Application cases of MMG library in the Kratos Multiphysics (*AKA Kratos*) framework
MMG Day 2018/2019

Vicente Mataix Ferrándiz [1] Riccardo Rossi [1] Rubén Zorrilla Martínez [1] Carlos Roig Pina[1] Alejandro Cornejo Velázquez [1] Alejandro Cornejo Velázquez [1] Marc Núñez Corbacho [1] Eugenio Oñate Ibañez de Navarra [1]

vmataix@cimne.upc.edu

[1]CIMNE. International Center for Numerical Methods in Engineering, Technical University of Catalonia (UPC). Barcelona. Spain

December 13, 2018

Overview

Section 1

Introduction

Kratos

Kratos is a framework for building multi-disciplinary finite element programs.

Features

- **KERNEL**: The kernel and application approach is used to reduce the possible conflicts arising between developers of different fields.
- **OBJECT ORIENTED**: The modular design, hierarchy and abstraction of these approaches fits to the generality, flexibility and re-usability required for the current and future challenges in numerical methods. The main code is developed in *C++* and the *Python* language is used for scripting
- **OPEN SOURCE**: The *BSD (Berkeley Software Distribution)* licence allows to use and distribute the existing code without any restriction, but with the possibility to develop new parts of the code on an open or close basis depending on the developers.
- **FREE**: Because is devoted mainly to developers, researchers and students and, therefore, is the most fruitful way to share knowledge and built a robust numerical methods laboratory adapted to their users' needs. Please, read the license for more detailed information.

Kratos structure classes

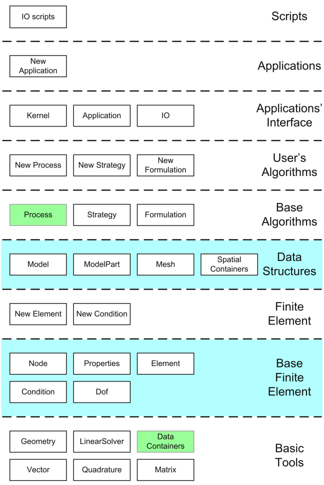


Figure 1: Kratos structure classes

Groups

- *Scripts:* Simple scripted programs created in order to reduce the workload and simplify run problems
- *Applications:* This is the base of the modularity of **Kratos**. Each application can be defined o solve an specific problem and couple them later
- *App interface (core):* Communicate each components and define the framework behaviour
- *Algorithms:* Operations that are used to solve the problem (strategies, time schemes, algorithms, etc...)
- *Data structure:* Contains the information of the problem (geometries, elements, etc...)
- *Finite element:* The base components necessaries to define a **FE** problem (DoF, elements, nodes, etc...)
- *Basic tools:* Algebraic and mathematic components

The groups and classes in **cyan** and **green** will be detailed later for being more related with the *Kratos-MMG* integration

Model

`Model` stores the whole model to be analyzed. All `Nodes`, `Properties`, `Elements`, `Conditions` and solution data

ModelPart

`ModelPart` holds all data related to an arbitrary part of model. It stores all existing components and data like `Nodes`, `Properties`, `Elements`, `Conditions` and solution data related to a part of model

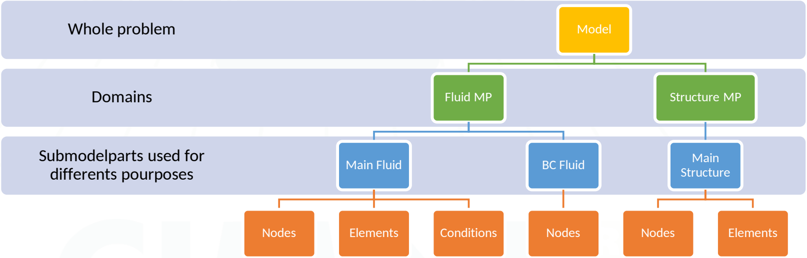


Figure 2: Example

Finite element classes

Node

`Node` It is a point with additional facilities. Stores the nodal data, historical nodal data, and list of *DoF*

Condition

`Condition` encapsulates data and operations necessary for calculating the local contributions of `Condition` to the global system of equations. *Neumann* conditions are example

Elements

`Element` encapsulates the elemental formulation in one object and provides an interface for calculating the local matrices and vectors necessary for assembling the global system of equations. It holds its geometry that meanwhile is its array of `Nodes`. Also stores the elemental data

Properties

`Properties` encapsulates data shared by different `Elements` or `Conditions`. It can store any type of data

DoF

`DoF` represents a degree of freedom (*DoF*). This class enables the system to work with different set of *DoFs* and also represents the *Dirichlet* condition assigned to each *DoF*

