New unstructured tetrahedra octree-based mesher

Abel Coll Sans

7th GiD Convention on Advances and Applications of GiD Workshop, July 17th 2014

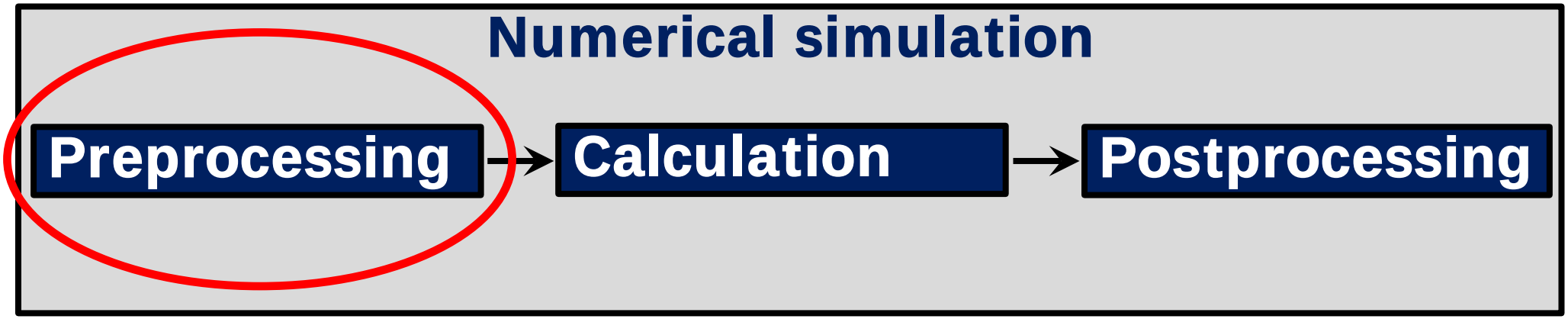
Outline

- Introduction
- Mesher characteristics
- Meshing algorithm step by step
- Examples
- How to use it inside GiD

Outline

- Introduction
- Mesher characteristics
- Meshing algorithm step by step
- Examples
- How to use it inside GiD

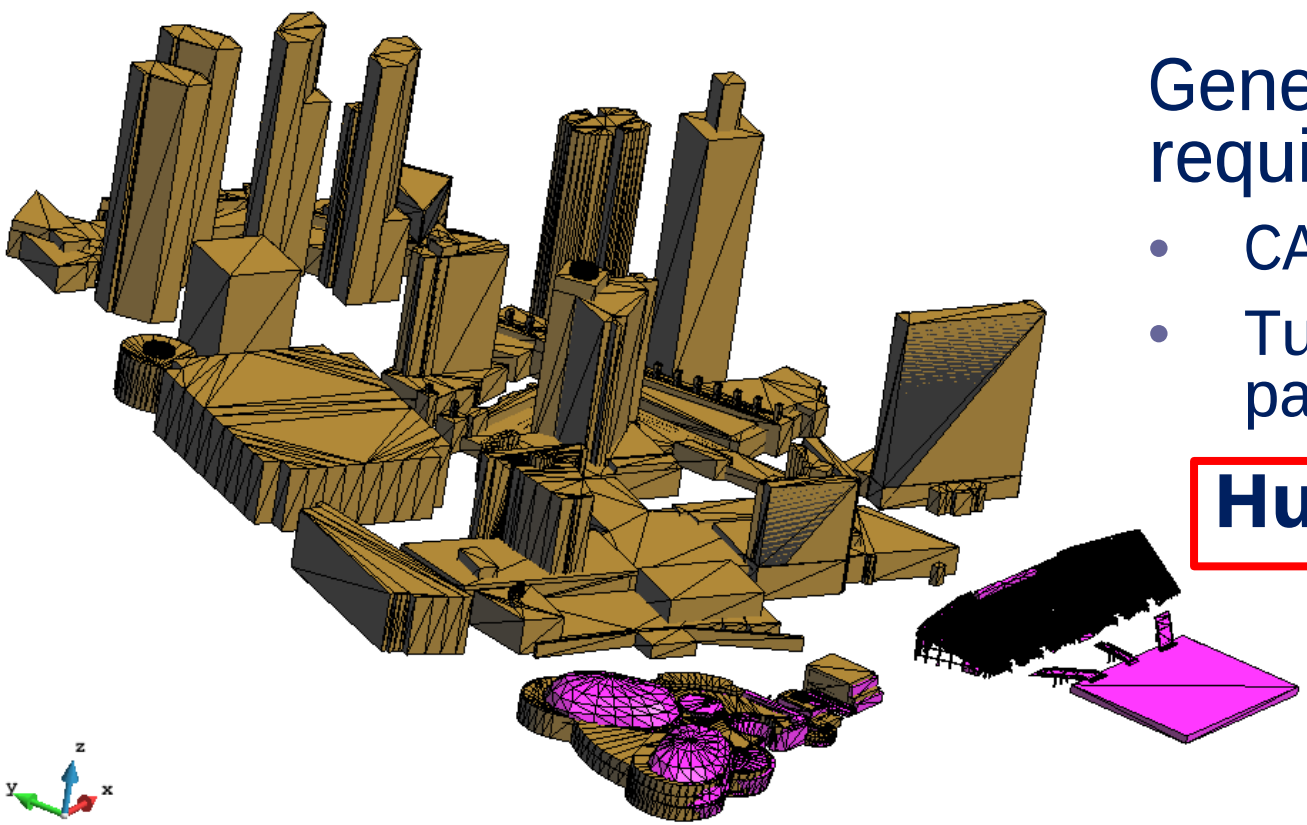
Motivation



The **most time consuming** part of a numerical simulation (in complex industrial problems) is the **preprocessing** part

Motivation

Complex geometries for industrial problems are really difficult to be meshed. Input boundaries often non-watertight.

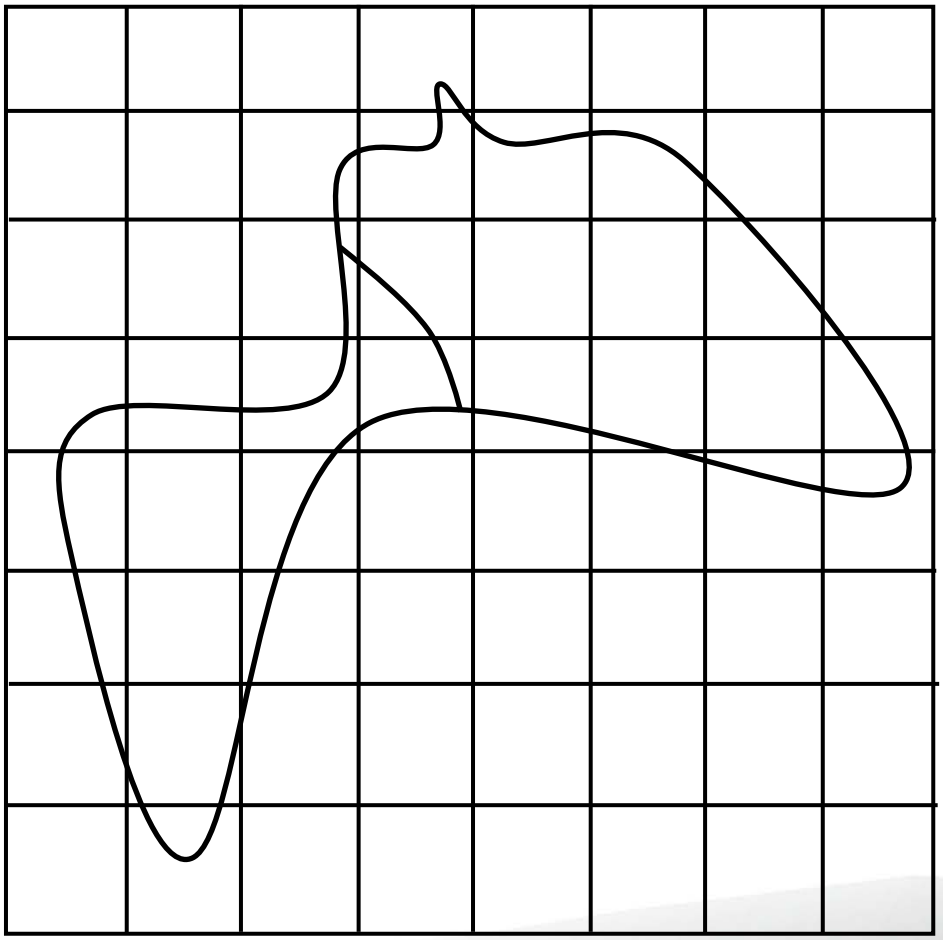


Generate the mesh often requires:

