Unsteady Mesh Adaptation

Nicolas Barral, Adrien Loseille and Frédéric Alauzet

INRIA - Gamma3 Project - Rocquencourt, France

Kick-off ANR MAIDESC

1. Unsteady solutions: metric-based anisotropic mesh adaptation for unsteady flows
2. Moving geometries: moving mesh algorithm
3. Perspectives

1. Unsteady Mesh Adaptation
2. Moving Mesh Algorithm
3. Perspectives

Motivations

Reminder: Metric-based Mesh Adaptation

To address these problems : $\mathbf{L}^p$ metric-based adaptation

- Error estimate $\longrightarrow$ metric field $\longrightarrow$ unit mesh
  [Vallet, 1992], [Casto-Diaz et Al., 1997], [Hecht et Mohammadi, 1997]
  [Peraire et Al., 1987] [Zienkiewicz et Wu, 1994]
  $\Longrightarrow$ based on continuous mesh model
- Unit mesh obtained with a local remesher
  $\Longrightarrow$ cavity-based primitive [Loseille, IMR, 2012]

Other mesh adaptation methods:
[Bottasso, Tango, Poly. Milano], [Coupez, Forge3D, CEMEF], [Dobrinsky et al., MMG3D, UMB], [George et al., INRIA] [Lipnikov et al. Los Alamos], [Loseille, Feflo.a, INRIA], [Park, Nasa Langley], [Piggot, MadLib, Imp. College] [Oubay et al., Swansea U.], [Remacle et al., Louvain U.], [Shepard et al., MeshAdapt, SCOREC]

Reminder: Continuous Mesh Model [Loseille and Alauzet, SIAM 2011]

Discrete	**Continuous**
Element K	Metric tensor $\mathcal{M}$
Volume $\lvert K \rvert$	Volume $\alpha\,(\det\mathcal{M})^{-\frac{1}{2}}$
Mesh $\mathcal{H}$ of Ω_h	Riemannian metric space $\mathbf{M} = (\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$
Number of vertices N_v	Complexity $\mathcal{C}(\mathbf{M}) = \int_\Omega \sqrt{\det(\mathcal{M}(\mathbf{x}))}\,d\mathbf{x}$
Linear interpolate $\Pi_h u$	Continuous linear interpolate $\pi_{\mathcal{M}} u$
Interpolation error $\lVert u - \Pi_h u \rVert$	$\mathrm{Trace}(\mathcal{M}^{-\frac{1}{2}}\lvert H_u\rvert\mathcal{M}^{-\frac{1}{2}})$

Problem

- **Discrete** space-time mesh adaptation :

Find $\mathcal{H}_{opt}$ having N_{st} space-time vertices such that

$$\mathcal{H}_{opt}(u) = \text{Arg}\min_{\mathcal{H}} \|u - \Pi_h u\|_{\mathcal{H},\mathbf{L}^p(\Omega\times[0,T])}$$

- **Continuous** space-time mesh adaptation:

Find $\mathbf{M}_{\mathbf{L}^p} = (\mathcal{M}_{\mathbf{L}^p}(\mathbf{x},t))_{(\mathbf{x},t)\in\mathcal{Q}}$ of space-time complexity N_{st} such that

$$\begin{aligned} E_{\mathbf{L}^p}(\mathbf{M}_{\mathbf{L}^p}) &= \min_{\mathbf{M}} \left(\int_0^T\int_\Omega |u(\mathbf{x},t) - \pi_{\mathcal{M}} u(\mathbf{x},t)|^p \,\mathrm{d}\mathbf{x}\mathrm{d}t\right)^{\frac{1}{p}} \\ &= \min_{\mathbf{M}} \left(\int_0^T\int_\Omega \mathrm{Trace}\left(\mathcal{M}(\mathbf{x},t)^{-\frac{1}{2}}|H_u(\mathbf{x},t)|\mathcal{M}(\mathbf{x},t)^{-\frac{1}{2}}\right)^p \mathrm{d}\mathbf{x}\mathrm{d}t\right)^{\frac{1}{p}} \end{aligned}$$

under constraint $\quad \mathcal{C}_{st}(\mathbf{M}) = N_{st} = \displaystyle\int_0^T \tau(t)^{-1}\left(\int_\Omega d_{\mathcal{M}}(\mathbf{x},t)\mathrm{d}\mathbf{x}\right)\mathrm{d}t$

Fixed-Point Mesh Adaptation Algorithm [Alauzet et al., JCP 2007]

- A global fixed-point algorithm

  $\Longrightarrow$ to compute the space-time metric complexity

  $\Longrightarrow$ to converge the non-linear mesh adaptation problem

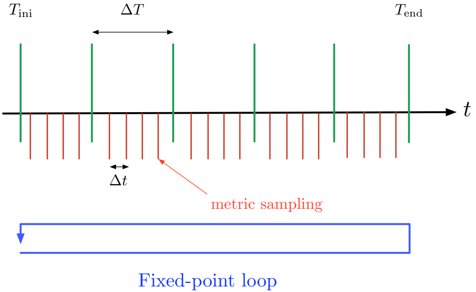
  $\Longrightarrow$ to predict the solution evolution