Section 2

Theory

Level set remeshing

We compute the gradient (1) of a scalar variable f in order to compute an anisotropic metric to remesh, using the procedure from (2)

$$\nabla f = \frac{\partial f}{\partial x}\mathrm{i} + \frac{\partial f}{\partial y}\mathrm{j} + \frac{\partial f}{\partial z}\mathrm{k} \tag{1}$$

Level set metric computation

Calling h the element size and ρ the anisotropic ratio
The scalar value f and ∇f the gradient from that scalar. $\mathcal{M}$ is the metric
We compute the following auxiliar coefficients:

$$\begin{cases} c_0 = \frac{1.0}{h^2} & \text{Isotropic metric} \\ c_1 = \frac{c_0}{\rho^2} & \text{Applying anisotropic ratio} \end{cases} \tag{2a}$$

For 2D:

$$\mathcal{M} = \begin{pmatrix} c_0(1-\nabla f_x^2) + c_1\nabla f_x^2 & (c_1 - c_0)\nabla f_x \nabla f_y \\ (c_1 - c_0)\nabla f_x \nabla f_y & c_0(1-\nabla f_y^2) + c_1\nabla f_y^2 \end{pmatrix} \tag{2b}$$

For 3D:

$$\mathcal{M} = \begin{pmatrix} c_0(1-\nabla f_x^2) + c_1\nabla f_x^2 & (c_1 - c_0)\nabla f_x \nabla f_y & (c_1 - c_0)\nabla f_x \nabla f_z \\ (c_1 - c_0)\nabla f_x \nabla f_y & c_0(1-\nabla f_y^2) + c_1\nabla f_y^2 & (c_1 - c_0)\nabla f_y \nabla f_z \\ (c_1 - c_0)\nabla f_x \nabla f_z & (c_1 - c_0)\nabla f_y \nabla f_z & c_0(1-\nabla f_z^2) + c_1\nabla f_z^2 \end{pmatrix} \tag{2c}$$

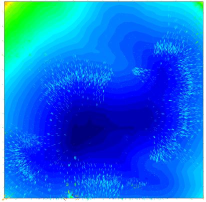


Figure 3: Scalar and its gradient

Hessian remeshing

Following a similar procedure like in the case of the level set, we can compute the hessian matrix (3) of a scalar variable f

$$\mathrm{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1\, \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n\, \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \text{ or, just: } \mathrm{H}_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j} \tag{3}$$

Hessian metric computation

Once the *Hessian* matrix has been computed we can compute the corresponding anisotropic metric by the following

$$\mathcal{M} = \mathcal{R}^t \tilde{\Lambda}^t \mathcal{R} \text{ where } \tilde{\Lambda} = (\tilde{\lambda}_i) \text{ being } \tilde{\lambda}_i = \min\left(\max\left(\frac{c_d|\lambda_i|}{\epsilon}, \frac{1}{h_{max}^2}\right), \frac{1}{h_{min}^2}\right) \tag{4a}$$

Being ϵ the error threshold and c_d a constant ratio of a mesh constant and the interpolation ratio
For an isotropic mesh the metric will be:

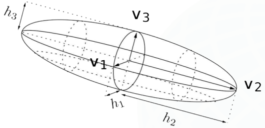
$$\mathcal{M}_{iso} = diag(\max(\tilde{\lambda}_i)) = \begin{pmatrix} \max(\tilde{\lambda}_i) & 0 & 0 \\ 0 & \max(\tilde{\lambda}_i) & 0 \\ 0 & 0 & \max(\tilde{\lambda}_i) \end{pmatrix} \tag{4b}$$

For anisotropic mesh will be:

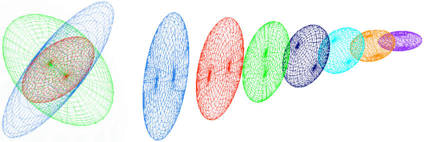
$$\mathcal{M}_{aniso} = \mathcal{R}^t \begin{pmatrix} \max(\min(\tilde{\lambda}_1, \tilde{\lambda}_{max}), R_{\lambda rel}) & 0 & 0 \\ 0 & \max(\min(\tilde{\lambda}_2, \tilde{\lambda}_{max}), R_{\lambda rel}) & 0 \\ 0 & 0 & \max(\min(\tilde{\lambda}_3, \tilde{\lambda}_{max}), R_{\lambda rel}) \end{pmatrix} \mathcal{R} \tag{4c}$$

Being $R_{\lambda rel} = |\tilde{\lambda}_{max} - R_\lambda|$ where $R_\lambda = (1-\rho)|\tilde{\lambda}_{max} - \tilde{\lambda}_{min}|$