- CAD cleaning operations
- Tunning of mesher parameters

Human interaction

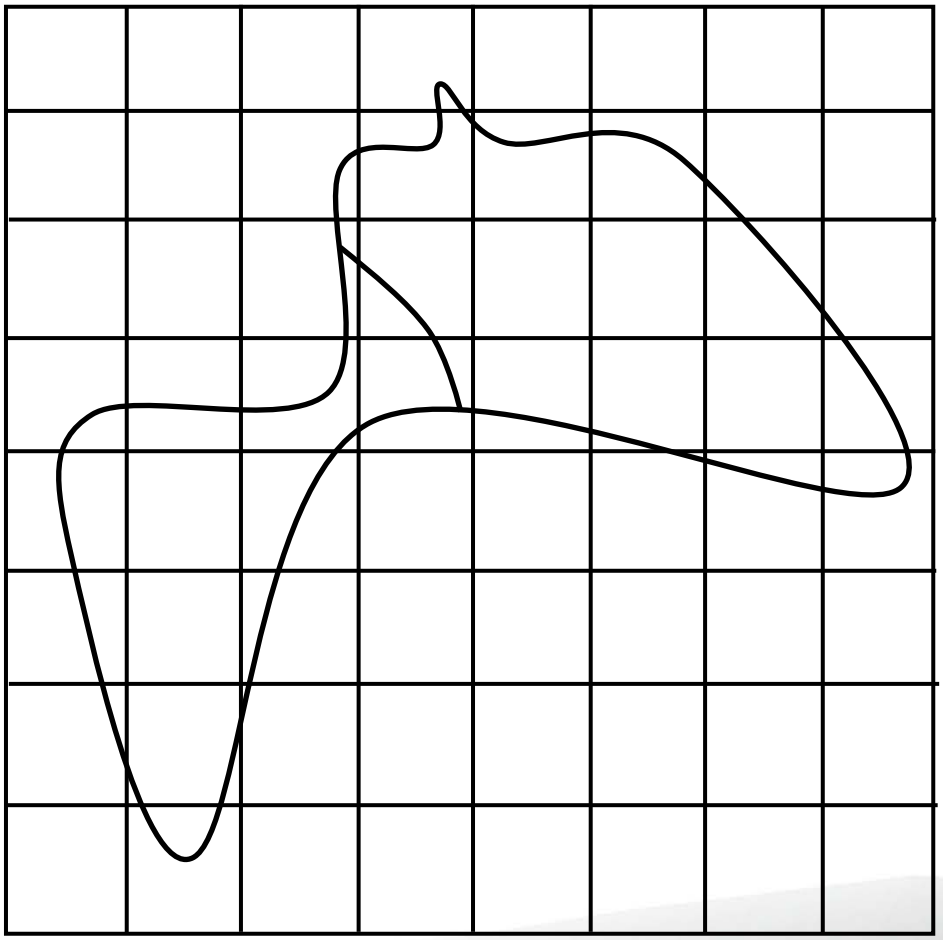
General idea: 3 main steps



1. Generate a regular grid for the space decomposition enclosing the model

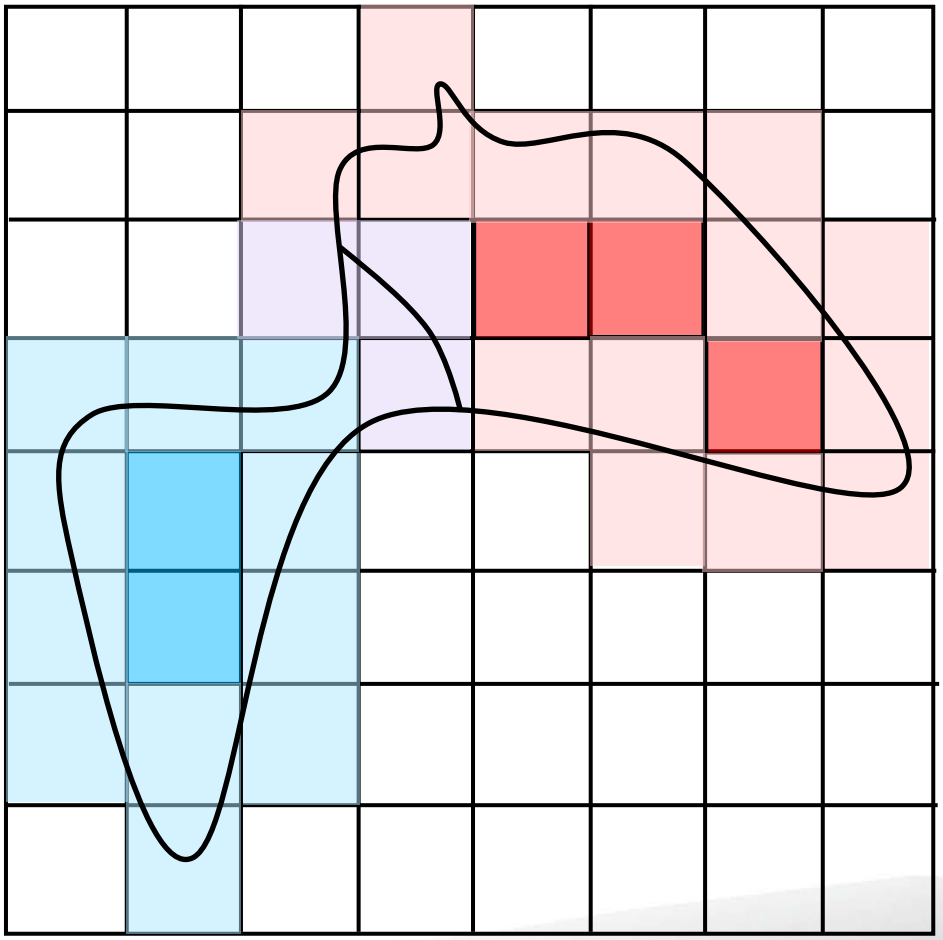
Domain decomposition methods

General idea: 3 main steps



1. Generate a regular grid for the space decomposition enclosing the model
2. Generate the elements of the final mesh from given patterns from the regular grid

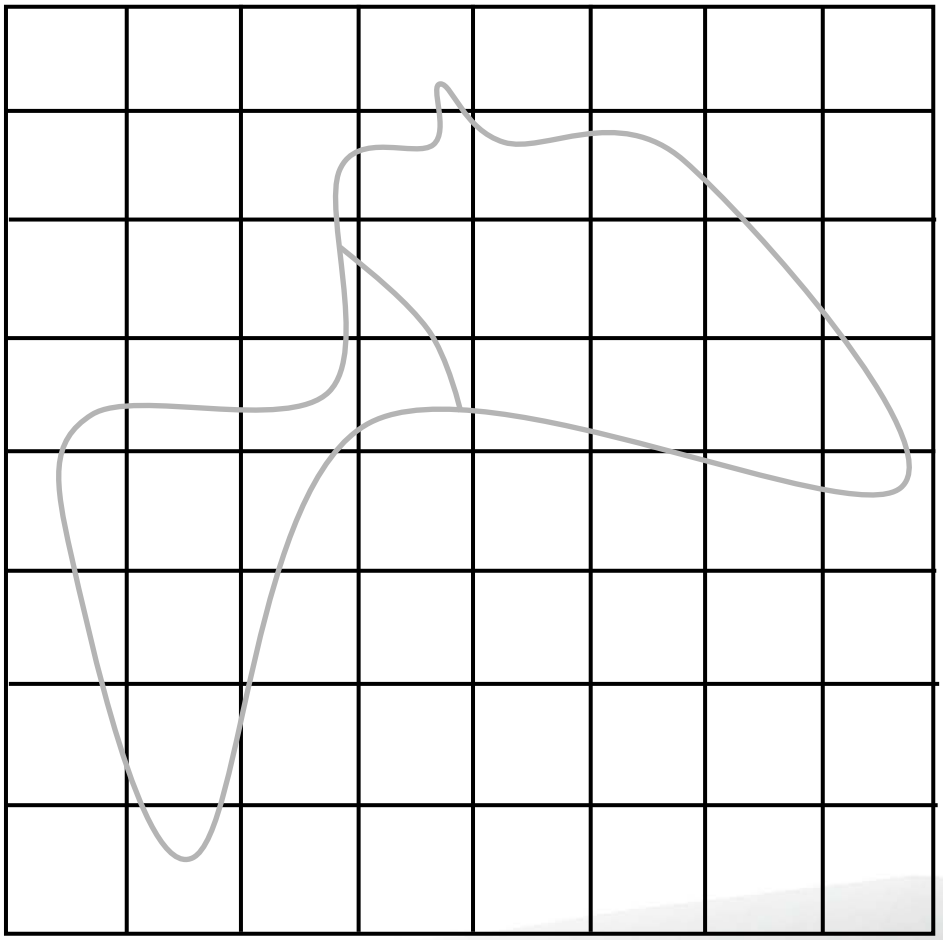
General idea: 3 main steps



1. Generate a regular grid for the space decomposition enclosing the model
2. Generate the elements of the final mesh from given patterns from the regular grid
3. Fit ‘somehow’ the boundaries of the domain

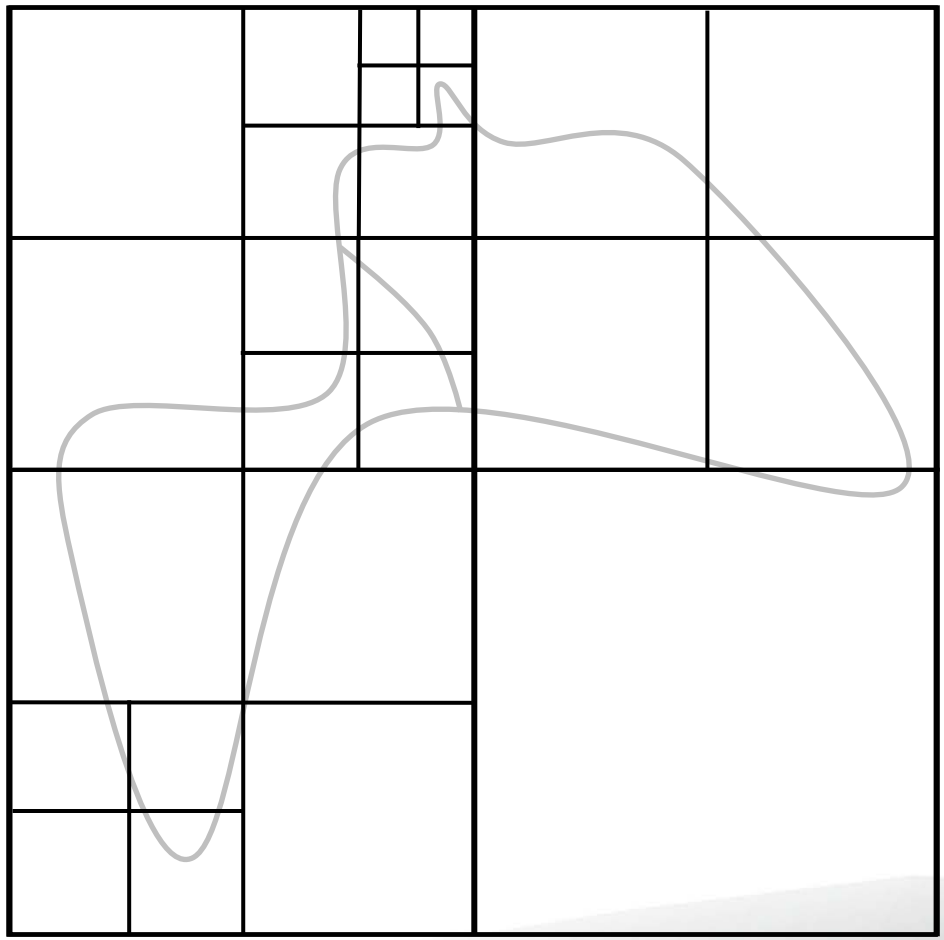
Domain decomposition methods

Two main families of decomposition patterns



- Bin

Two main families of decomposition patterns



- Octree

Domain decomposition methods

Advantages

- Robust
- Fast
- Naturally parallelizable
- Allow not cleaned geometry as input

Drawbacks

- Naturally not constrained
- Hard to preserve geometric features
- Hard to preserve topology
- Restricted size transitions
- Too refined meshes in some regions

Proposed solution

Octree-based mesher with specific techniques to deal with:

Advantages

- Robust
- Fast
- Naturally parallelizable
- Allow not cleaned geometry as input

Drawbacks

- Naturally not constrained
- Hard to preserve geometric features
- Hard to preserve topology
- Restricted size transitions
- Too refined meshes in some regions

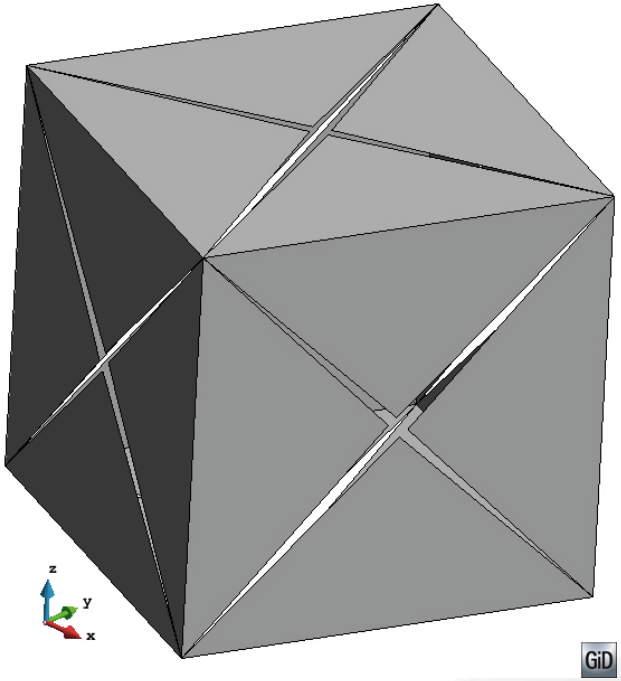
Outline

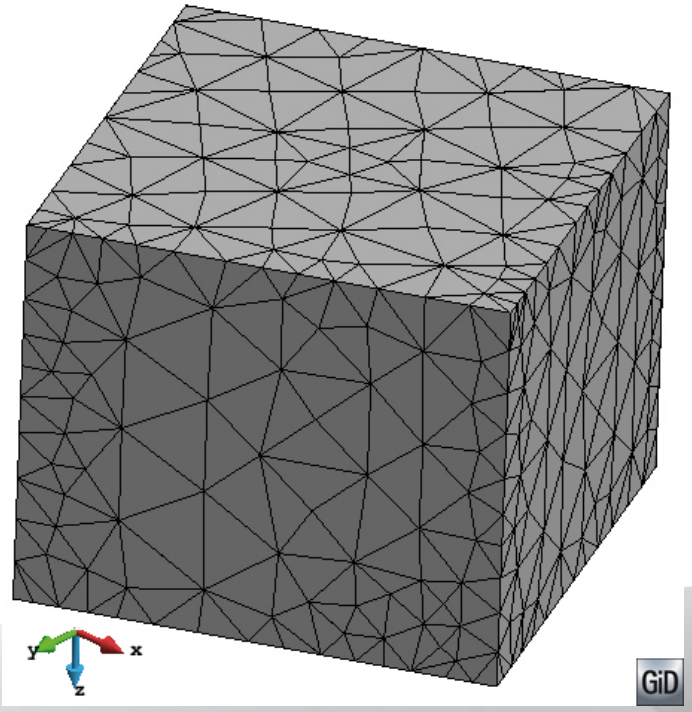
- Introduction
- Mesher characteristics
- Meshing algorithm step by step
- Examples
- How to use it inside GiD

