles



Rotation of a rectangular block



0 isoline, from left to right : $t = 0$, $t = 1.64$, $t = 2.71$



Rotation of a rectangular block: mass loss