Metric intersection



(a) Metric analogy



(b) Interpolation

The metric intersection consists in keeping the most restrictive size constraint in all directions imposed by this set of metrics[2]

Procedure

The simultaneous reduction enables to find a common basis such that $\mathcal{M}_1$ and $\mathcal{M}_2$ are congruent to a diagonal matrix, in this basis then $\mathcal{N}$ is introduced

$$\mathcal{N} = \mathcal{M}_1^{-1}\mathcal{M}_2 \text{ considering that can be decomposed in } \lambda_i = \mathrm{e}_i^t \mathcal{M}_1 \mathrm{e}_i \text{ and } \mu_i = \mathrm{e}_i^t \mathcal{M}_2 \mathrm{e}_i \tag{5a}$$

Considering $\mathcal{P} = (\mathrm{e}_1 \mathrm{e}_2 \mathrm{e}_3)$ be the matrix the columns of which are the eigenvectors of $\mathcal{N}$ (common basis)

$$\mathcal{M}_1 = \mathcal{P}^{-t} \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \mathcal{P}^{-1} \text{ and } \mathcal{M}_2 = \mathcal{P}^{-t} \begin{pmatrix} \mu_1 & 0 & 0 \\ 0 & \mu_2 & 0 \\ 0 & 0 & \mu_3 \end{pmatrix} \mathcal{P}^{-1} \tag{5b}$$

Computing the metric intersection as:

$$\mathcal{M}_{1\cap 2} = \mathcal{M}_1 \cap \mathcal{M}_2 = \mathcal{P}^{-t} \begin{pmatrix} \max(\lambda_1, \mu_1) & 0 & 0 \\ 0 & \max(\lambda_2, \mu_2) & 0 \\ 0 & 0 & \max(\lambda_3, \mu_3) \end{pmatrix} \mathcal{P}^{-1} \tag{5c}$$

Internal variables transfer

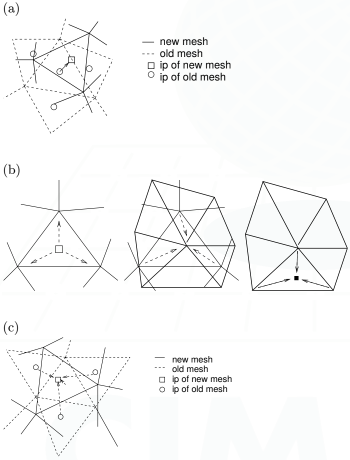


Figure 4: Transfer operators

The Figure 4 shows graphically how each one of the transfer methods work

Techniques

- *CPT*: Closest Point Transfer. **(a)**

  It just takes the value from the closest point

  It provides acceptable results at low cost

- *SFT*: Shape Function Projection transfer. **(b)**

  It interpolates the values using the standard **FEM** shape functions

  Leads to an artificial damage diffusion, but preserves the original shape of the damage profile

- *LST*: Least-Square Projection transfer. **(c)**

  It uses an *least-square* transfer across the closest points

  It is probably the most accurate technique but computationally more expensive

Some example will be shown following

Numerical contact remeshing (I)

NOTE: Work of *Anna Rehr* from TUM

Error estimation[8]

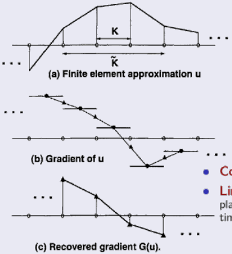
Residual based methods

- **Internal residual** (r): Error in the differential equation
- **Boundary error** (R):
  - *Traction boundaries*: Difference stress and traction
  - *Interelement boundaries*: Stress jumps
  - *Contact boundary*: Difference stress and contact pressure

$$\|e\|_{E,K} \approx \|\hat{e}\|_{E,K} = C[h_K^2\|r\|_{L_2(K)} + h_K^2\|R\|_{L_2(\partial K)}]$$

- Sound mathematical error bounds
- Determination of the constant C not trivial
- Quality depends heavily on the chosen constant

Recovery based methods



- **Constant stress FE solution**
- **Linear recovered solution:** Replaces the exact solution in the error estimation

$$\|e\|_{E,K} \approx \|\hat{e}\|_{E,K} = \left[\int_{\Omega_K} (\boldsymbol{\sigma}^* - \boldsymbol{\sigma}_h)^T D^{-1} (\boldsymbol{\sigma}^* - \boldsymbol{\sigma}_h)\right]^{\frac{1}{2}} d\Omega_K$$

- Robust
- Easy implementation, no unknown constant, easy extensibility
- No sound mathematical error bounds