Mesher characteristics

The mesher has been designed and implemented to be very **robust** and **fast**, with the following characteristics:

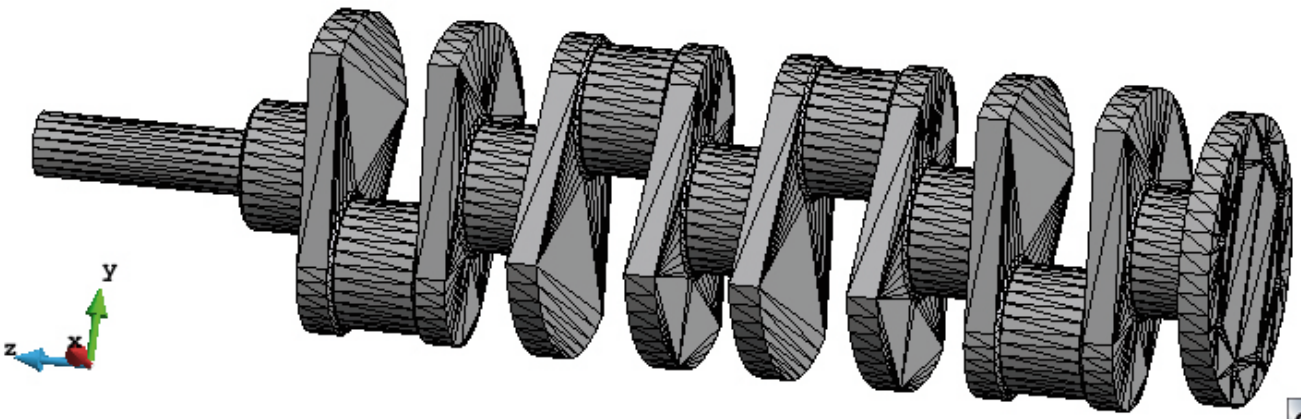
- Capable to deal with non-watertight geometries