- Split the simulation into several time sub-intervals and set an adapted mesh for each sub-interval

  $\Longrightarrow$ to limit the number of meshes

$$[0, T] \;=\; [0 = t_0, t_1] \cup \ldots \cup [t_i, t_{i+1}] \cup \ldots \cup [t_{n_{adap}-1}, t_{n_{adap}}].$$

  $\Longrightarrow$ use of a *mean* hessian on each sub-interval



Problem

- **Continuous** space-time mesh adaptation :

$$E_{\mathbf{L}^p}(\mathbf{M}_{\mathbf{L}^p}) \quad = \quad \min_{\mathbf{M}} \left(\int_0^T \int_\Omega \mathrm{Trace}\Big(\mathcal{M}(\mathbf{x},t)^{-\frac{1}{2}} |H_u(\mathbf{x},t)| \mathcal{M}(\mathbf{x},t)^{-\frac{1}{2}} \Big)^p \,\mathrm{d}\mathbf{x}\mathrm{d}t \right)$$

- **Semi-discrete** space-time mesh adaptation with **sub-intervals**

Find $\mathbf{M}^i_{\mathbf{L}^p} = (\mathcal{M}^i_{\mathbf{L}^p}(\mathbf{x}))_{(\mathbf{x})\in\Omega}$ of space-time complexity N^i such that

$$E^i_{\mathbf{L}^p}(\mathbf{M}^i_{\mathbf{L}^p}) \quad = \quad \min_{\mathbf{M}^i} \int_\Omega \mathrm{trace}\Big((\mathcal{M}^i)^{-\frac{1}{2}}(\mathbf{x})\ \mathbf{H}^i_u(\mathbf{x})\ (\mathcal{M}^i)^{-\frac{1}{2}}(\mathbf{x}) \Big)^p \,\mathrm{d}\mathbf{x}$$

under constraint $\mathcal{C}(\mathbf{M}^i) = N^i$ with $\mathbf{H}^i_u$ defined by:

$$\mathbf{H}^i_{\mathbf{L}^1}(\mathbf{x}) = \int_{t_{i-1}}^{t_i} |H_u(\mathbf{x},t)|\,\mathrm{d}t \quad \text{or} \quad \mathbf{H}^i_{\mathbf{L}^\infty}(\mathbf{x}) = \Delta t_i \max_{t\in[t_{i-1},t_i]} |H_u(\mathbf{x},t)|\,,$$

Minimizing the Interpolation Error in $\mathbf{L}^p$-norm

Optimization problem for **time sub-intervals**, two steps resolution:

1. Spatial minimization on a sub-interval
2. Temporal minimization on a sub-interval

Spatial minimization on a sub-interval

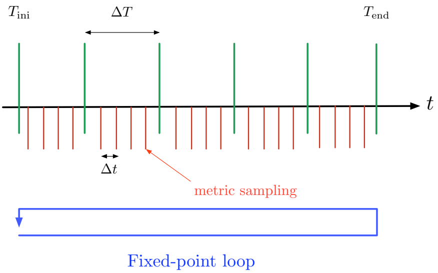
$$\mathcal{M}^i_{\mathbf{L}^p}(\mathbf{x}) = (N^i)^{\frac{2}{3}} \, (\mathcal{K}^i)^{-\frac{2}{3}} \, (\det \mathbf{H}^i_u(\mathbf{x}))^{-\frac{1}{2p+3}} \; \mathbf{H}^i_u(\mathbf{x})$$

Temporal minimization on a sub-interval for τ [Belme et al., JCP 2012]

$$\mathcal{M}^i_{\mathbf{L}^p}(\mathbf{x}) \;=\; N_{st}^{\frac{2}{3}} \left(\sum_{j=1}^{n_{adap}} (\mathcal{K}^j)^{\frac{3}{2p+3}} \left(\int_{t_{j-1}}^{t_j} \tau(t)^{-1} \mathrm{d}t \right)^{\frac{2p}{2p+3}} \right)^{-\frac{2}{3}} \left(\int_{t_{i-1}}^{t_i} \tau(t)^{-1} \mathrm{d}t \right)^{-\frac{2}{2p+3}} (\det \mathbf{H}^i_u(\mathbf{x}))^{-\frac{1}{2p+3}} \; \mathbf{H}^i_u(\mathbf{x})$$

with $\quad \mathcal{K}^i = \left(\int_\Omega (\det \mathbf{H}^i_u(\bar{\mathbf{x}}))^{\frac{p}{2p+3}} \mathrm{d}\bar{\mathbf{x}} \right)$

Fixed-Point Mesh Adaptation Algorithm [Alauzet et al., JCP 2007]



For j=1,nptfx

For i=1,nadap

- $\mathcal{S}^j_{0,i} =$ InterpolateSolution$(\mathcal{H}^j_{i-1}, \mathcal{S}^j_{i-1}, \mathcal{H}^j_i)$
- $\mathcal{S}^j_i =$ SolveState$(\mathcal{S}^j_{0,i}, \mathcal{H}^j_i)$
- $|H_{\max}|^j_i =$ ComputeHessianMetric$(\mathcal{H}^j_i, \{\mathcal{S}^j_i(k)\}_{k=1,nk})$