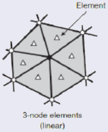
NOTE: Work of *Anna Rehr* from **TUM**

The present work introduces a modified version of the **SPR**[8] method

Modified SPR

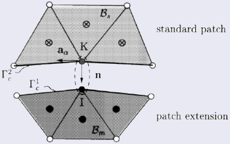
Element patch:

All elements that are neighboring one node



- **Concept**:
  - Stresses at the integration points show superior convergence behavior
  - Use these points to compute the superior stress field
- **Procedure**:
  - Execute a polynomial least square fit with the integration points
  - Compute with this polynomial recovered stress at the center node
  - Compute the stress field by interpolation with the shape functions

Extension for contact mechanics



- **Existing approach (penalty formulation)**:
  - Couple patches at the contact boundary
  - Enforce stress continuity in the recovery procedure by a penalty formulation
- **Anna Rehr's work**:
  - Patch coupling not necessary: contact pressure is known (Lagrange Multiplier)
  - Contact BC are regarded in the recovered stress calculation by a penalty formulation which forces the stresses to coincide with the contact pressure

Section 3

Cases

Coarse sphere

In this problem we remesh using the gradient of the distance function, which is the distance to the plane contained in the sphere center.

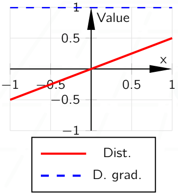


Figure 5: Distance function

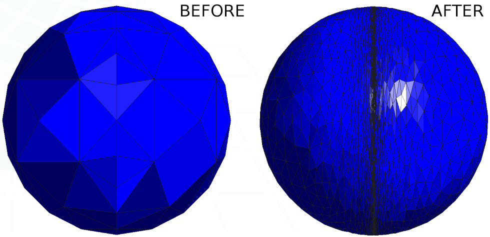


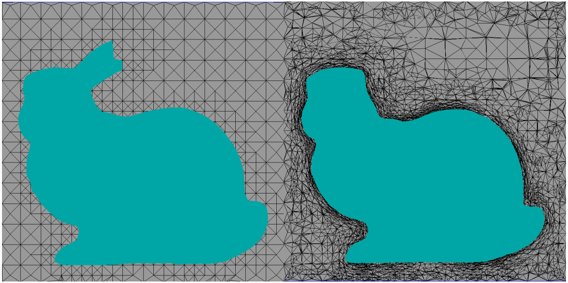
Figure 6: Mesh before and after remeshing

The function can be seen in the Figure5

Stanford's bunny

Geometry of the *Standford*'s bunny[4].

Figure 7: Standford's bunny



(a) Octree mesh (b) Anisotropic mesh

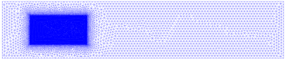
Figure 8: Mesh before and after remeshing

Anisotropically remesh the geometry using the distance gradient as error measure .
Previously meshed with an embedded octree mesher (*GiD*).

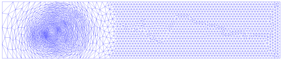
Embedded fluid channel 2D

Adaptative anisotropic remeshing of 2D fluid channel with sphere using as level set the distance function. The problem is solved using an embedded formulation
It consists in a channel 5x1, a sphere of 0.3 diameter and with a velocity of 1 m/s in the inlet an zero pressure in the outlet. The resulting flow has *Reynolds* number of **100**.

(a) Problem

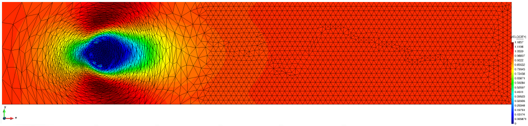


(b) Initial mesh

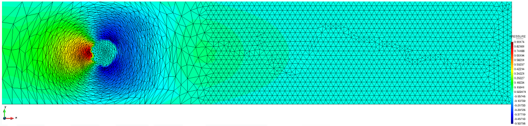


(c) Remeshed

Figure 9: Setup



(a) Velocity

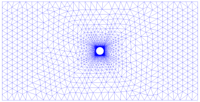


(b) Pressure

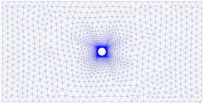
Potential fluid simulation 2D. Sphere

NOTE: Work of *Marc Núñez Corbacho* from **CIMNE**

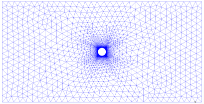
Adaptative anisotropic remeshing of 2D fluid channel with sphere using as level set the distance function. The problem is solved using a potential fluid formulation