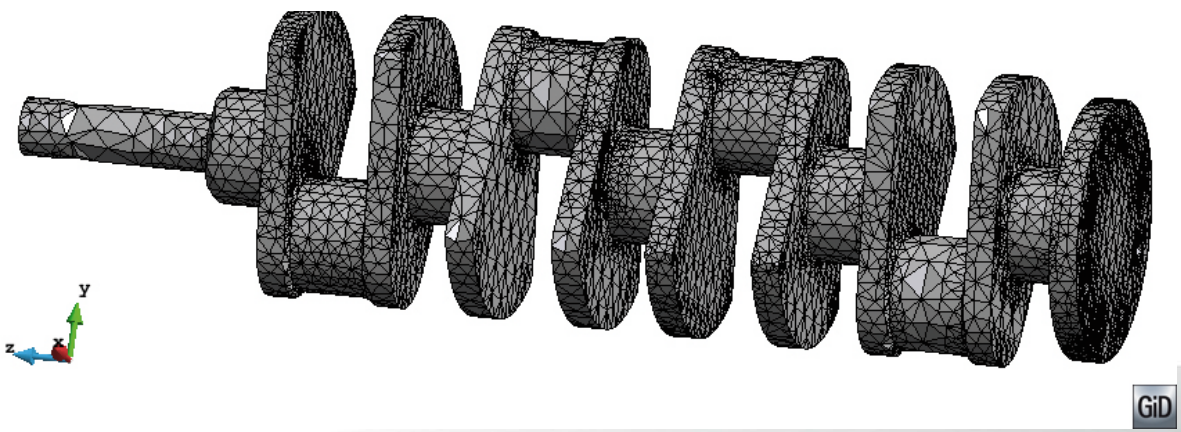


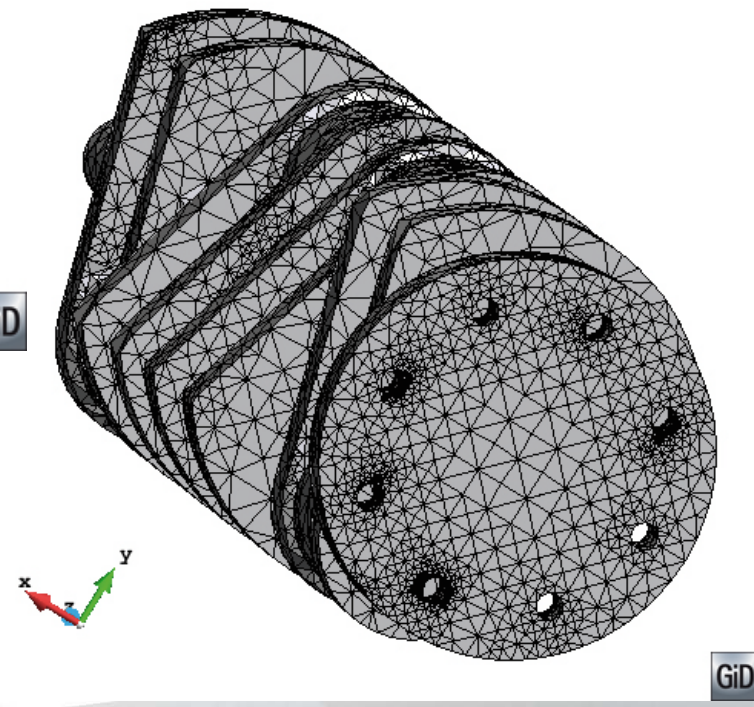


Mesher characteristics

- Geometrical features preservation

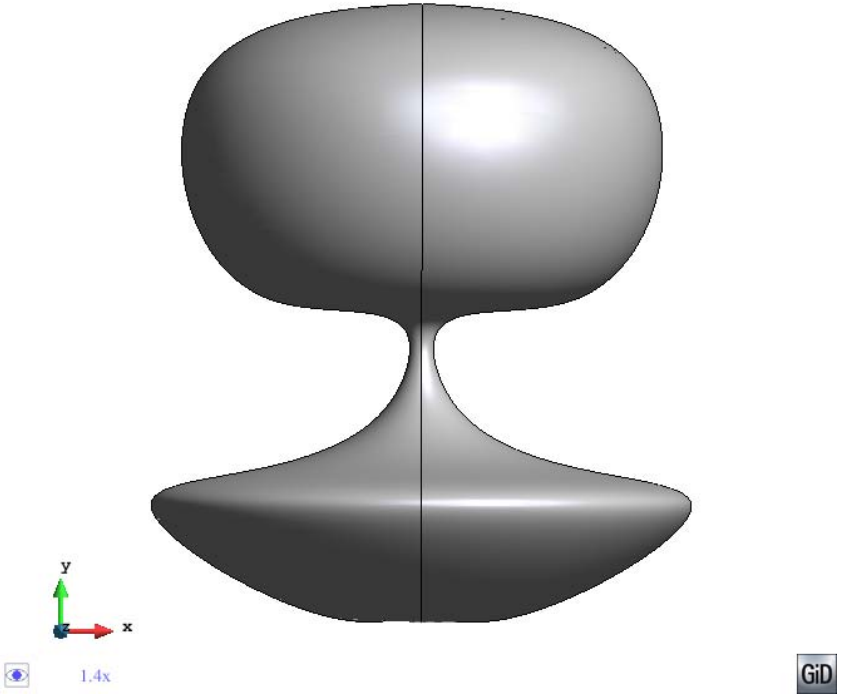


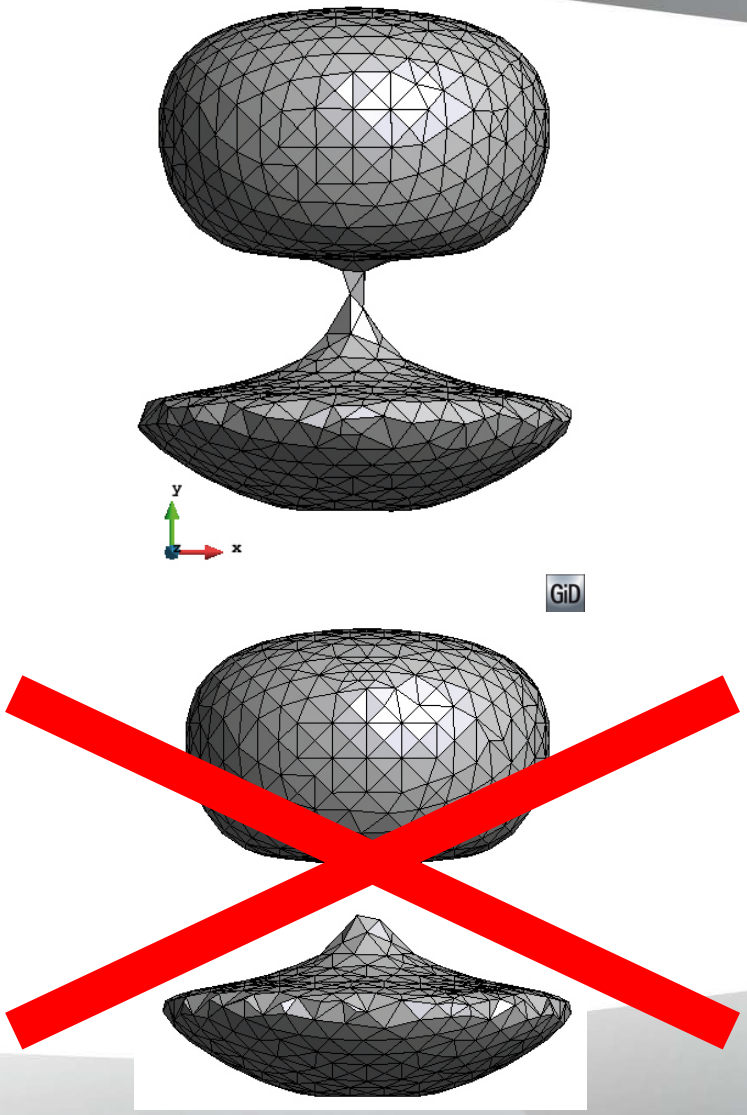




Mesher characteristics

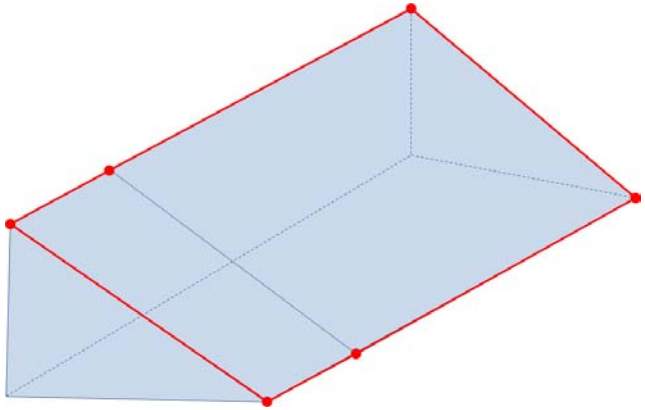
- Topology preservation

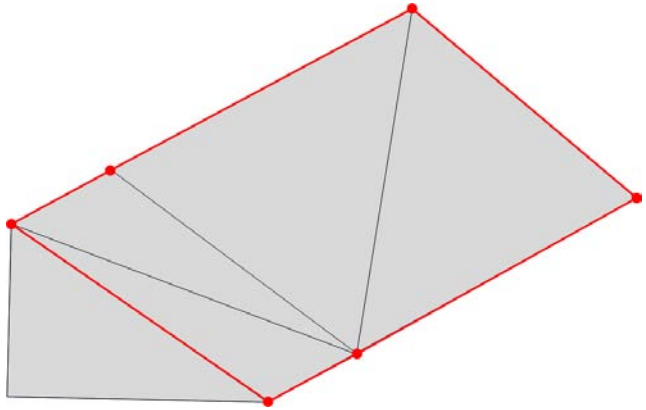




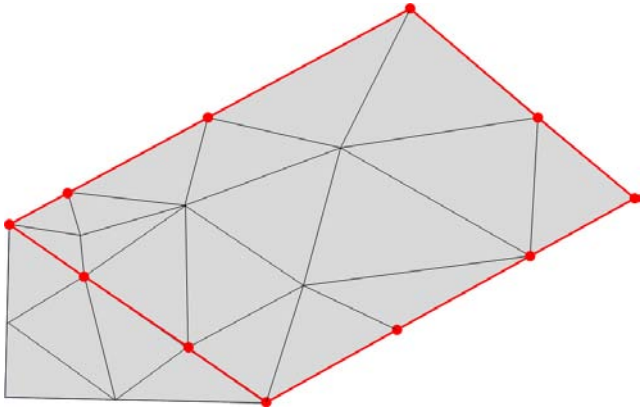
Mesher characteristics

- Maintain a representative topology of lines and surfaces





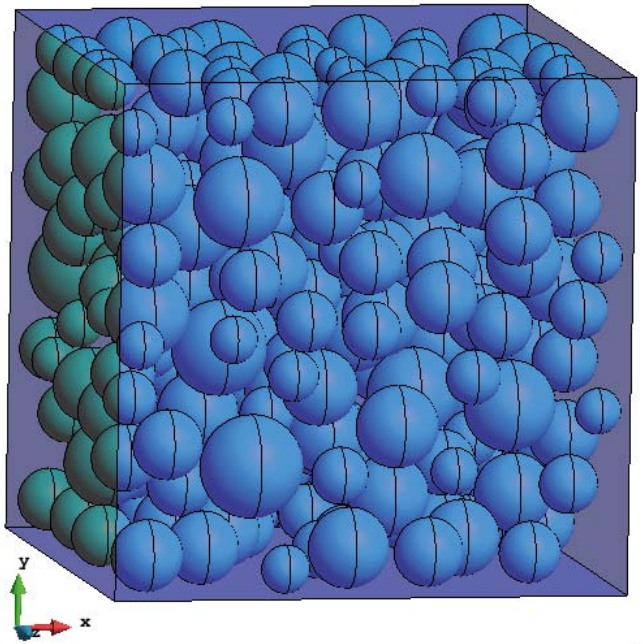
Constrained

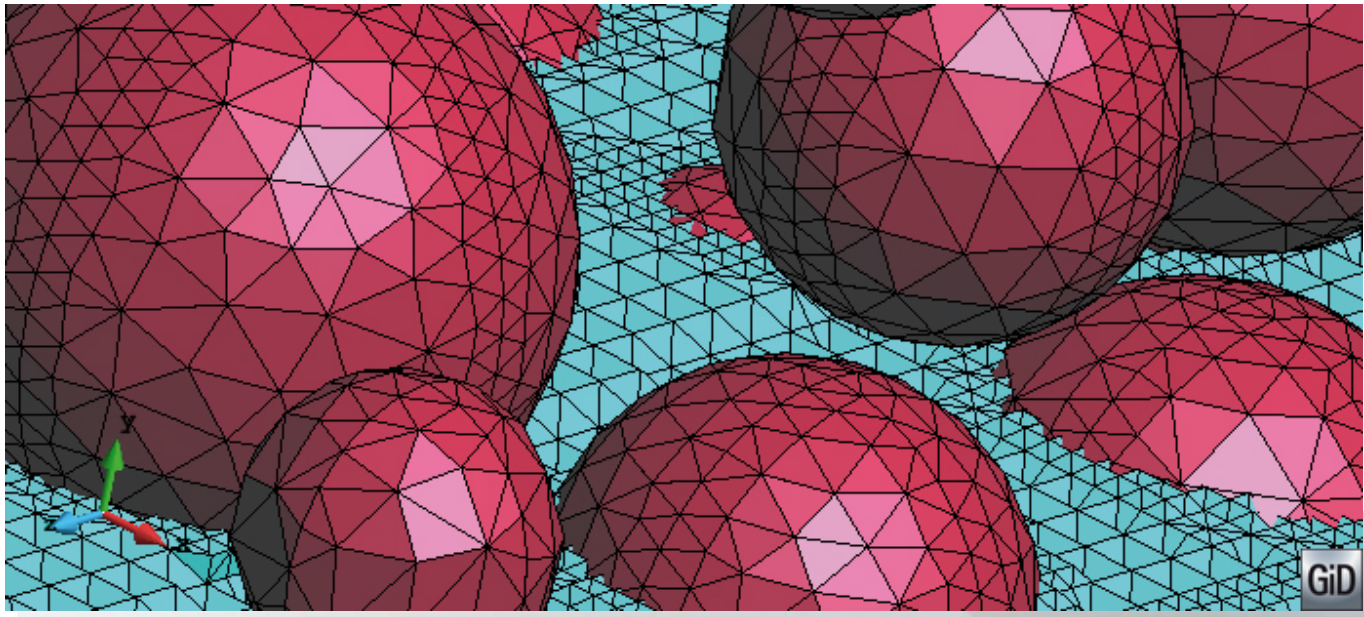


Partially constrained

Mesher characteristics

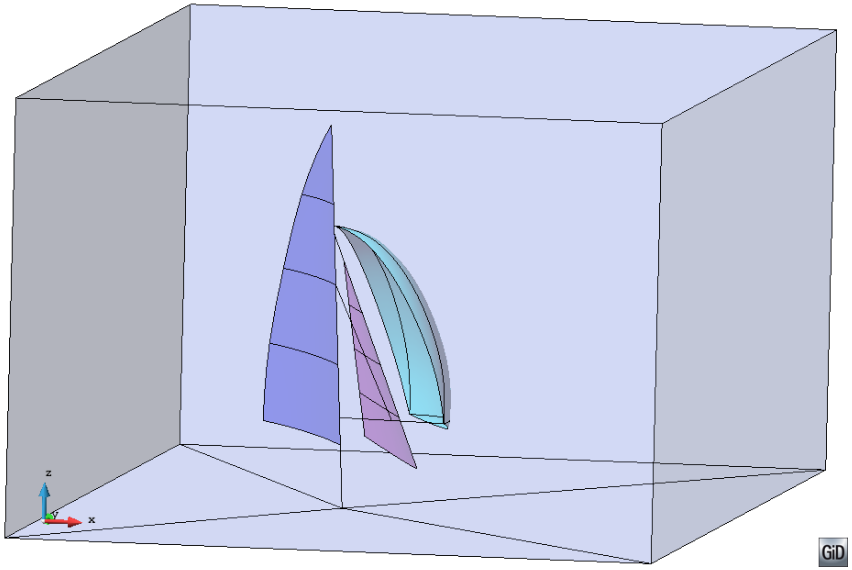
- Able to get the definition of geometry in common formats (Geometry and mesh format)
- Allow spatial mesh size distribution
- Applicable to embedded methods

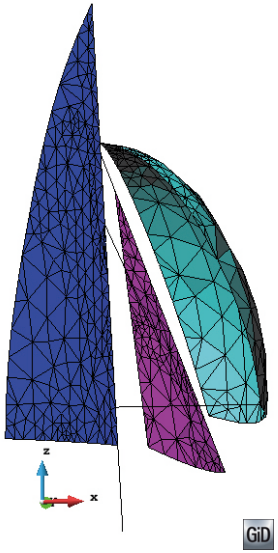


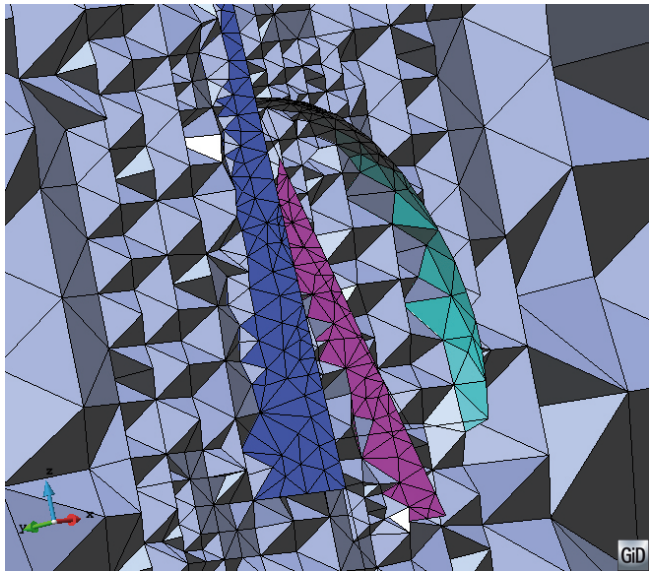


Mesher characteristics

- Able to mesh surfaces (in further developer versionss
  - Inner
  - isolated





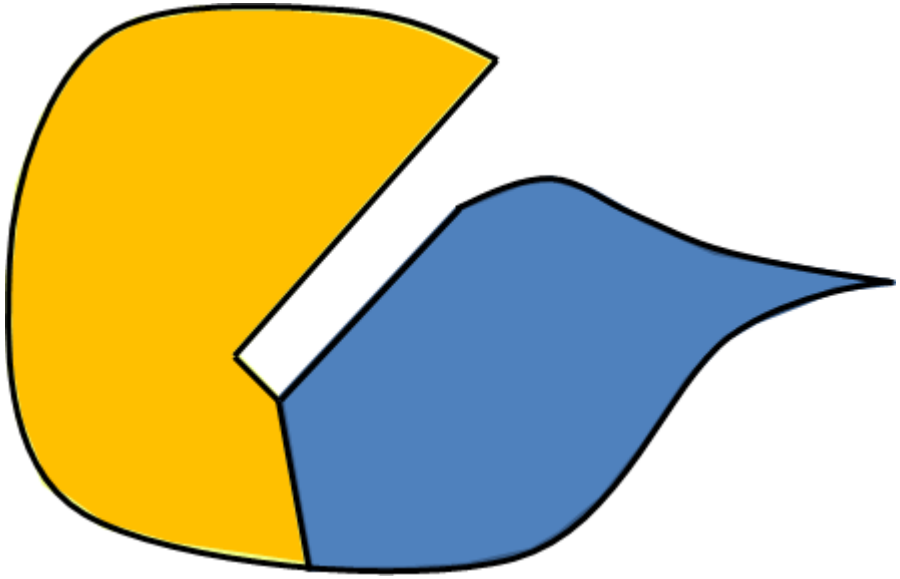


Outline

- Introduction
- Mesher characteristics
- Meshing algorithm step by step
- Examples
- How to use it inside GiD