End for

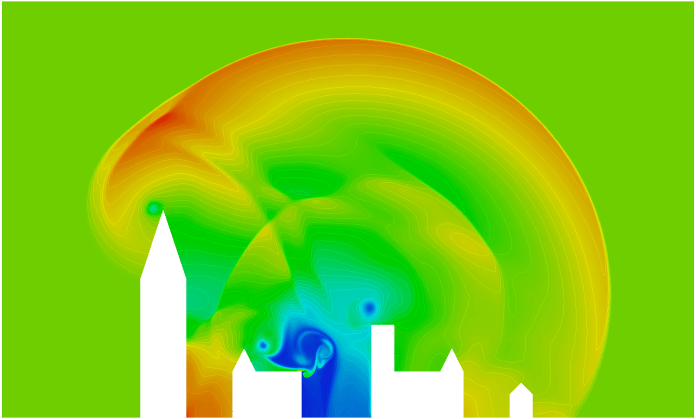
$\mathcal{C}^j =$ ComputeSpaceTimeComplexity$(\{|H_{\max}|^j_i\}_{i=1,nadap})$

$\mathcal{M}^{j-1}_i =$ ComputeUnsteadyLpMetrics$(\mathcal{C}^{j-1}, |H_{\max}|^{j-1}_i)$

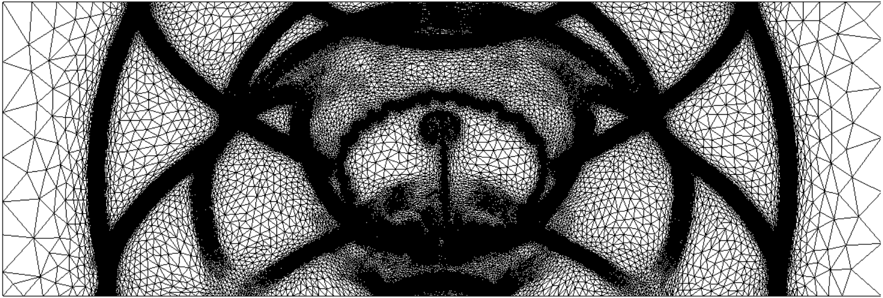
$\mathcal{H}^j_i =$ GenerateAdaptedMeshes$(\mathcal{H}^{j-1}_i, \mathcal{M}^{j-1}_i)$

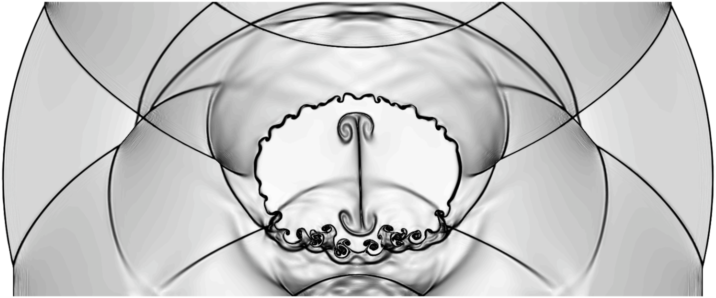
End for

Examples: First a Movie to Illustrate the Approach

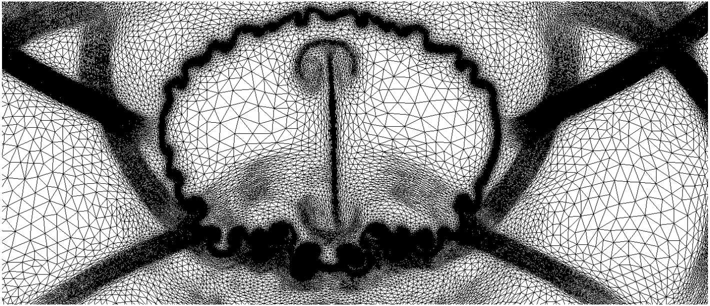


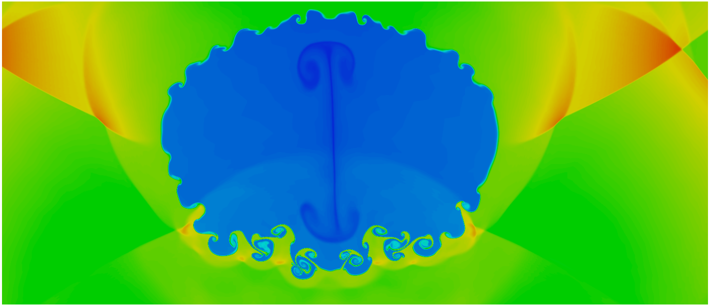
A 2D Blast (Leveque)