(c) No remesh

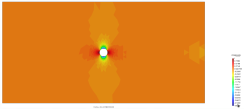


(d) Remesh 0.01

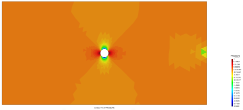


(e) Remesh 0.005

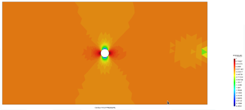
Figure 10: Meshes



(a) No remesh

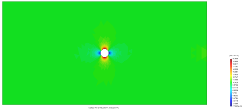


(b) Remesh 0.01

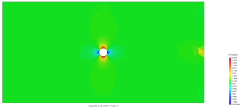


(c) Remesh 0.005

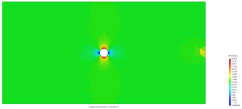
Figure 11: Pressure



(a) No remesh



(b) Remesh 0.01



(c) Remesh 0.005

Figure 12: Velocity

Potential fluid simulation 2D. NACA 12 (I)

NOTE: Work of *Marc Núñez Corbacho* from **CIMNE**

Adaptative anisotropic remeshing of 2D fluid channel with a NACA 12 aerofoil using as level set the distance function. The problem is solved using a potential fluid formulation

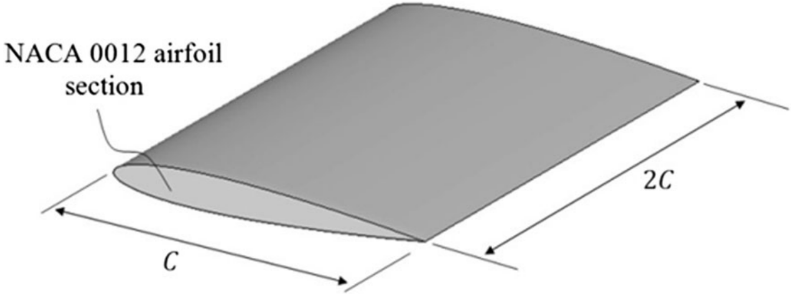
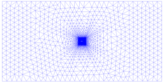
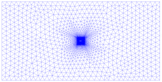


Figure 13: NACA 12 aerofoil

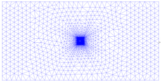
Potential fluid simulation 2D. NACA 12 (II)



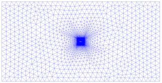
(a) No remesh 0°



(b) Remesh 0°

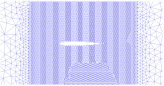


(c) No remesh 5°

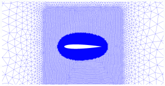


(d) Remesh 5°

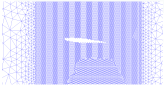
Figure 14: Meshes



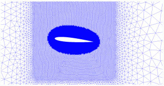
(a) No remesh 0°



(b) Remesh 0°

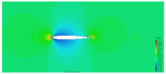


(c) No remesh 5°

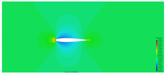


(d) Remesh 5°

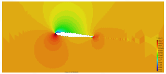
Figure 15: Detail



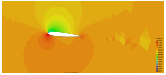
(a) No remesh 0°



(b) Remesh 0°

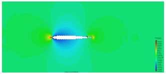


(c) No remesh 5°

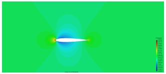


(d) Remesh 5°

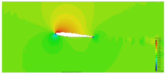
Figure 16: Pressure



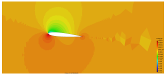
(a) No remesh 0°



(b) Remesh 0°



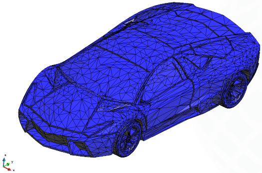
(c) No remesh 5°



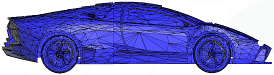
(d) Remesh 5°

Figure 17: Velocity

Lamborghini



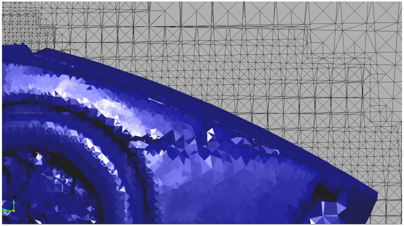
(a) View 1



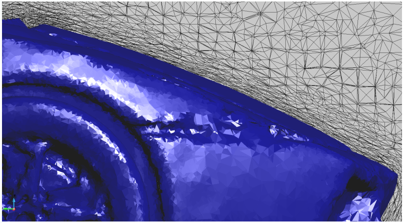
(b) View 2

Figure 18: Lamborghini

In this test case we want to remesh anisotropically the geometry of *Lamborghini*, more complex that the previous bunny.
Anisotropically remesh the geometry using the distance gradient as error measure.
Previously meshed with an embedded octree mesher (*GiD*).