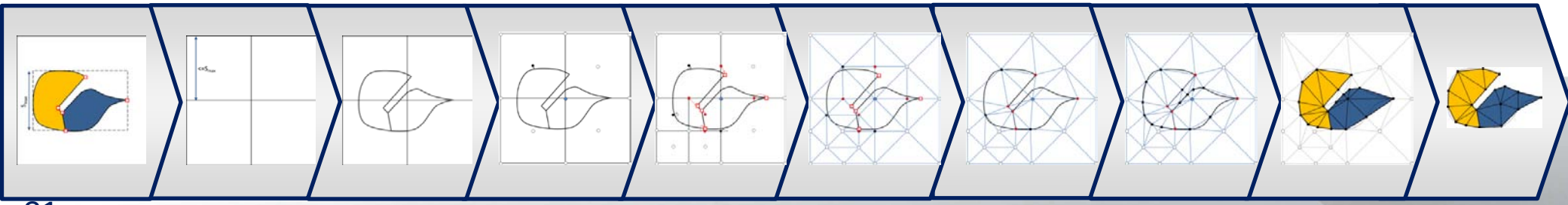
Meshing algorithm step by step

An example is used to illustrate the algorithm:

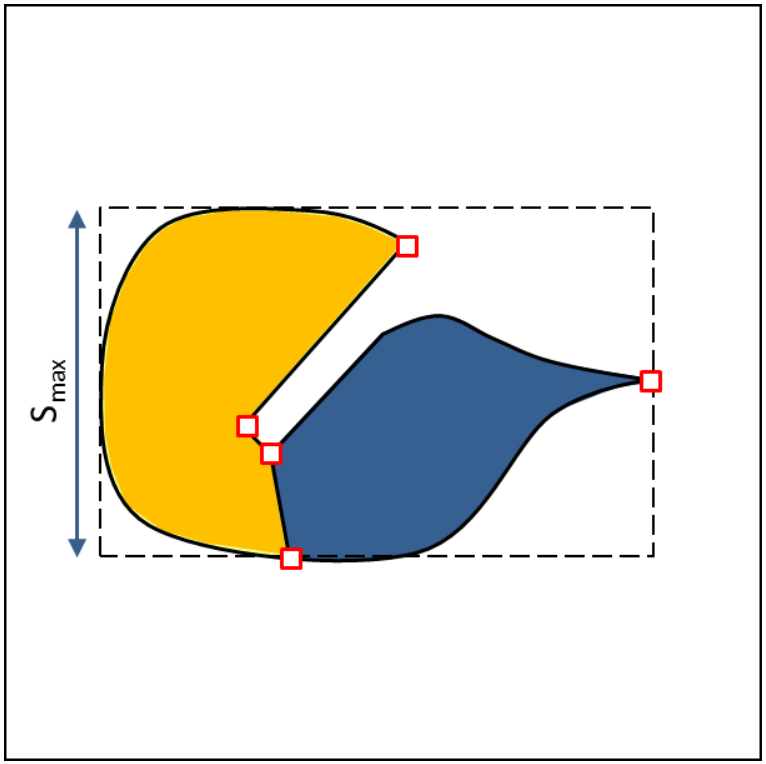


No sizes assigned

The algorithm consists basically in 10 steps:

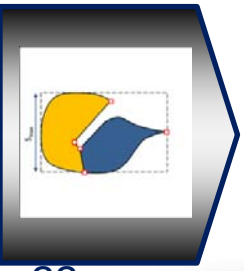


1- Process input data and create octree root

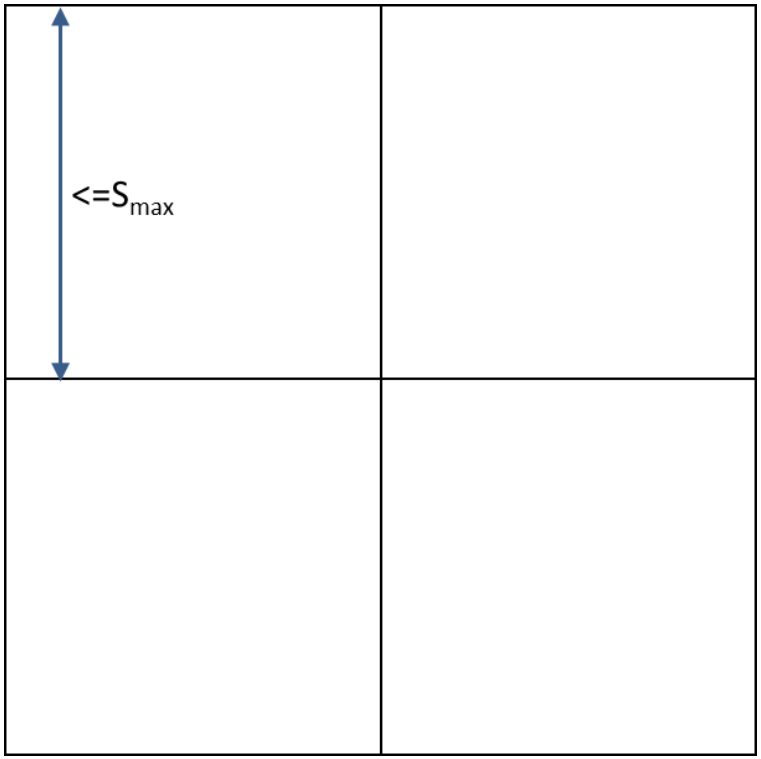


Input data:

- Contours of the volumes
- Mesh size entities
- **Forced points**
- **Forced lines**
- General parameters

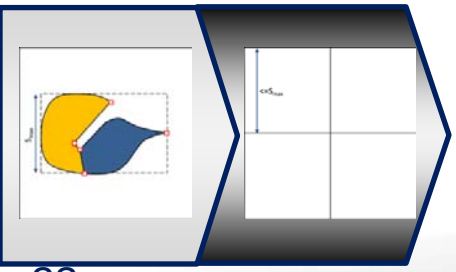


2 - Octree refinement with size refinement criteria

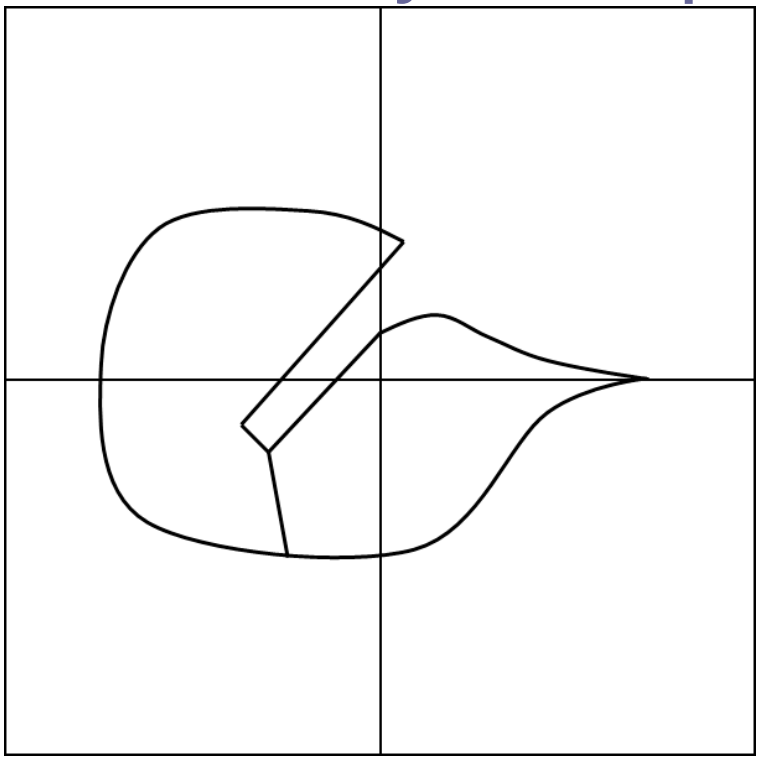


Same criteria as embedded mesher:

- Mesh sizes
- Sizes transition function
- Contrained 2 to 1

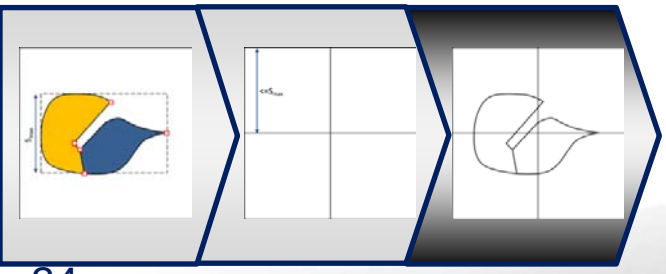


3 – Classify the input boundary entities in the octree

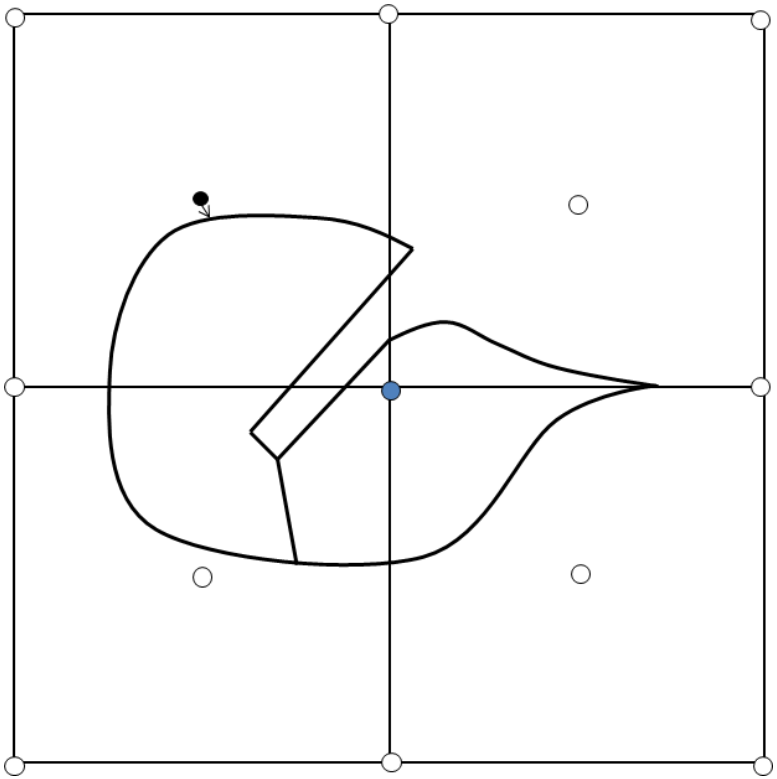


Detection which cell collides with the input boundaries. Then the cells are classified in:

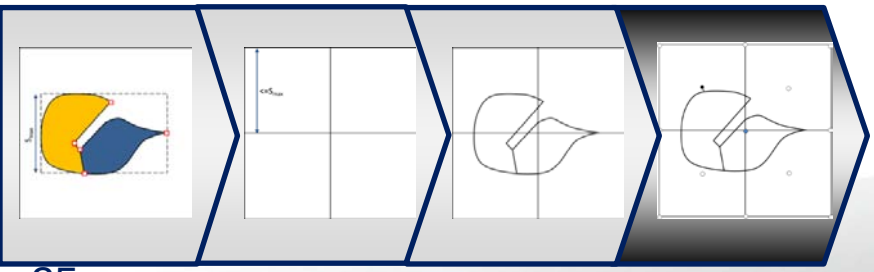
- Inner cells
- Outer cells
- Interface cells



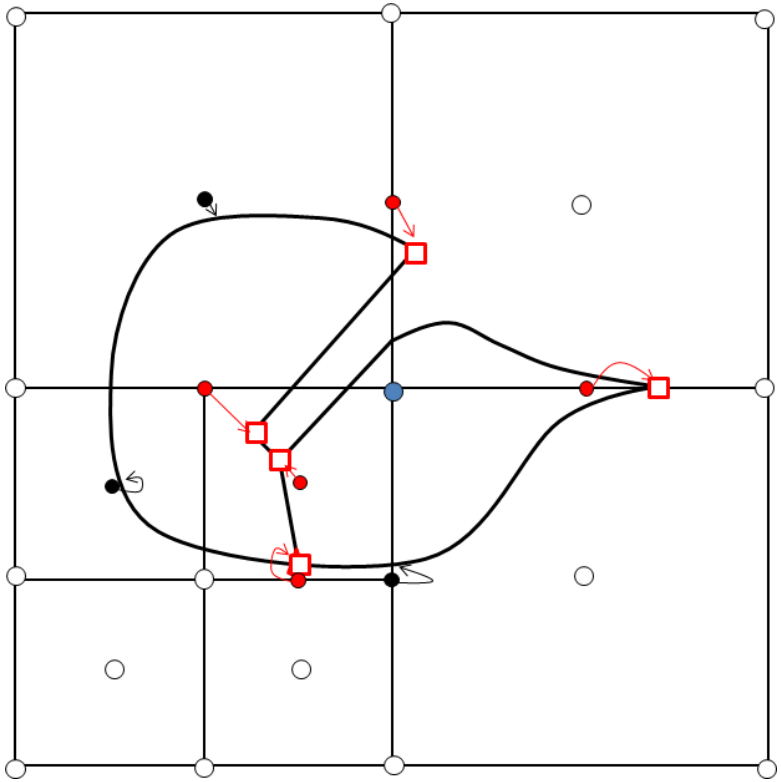
4 – Color the octree nodes and set Forced Interface Nodes



- Create nodes
- Color nodes (determine where they are topologically, solving the Point in Polygon Problem PiP)
- Set Forced interface nodes (close to the boundaries)



5 – Refine the octree with the body-fitted criteria

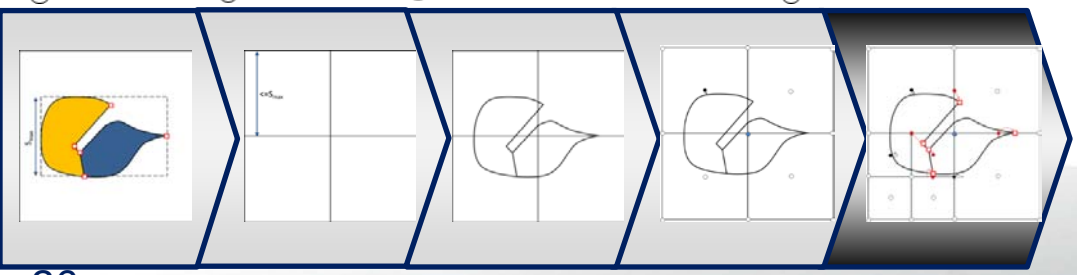


Refinement criteria:

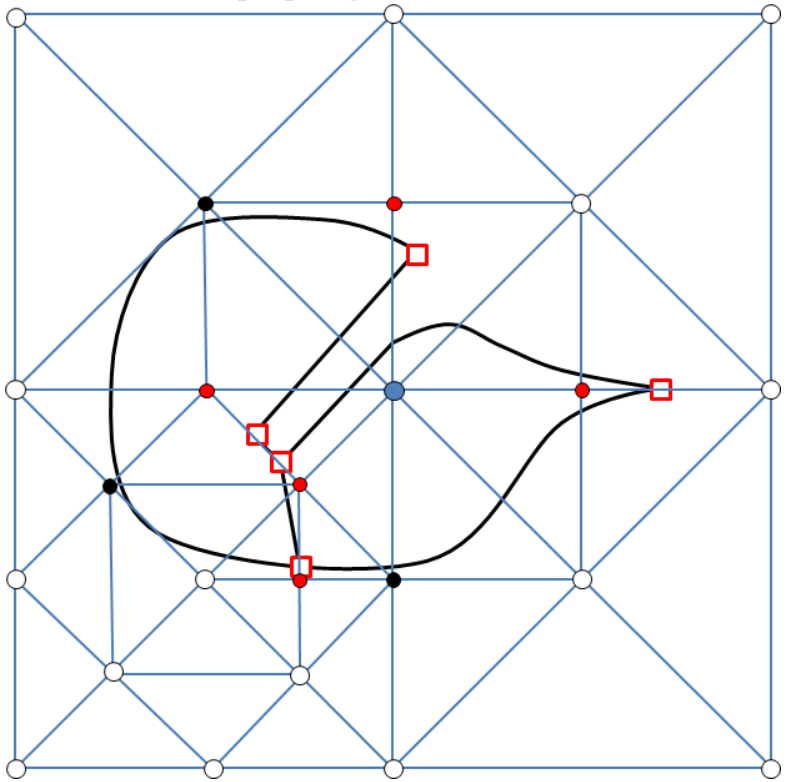
- Forced nodes
- Tetrahedra distortion
- Topological criteria

Each time a cell is subdivided:

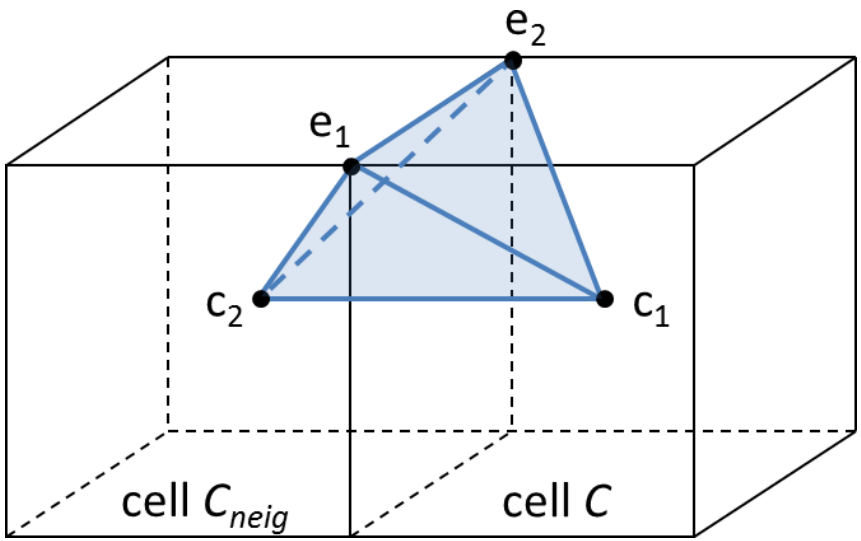
- New nodes are created
- Forced nodes are updated (created and deleted)