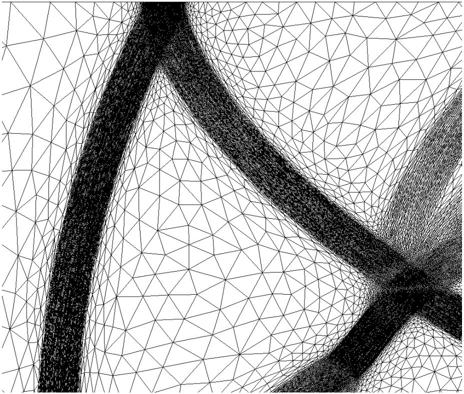
A 2D Blast (Leveque)

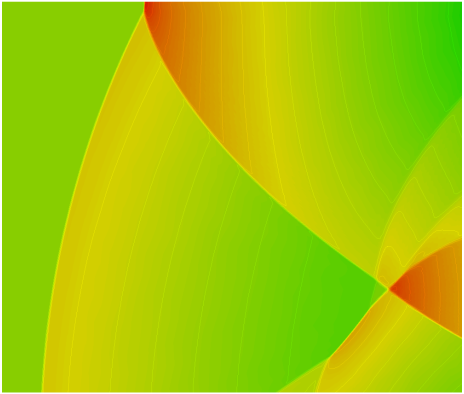




A 2D Blast (Leveque)

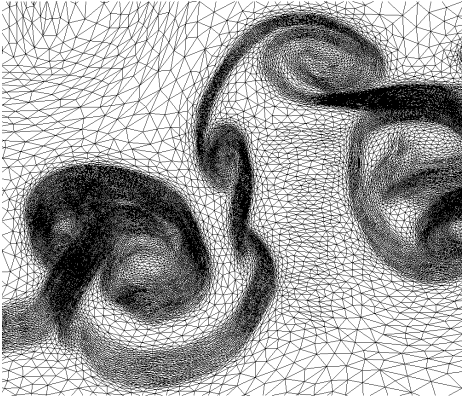
Mesh size: 182 500 vertices and 364 000 triangles

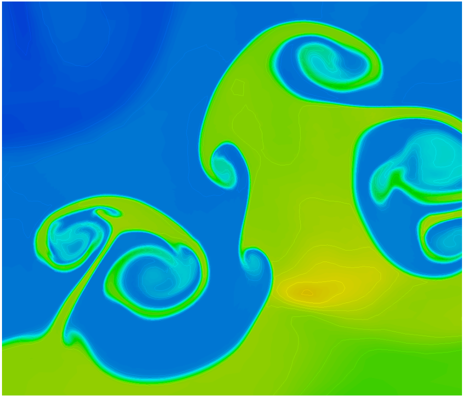




A 2D Blast (Leveque)

Mesh size: 182 500 vertices and 364 000 triangles





3D Blast in a Town

- **Problem:**
  - Bast initialization:
    high-density $(10, 0, 0, 0, 25)$ and air $(1, 0, 0, 0, 2.5)$
  - 3D town geometry $(85 \times 70 \times 70\, m^3)$

  $\Longrightarrow$ 3D *a priori* unpredictable physical phenomena

- **Simulation :**
  - 8-processors 64-bits MacPro with an IntelCore2 chipsets with a clockspeed of 2.8GHz with 32Gb of RAM
  - 40 mesh adaptations, 5 fixed point iterations, 21 metric intersection in time
  - Simultion CPU time 4h32m
    (Computation: first 9m and last 40m)

	Solver / Metric / Interpolation	Gradation / Mesh	Global
Total CPU time	2h52m	1h40m	4h32m
Percentage	63.23%	36.77%	100%

3D Blast in a Town

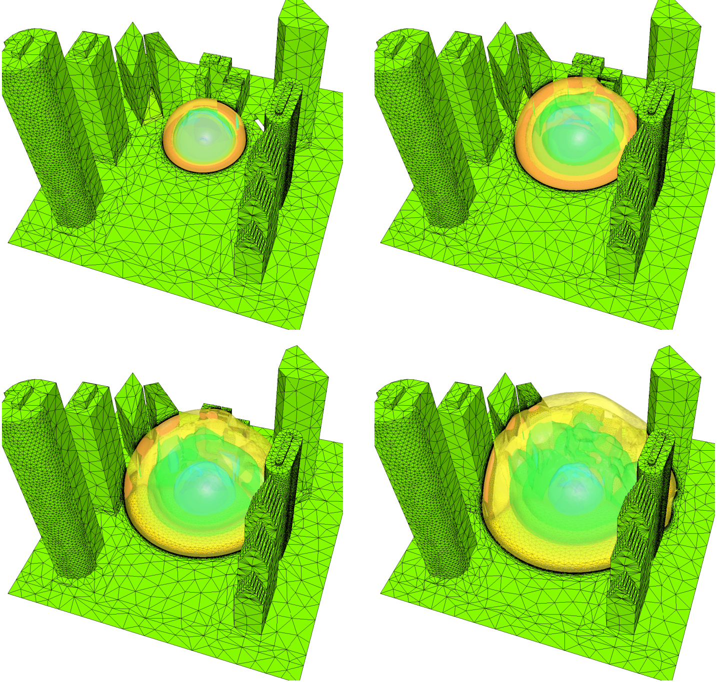
- **Problem:**
  - Bast initialization:
    high-density $(10, 0, 0, 0, 25)$ and air $(1, 0, 0, 0, 2.5)$
  - 3D town geometry $(85 \times 70 \times 70\, m^3)$