(a) Octree mesh



(b) Anisotropic mesh

Figure 19: Mesh before and after remeshing

Hessian 2D

The problem corresponds with the example proposed in reference[2]

The objective is to remesh the structured 1x1 mesh with the error function from Figure 20 and equation (6)

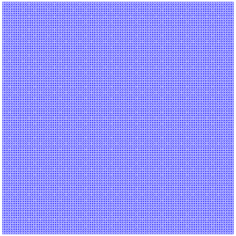
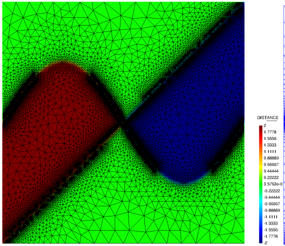
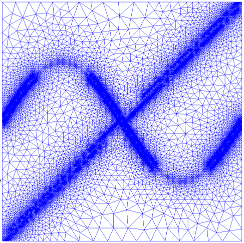


Figure 20: Initial mesh



(a) Error estimation



(b) New mesh

Figure 21: Solution

The χ shaped function:

$$f(x,y) = \tanh(-100(y - 0.5 - 0.25\sin(2\pi x))) + \tanh(100(y - x)) \quad (6)$$

Hessian 3D

An extension of the previous problem to 3D in several remeshing iterations

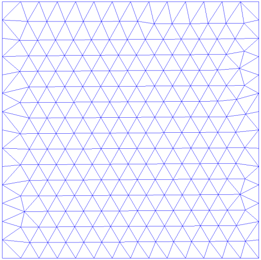
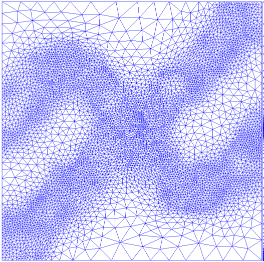
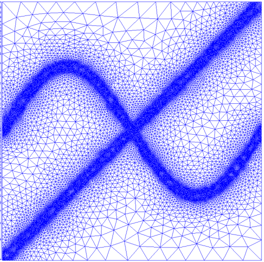


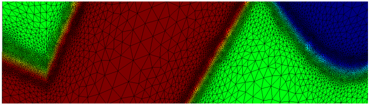
Figure 22: Initial mesh



(a) Iteration 1



(b) Iteration 2

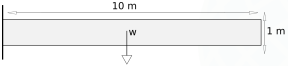


(c) Error estimation

Figure 23: Solution

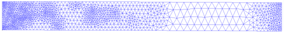
Beam 2D

The simulation considers 100 time steps of 0.01s. The problem will be remeshed each ten steps considering the Hessian of the displacement

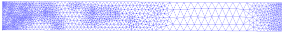


(a) Problem

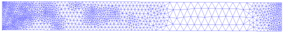
(b) Initial mesh



(c) Mesh 1

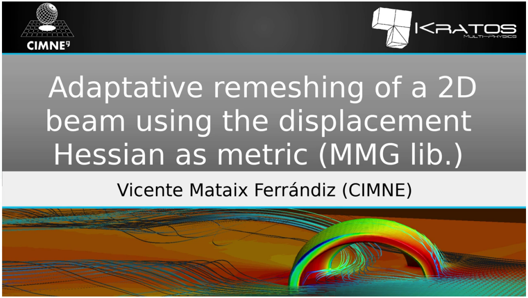


(d) Mesh 2



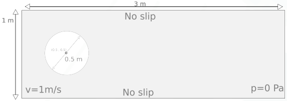
(e) Mesh 3

Figure 24: Setup

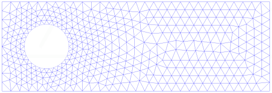


Fluid channel 2D

Adaptative remeshing of 2D fluid channel with sphere using Hessian of velocity as metric measure. It consists in a channel 3x1, a sphere of 0.5 diameter and with a velocity of 1 m/s in the inlet an zero pressure in the outlet.

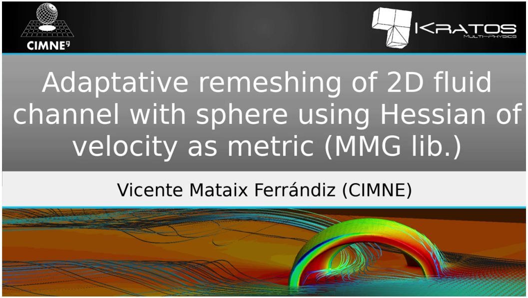


(a) Problem



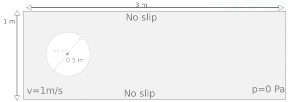
(b) Initial mesh

Figure 25: Setup

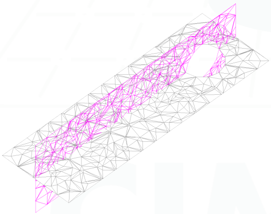


Fluid channel 3D

Adaptative remeshing of 3D fluid channel with sphere using Hessian of velocity as metric measure. It consists in a channel 3x1x1, a sphere of 0.5 diameter and with a velocity of 1 m/s in the inlet an zero pressure in the outlet.