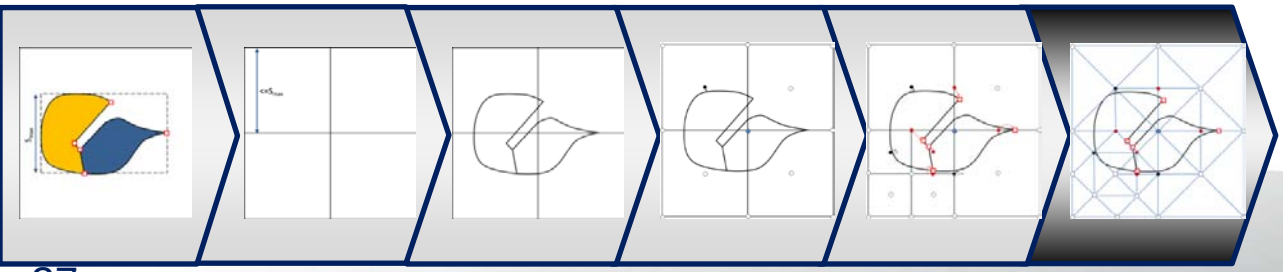


6 – Apply the tetrahedra pattern

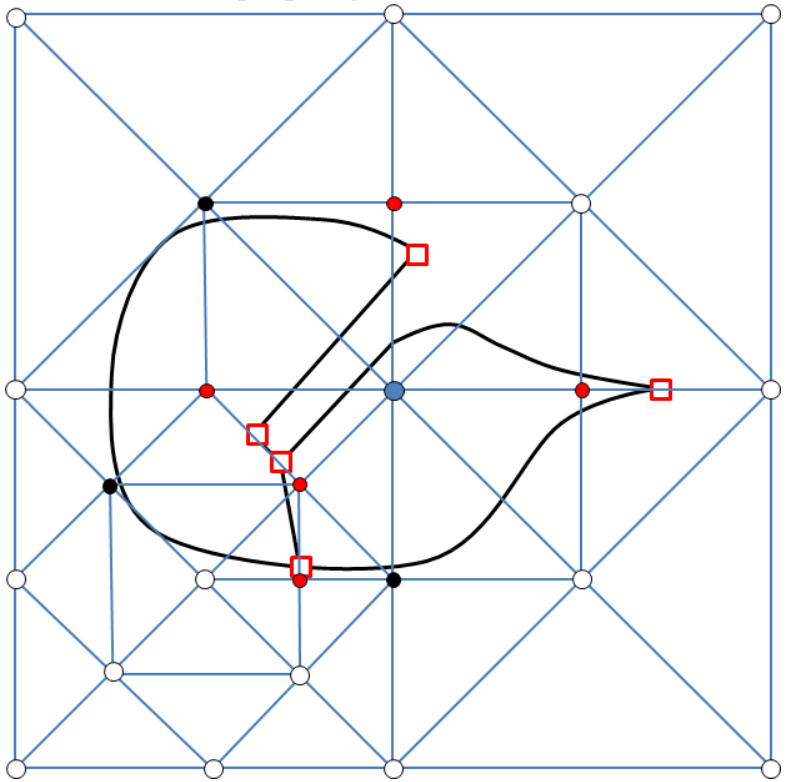


Tetrahedra pattern based on BCC

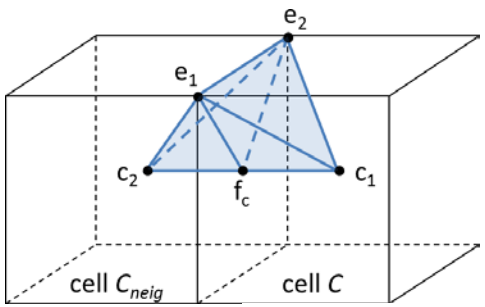


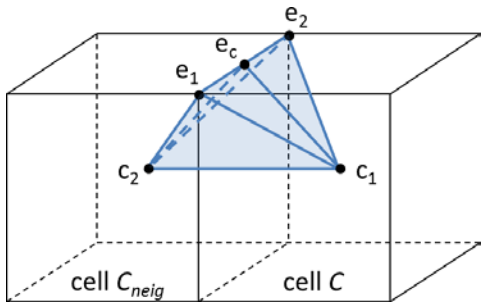


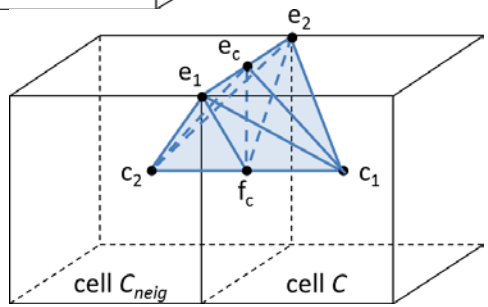
6 – Apply the tetrahedra pattern

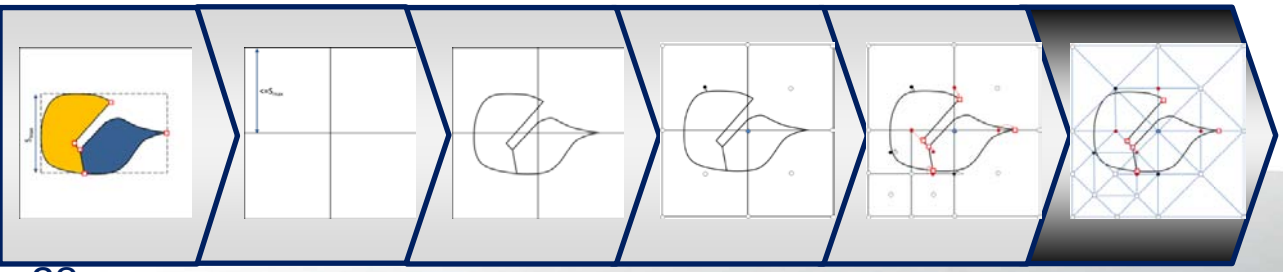


New patterns considering quadratic positions

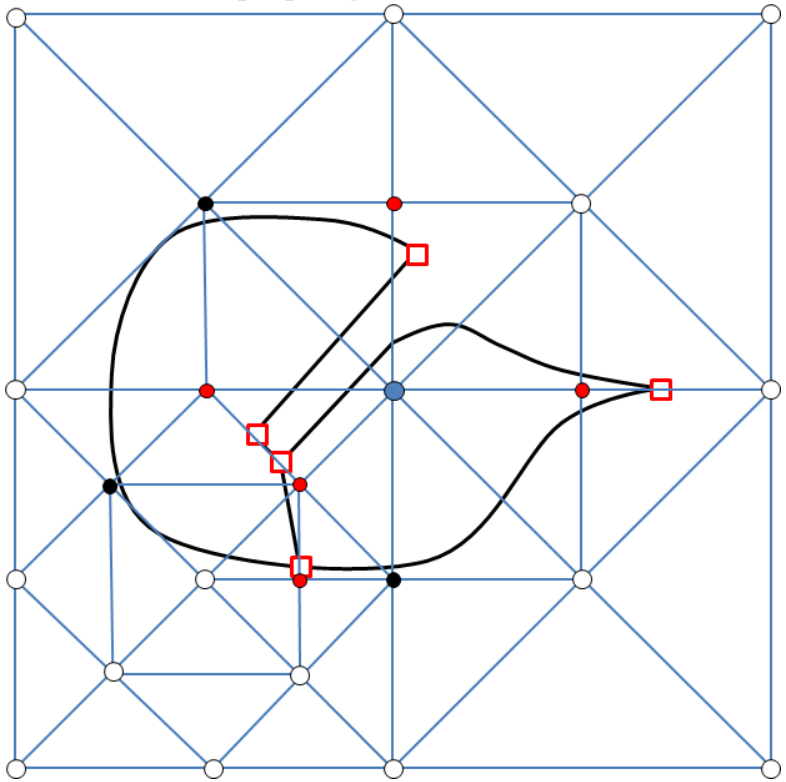


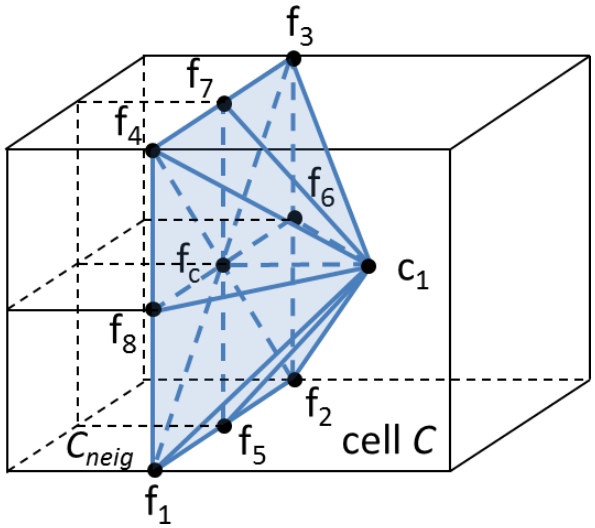


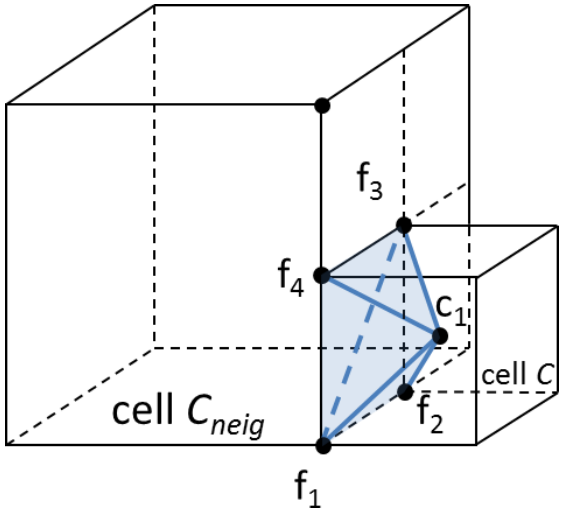


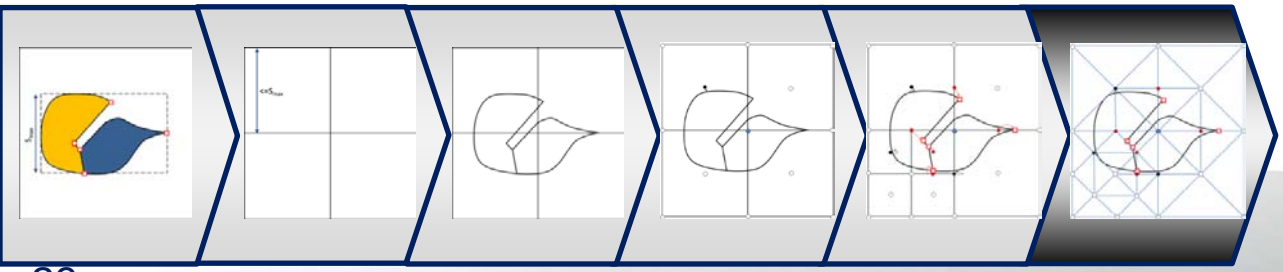


6 – Apply the tetrahedra pattern

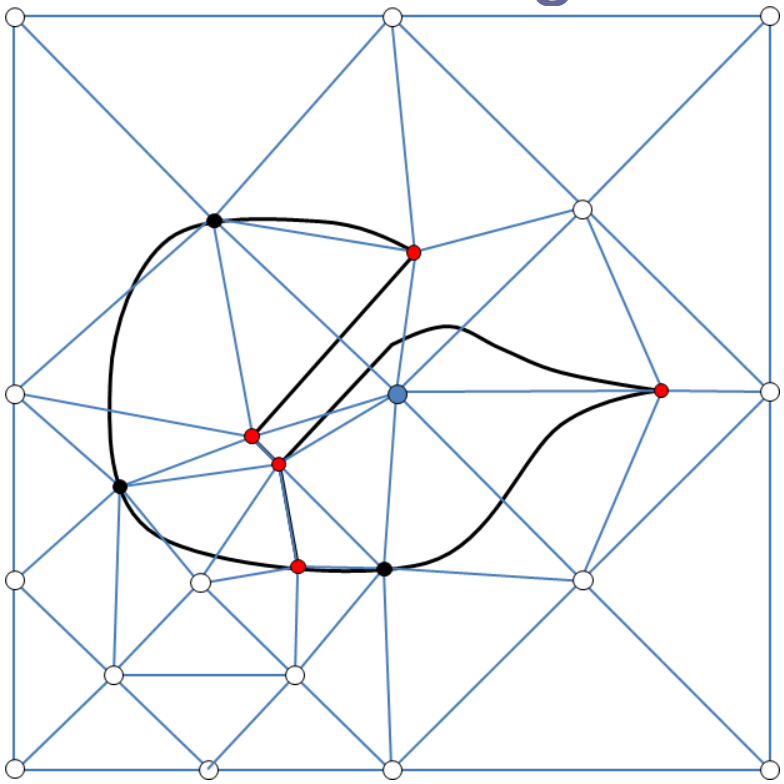




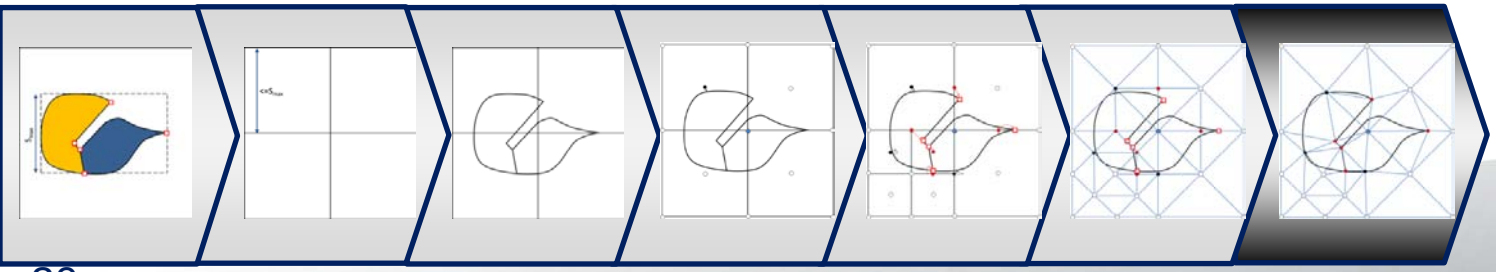




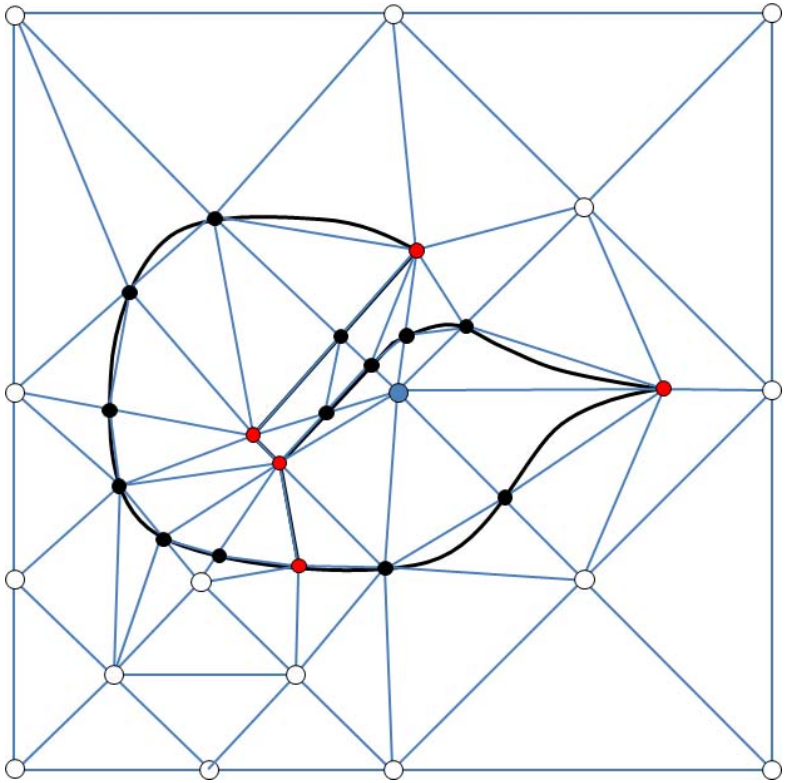
7 – Preserve geometric features



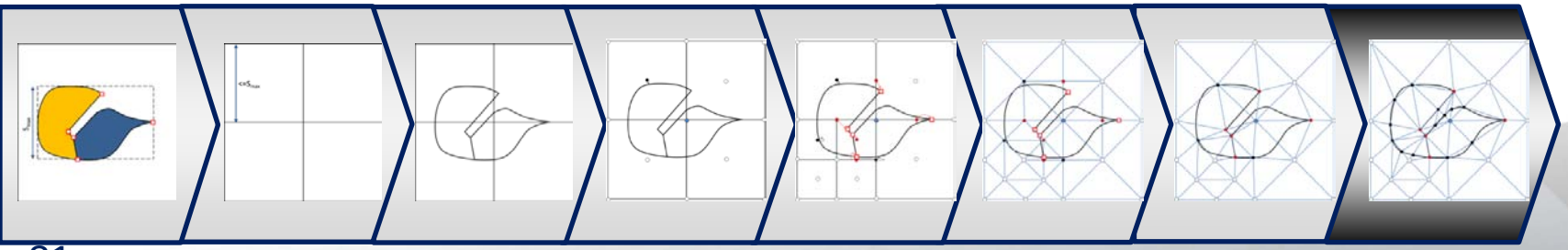