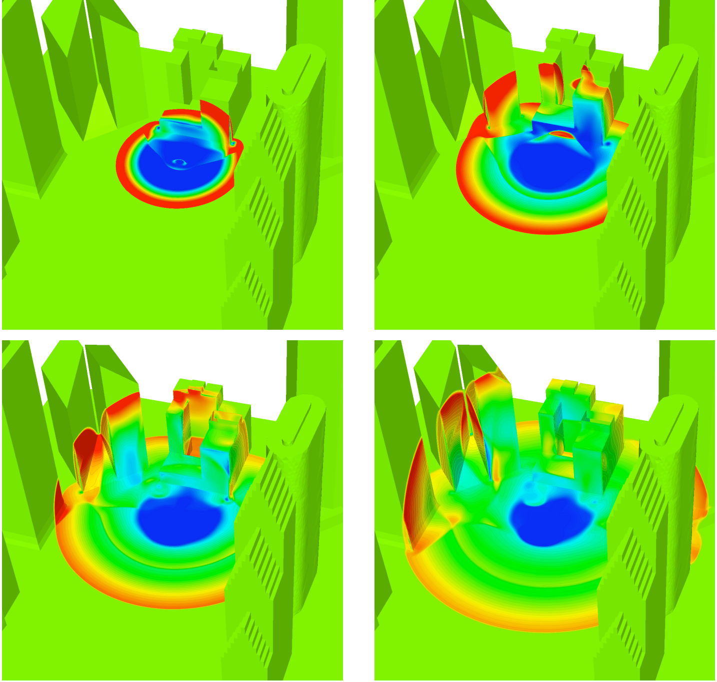
  $\Longrightarrow$ 3D *a priori* unpredictable physical phenomena

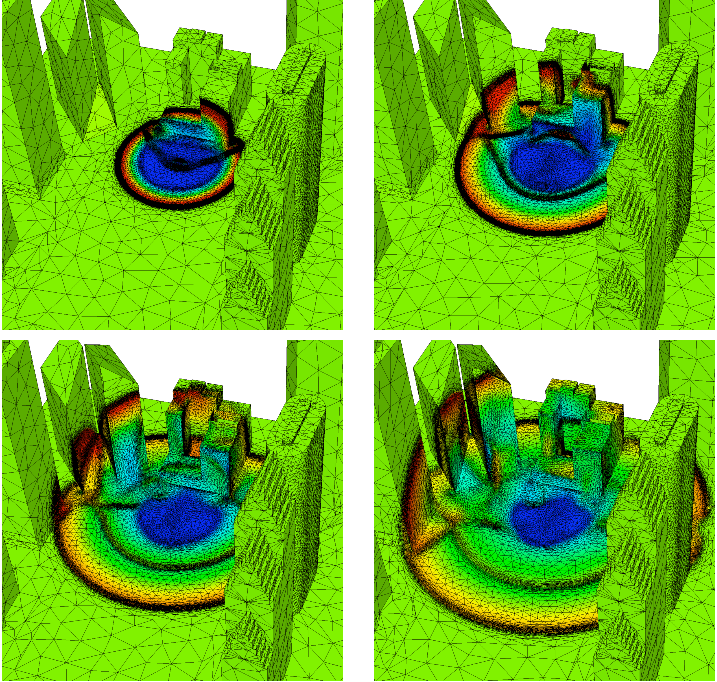
- **Adapted meshes characteristics:**

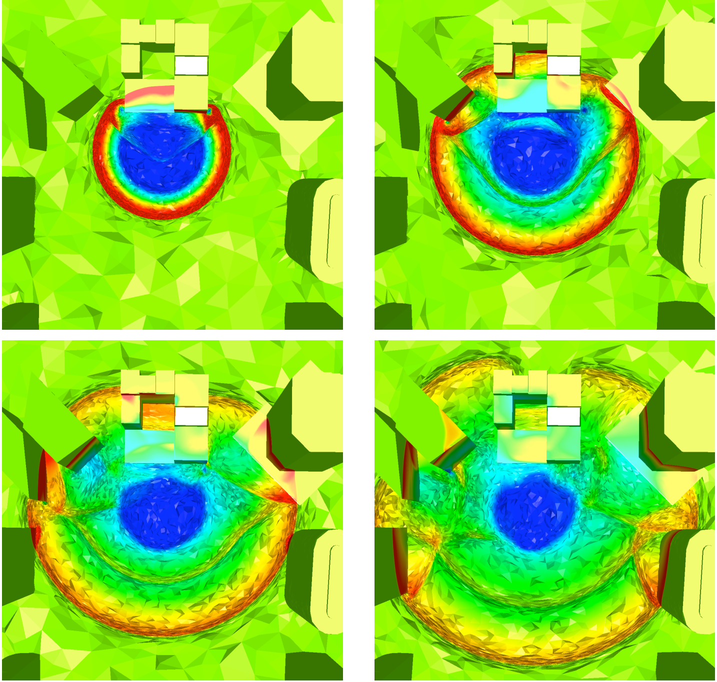
Iteration	*nv*	*nt*	*nf*	min *h*	ratio	quotient
Initial Unif.	99 255	549 128	35 420	35cm	2 (32)	4 (540)
10	305 027	1 746 040	42 486	4cm	15 (105)	164 (6804)
20	225 829	1 275 931	45 000	6cm	12 (72)	122 (3380)
30	189 858	1 057 022	48 594	9cm	9 (76)	77 (2890)
40	185 148	1 027 537	50 250	11cm	8 (71)	56 (2813)