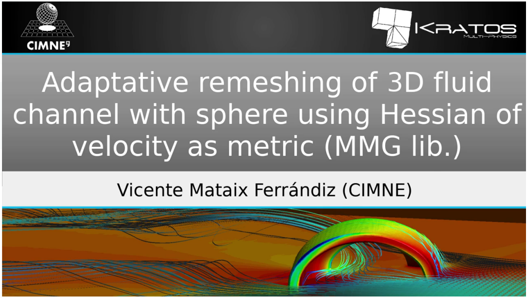


(a) Problem



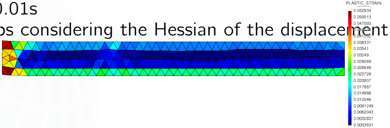
(b) Initial mesh

Figure 26: Setup

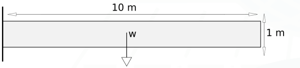


Beam 2D

The simulation considers 10 time steps of 0.01s
The problem will be remeshed each ten steps considering the Hessian of the displacement
A J2-plasticity law has been considered

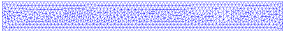


(a) Initial mesh



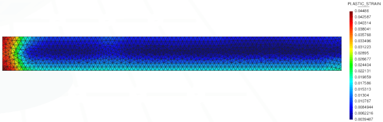
(a) Problem

(b) Initial mesh

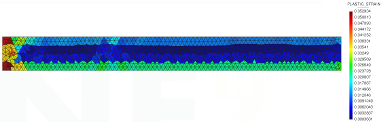


(c) After remeshing

Figure 27: Setup



(b) LST



(c) CPT

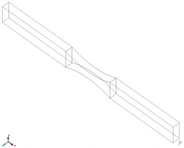
Figure 28: Plastic strain

Tensile test 3D

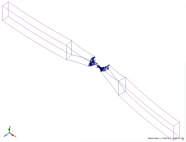
NOTE: Work of *Alejandro Corbacho Velázquez* from **CIMNE**

The problem will be remeshed each ten steps considering the Hessian of the equivalent stress at nodes. A damage law has been considered

(a) Problem



(b) Initial conf.



(c) Final conf.

Figure 29: Tensile test

Patch test

NOTE: Work of *Anna Rehr* from **TUM**

The patch test is the most absic test to pass to verify a contact formulation
It has been solved in 2D using the modification of the **SPR**

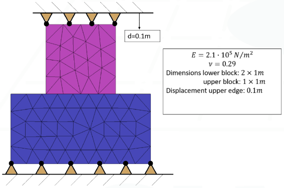


Figure 30: Setup

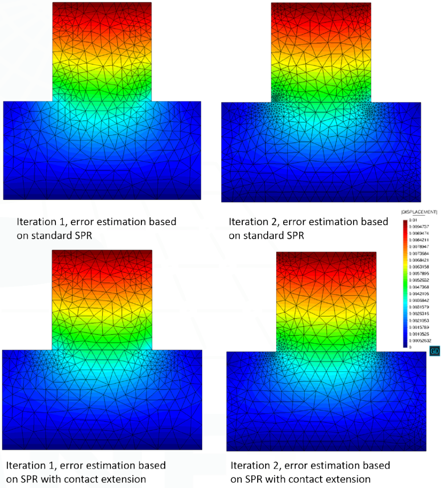


Figure 31: Solution

Hertz problem

NOTE: Work of *Anna Rehr* from **TUM**

The *Hertz* test is a very used benchmark for contact mechanics
It has been solved both in 2D and 3D using the modification of the **SPR**

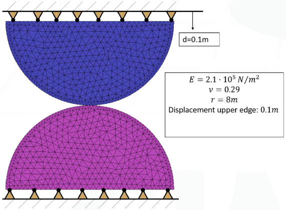
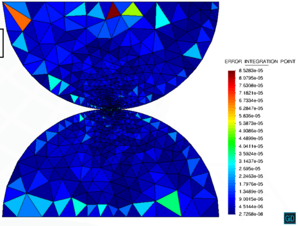
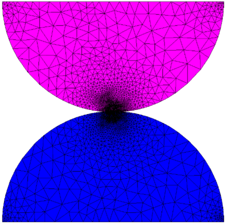