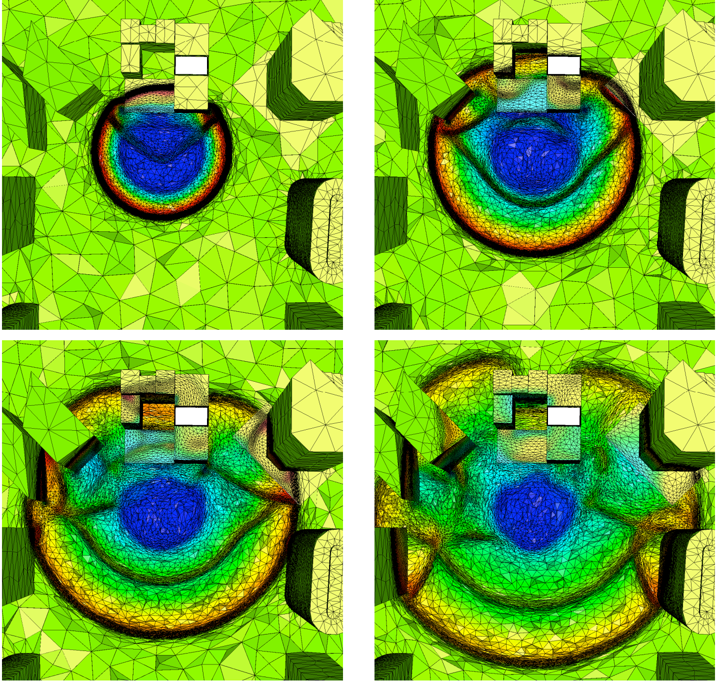
$$\text{ratio} = \sqrt{\frac{\min_i \lambda_i}{\max_i \lambda_i}} = \frac{\max_i h_i}{\min_i h_i} \quad \text{and} \quad \text{quo} = \frac{\max_i h_i^3}{h_1 h_2 h_3}$$

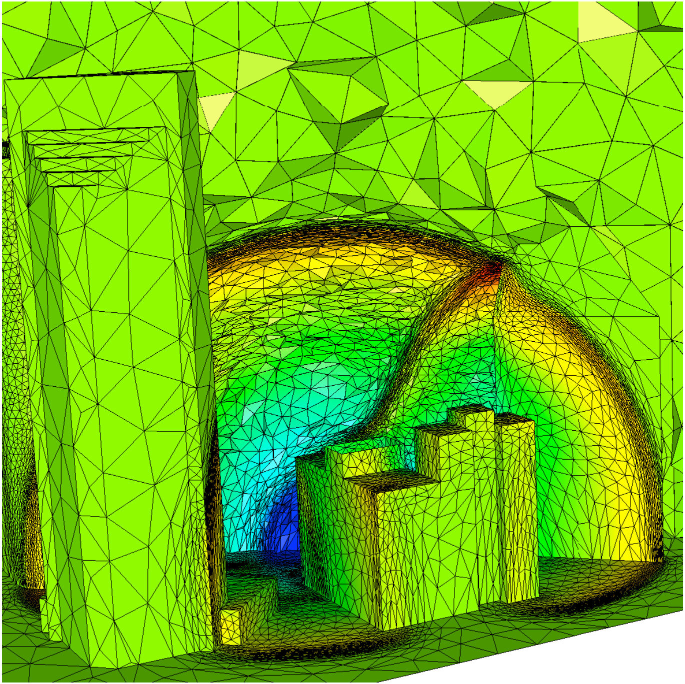












Motivation in CFD

Industrial problems

Most of the industrial problems are unsteady and involve moving geometries.

Examples of moving geometry computations:

- landing or take off of an aircraft: slat, flap, landing gear, ...
- turbo machinery, open rotor
- fluid structure interaction problems: off-shore, aeroelasticity, bioinformatics, contact problems...
- ...

Main numerical difficulty: how to handle the geometry displacement ?

Existing Approaches

Possible strategies to handle geometry displacement:

- **Embedded/immersed grid techniques**
  [Peskin, JCP 1972], [Löhner et al., IJNME 2004], ...
- **Body-fitted Chimera approach** (overlapping structured grids)
  [Benek et al., AIAA 1985], [Brezzi et al., CRAS 2001], ...

✓ **Body-fitted moving mesh techniques** (single mesh)

  - Move the mesh with a constant connectivity as far as possible and globally remesh the domain when the mesh is too distorted [Batina, AIAA 1990], [Baum and Löhner, AIAA 1997], [Hassan et al., CMAME 2000], [Degand and Farhat, C&S 2002], [Bottasso et al., CMAME 2005], [Hassan et al., IJNMF 2007]...
  - Move the mesh and optimize it with local mesh modification at each time step [Dobrzynski and Frey, IMR 2008], [Compère et al., IJNME 2010]

Chosen Approach

Goal: Coupling ALE simulations with anisotropic mesh adaptation

$\Longrightarrow$ remesh only when we want for adaptation

$\Longrightarrow$ handle efficiently anisotropic adapted meshes

$\Longrightarrow$ keep constant the number of dof

Strategy:

- relax the fixed topology constraint imposed by the ALE framework ie use only the swap and smoothing operators to move the mesh without remeshing

  $\Longrightarrow$ feasability ?

  $\Longrightarrow$ design an ALE formulation of the swap operator

- take into account the mesh movement in the adaptation

PDE-based Mesh Deformation

- Aim: assign a trajectory to inner vertices
- 2 methods tried: elasticity analogy and IDW

$\longrightarrow$ Elasticity analogy: displacement $\mathbf{d}(\mathbf{x})$ obtained by solving an elasticity problem [Baker and Cavallo, AIAA 1999]

$$\begin{cases} \mathrm{div}\,(\mathcal{S}(\mathcal{E})) = 0\,, \quad \text{with } \mathcal{E} = \dfrac{\nabla \mathbf{d} +^t \nabla \mathbf{d}}{2}, \quad \mathcal{S}(\mathcal{E}) = \lambda\, \mathrm{trace}\,(\mathcal{E})\, \mathcal{I}_n + 2\,\mu\,\mathcal{E} \\ \mathbf{d}_{|_{\partial\Omega}} \text{ prescribed by the movement of the bodies} \\ \lambda, \mu \approx \text{ very soft material} \end{cases}$$

PDE-based Mesh Deformation

- Aim: assign a trajectory to inner vertices
- 2 methods tried: elasticity analogy and IDW

$\longrightarrow$ Inverse Distance Weighted (IDW) interpolation [Luke et al., JCP 2012] :

- variant of the Radial Basis Functions (RBF) method
- speed of inner vertices: mean of the boundary vertices speeds

$$\vec{s}(\vec{r}) = \frac{\sum w_i(\vec{r})\vec{s}_i(\vec{r})}{\sum w_i(\vec{r})}$$

- boundary vertices speeds weighted by the inverse of the distance to inner vertices

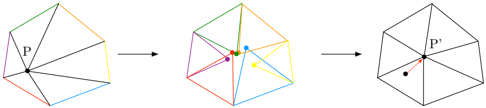
$$w_i(\vec{r}) = A_i * \left[\left(\frac{L_{def}}{\|\vec{r} - \vec{r}_i\|} \right)^a + \left(\frac{\alpha L_{def}}{\|\vec{r} - \vec{r}_i\|} \right)^b \right]$$

Improve efficiency

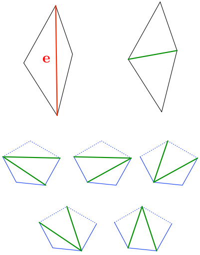
- Reduce number of elasticity resolution:
  - mesh optimisations
  - curved trajectories: each vertex is given a speed and an acceleration
- Stiffen small elements: stiffness alteration based on element Jacobian (volume)
- Rigidify regions: some inner regions are associated with a body
- Elasticity dedicated coarser mesh

Local Mesh Optimisations

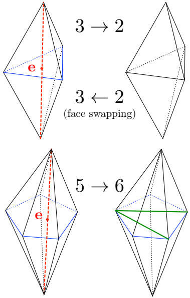
- Mesh quality criterium: $Q(K) = \frac{\sqrt{3}}{216} \left(\sum_{i=0}^{5} \ell^2_{\mathcal{M}}(\mathbf{e}_i) \right)^{\frac{3}{2}} |K|^{-1}_{\mathcal{M}} \in [1, +\infty]$
- Mesh vertices smoothing:



- Generalized face/edge swapping:



5 possible triangulations



Topology-change Moving Mesh Algorithm

Moving mesh algorithm:

$(\mathbf{v}, \mathbf{a}) = \texttt{SolveElasticities}\left(\mathbf{d}_{|\partial\Omega_h},\ dt^{els}\right)$

$\texttt{CheckMeshValidity}$

$\texttt{While}\ \left(t < t^{els} + dt^{els}\right)$

$\quad dt^{mov} = \texttt{GetMovingMeshTimeStep}\left(\mathcal{H}^k, CFL^{geom}\right)$

$\quad \mathcal{H}^k = \texttt{PerformSwaps}\left(\mathcal{H}^k, Q^{swap}_{target}\right)$

$\quad \mathbf{v}_{opt} = \texttt{PerformLaplacianSmoothing}\left(\mathcal{H}^k, Q^{smoothing}_{target},\ Q_{max}\right)$