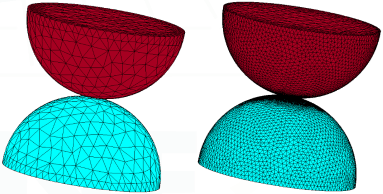
Figure 32: Setup



(a) Error



(b) Final mesh 2D



(c) Final mesh 3D

Section 4

Conclusions

Conclusions and future work

Conclusions

- *MMG:* We have implemented into *Kratos* the *MMG* **API**
- *Metrics*: We have implemented several metric measures and utilities
- *Internal values*: We have implemented several utilities in order to interpolate internal values
- *Contact*: We have implemented an extended version of **SPR** in order to be able to compute contact remeshing
- *Problems*: We have used the library in a set of different problems (fluid, structural analysis, potential flow)

Future works

- *New problems*: Use the process for new problems not computed yet (for example *FSI*)
- *Extend*: Extend the *Kratos*/*MMG* integration
- *Paralellization*: Using *ParMMG*. Part of the *ExaQUte* project

Visit us at GitHub

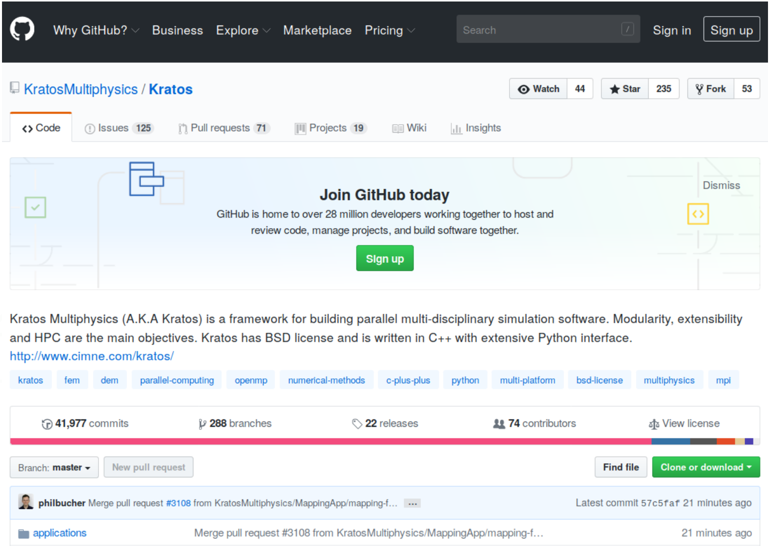


Figure 33: https://github.com/KratosMultiphysics/Kratos

Many thanks

(a) International Center for Numerical Methods in Engineering

(b) Chair of Structural Analysis Technical University of Munich

Figure 34: Main contributors

(a) Airbus Defence and Space Stress Methods Department

(b) Siemens AG Corporate Optimisation Technology

(c) ONERA, The French Aerospace Lab Applied Aerodynamics Department

Figure 35: Known Users

And many thanks to our community!!

Many thanks

Figure 36: EXAscale Quantification of Uncertainties for Technology and Science Simulation

And many thanks to the *ExaQUte* project!!!

References

- *P. Dadvand, R. Rossi, E. Oñate*: *An Object-oriented Environment for Developing Finite Element Codes for Multi-disciplinary Applications.* Computational Methods in Engineering. 2010
- *F. Alauzet*: *Metric-Based Anisotropic Mesh Adaptation.* Course material, CEA-EDF-INRIA Schools. Numerical Analysis Summer School. 2007
- *P. Tremblay*: *2-D, 3-D and 4-D Anisotropic Mesh Adaptation for the Time-Continuous Space-Time Finite Element Method with Applications to the Incompressible Navier-Stokes Equations.* PhD thesis Ottawa-Carleton Institute for Mechanical and Aerospace Engineering, Department of Mechanical Engineering, University of Ottawa. 2007
- *G.Turk, M. Levoy*: *The Stanford 3D Scanning Repository.*
- *P.J. Frey, F. Alauzet*: *Anisotropic mesh adaptation for CFD computations.* Comput. Methods Appl. Mech. 2004
- *P.J. Frey, F. Alauzet*: *Anisotropic mesh adaptation for transient flows simulations*
- *M. Bellet*: *Adaptive mesh technique for thermal–metallurgical numerical simulation of arc welding processes*, International Journal for Numerical Methods in Engineering, 2008.
- *Zienkiewicz, O. C. and Zhu, J.Z. and Taylor, Robert L.*: *The Finite Element Method: its Basis and Fundamentals*, Butterworth-Heinemann, 2013.
- *P.J. Frey, F. Alauzet*: *Estimateur d'erreur geometrique et metriques anisotropes pour l'adaptation de maillage. Partie I : aspects theoriques*, HAL Id: inria-00071827, 2006

Thank you very much for your attention