$\quad \mathcal{H}^{k+1} = \texttt{MoveTheMesh}\left(\mathcal{H}^k, \mathbf{v}, \mathbf{v}_{opt}, \mathbf{a}, dt^{mov}\right)$

$\texttt{End while}$

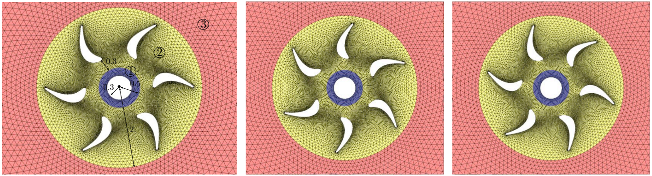
Two crossing F117s

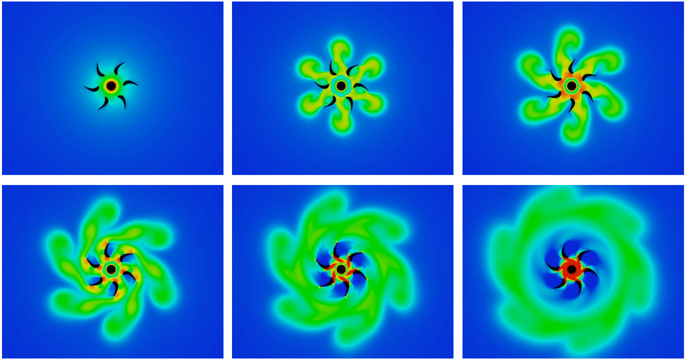
IFP engine

Coupling with an ALE solver

- Arbitrary Lagrangian Eulerian (ALE) framework : formulation of the fluid equations on a moving mesh
- Time discretization: speed of the edges chosen to be DGCL conservative
- Specific treatment for 2D topology changes: ALE swap formulation
- Loose FSI coupling

Example: A Turbopump





1. Unsteady Mesh Adaptation
2. Moving Mesh Algorithm
3. Perspectives
    - Mesh Adaptation for moving geometries
    - Adaptative mesh deformation

Mesh Adaptation for Moving Bodies Simulations

Optimal ALE metric [Alauzet and Olivier, AIAA 2011]

$$\mathcal{M}_{\mathbf{L}^p}^{n+1,ALE}[u](\mathbf{x}^n) = \left(\det\left|H_u^{n+1,*}(\mathbf{x}^n)\right|\right)^{\frac{-1}{2p+3}} \left|H_u^{n+1,*}(\mathbf{x}^n)\right|$$

with

$$H_u^{n+1,*}(\mathbf{x}^n) \; = \; \left(\det\left(\nabla^n\left[\phi\right]\left(\mathbf{x}^n\right)\right)\right)^{\frac{1}{p}} \nabla^n\left[\phi\right]\left(\mathbf{x}^n\right) \cdot H_u^{n+1}(\phi(\mathbf{x}^n)) \cdot \nabla^n\left[\phi\right]\left(\mathbf{x}^n\right)$$

and $\phi = Id + \mathbf{d}$ is the mapping between t^n and t^{n+1}

$\mathcal{M}_{\mathbf{L}^p}^{n+1,ALE}[u](\mathbf{x}^n)$ used to generate $\mathcal{H}^n$

It verifies the following properties:

- If $\mathcal{H}^{n+1}$ is the image of $\mathcal{H}^n$ by mapping ϕ. Then, $\mathcal{H}^{n+1}$ is optimal to control the interpolation error in $\mathbf{L}^p$-norm on sensor u^{n+1}
- There is no reason for $\mathcal{H}^n$ to be optimal to control the interpolation error in $\mathbf{L}^p$-norm on sensor u^n

Example: A 2D Adapted Pitching NACA0012 [Olivier and Alauzet, AIAA 2011]

Pitching NACA0012

Example: A 2D Adapted Blast Problem [Olivier and Alauzet, AIAA 2011]

Blasting a box

Adaptative Mesh Deformation

- Move the mesh following an adapted metric field

Problem

Given two metric fields at times t and $t + \Delta t$: $\mathcal{M}_t$ and $\mathcal{M}_{t+\Delta t}$, and a mesh at time t (adapted to $\mathcal{M}_t$ or not): $\mathcal{H}_t$
What is the displacement $v(x,t)$ of the mesh vertices so that $\mathcal{H}_{t+\Delta t}$ is adapted to $\mathcal{M}_{t+\Delta t}$?

- Ongoing preliminary work
- In 1D, good analytical results on polynomial metrics for moving equation:

$$m\frac{\partial v}{\partial x} + \frac{1}{2}\frac{\partial m}{\partial x}v = -\frac{1}{2}\frac{\partial m}{\partial t}$$

Conclusion

Several bricks available:

- Moving mesh algorithm
- Unsteady mesh adaptatiom
- Goal-oriented mesh adaptation
- Mesh adaptation for level set application

We're trying to combine them for more complex problems:

- Mesh adaptation for moving bodies ALE simulation
- Unsteady adaptation using moving mesh equations