- Forced nodes are snapped into their final position
- Forced edges are splitted



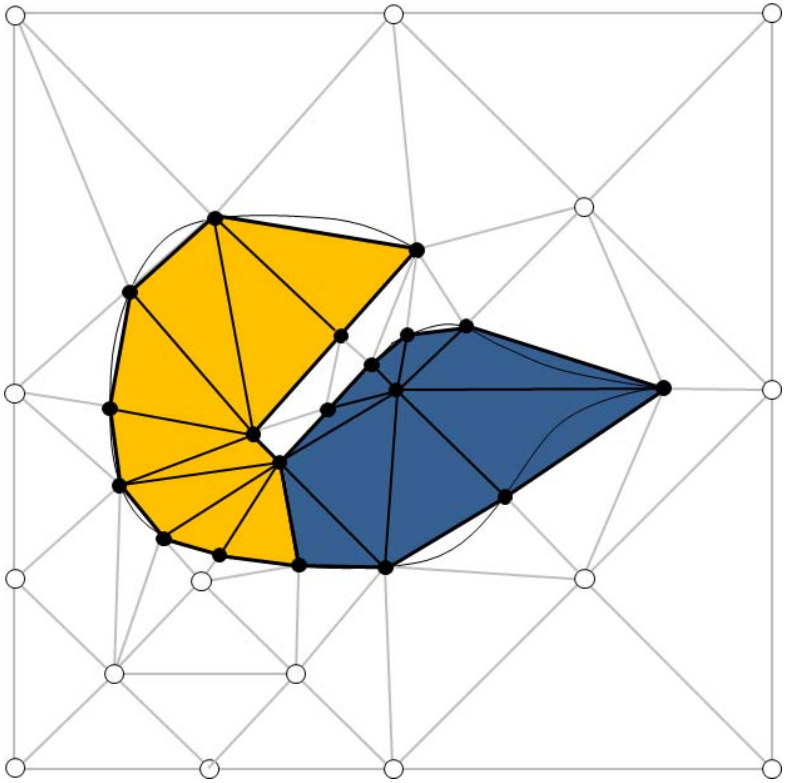
8 – Surface fitting process



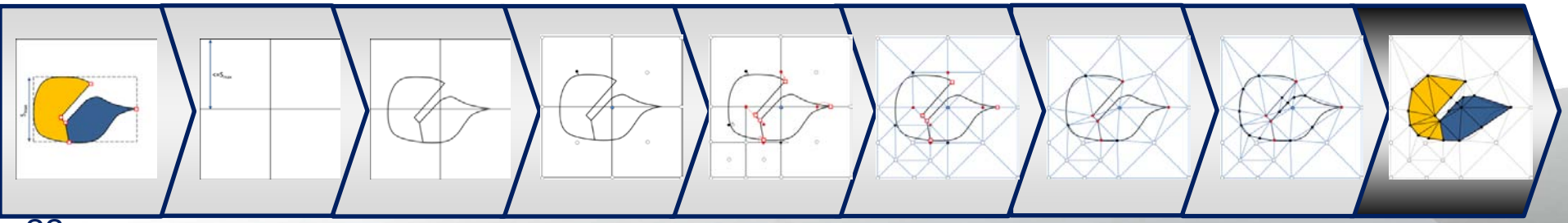
- Nodes near boundary are moved
- Edges intersecting boundary are splitted



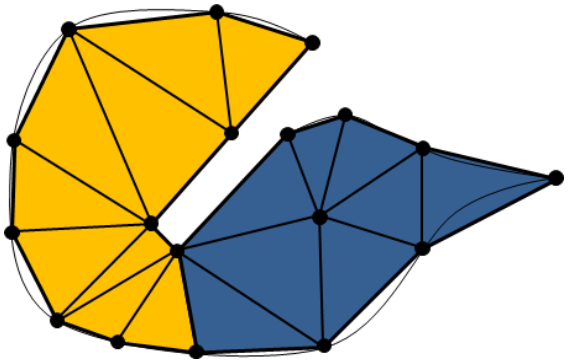
9 – Tetrahedra coloring and identification of skin mesh



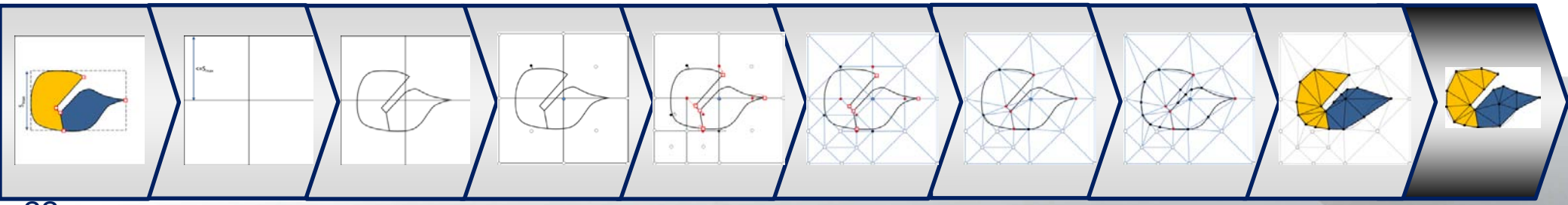
- Determine the volume each tetrahedron belongs to
- Extraction of the skin of the volumes meshes (triangles)



10 – Make-up and smoothing



- Edge collapsing
- Edge swapping
- Laplacian-like smoothing

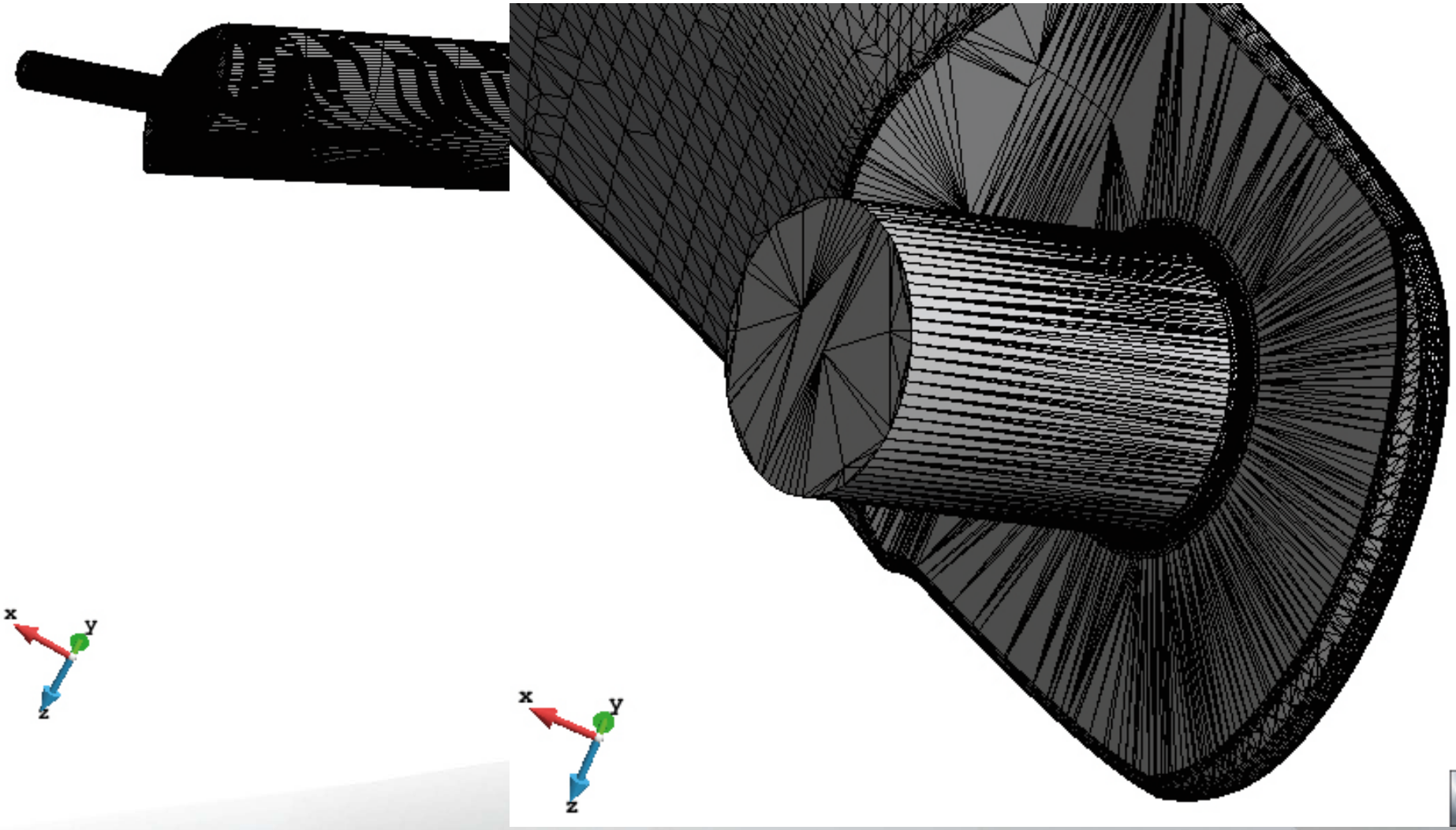


Outline

- Introduction
- Mesher characteristics
- Meshing algorithm step by step
- Examples
- How to use it inside GiD

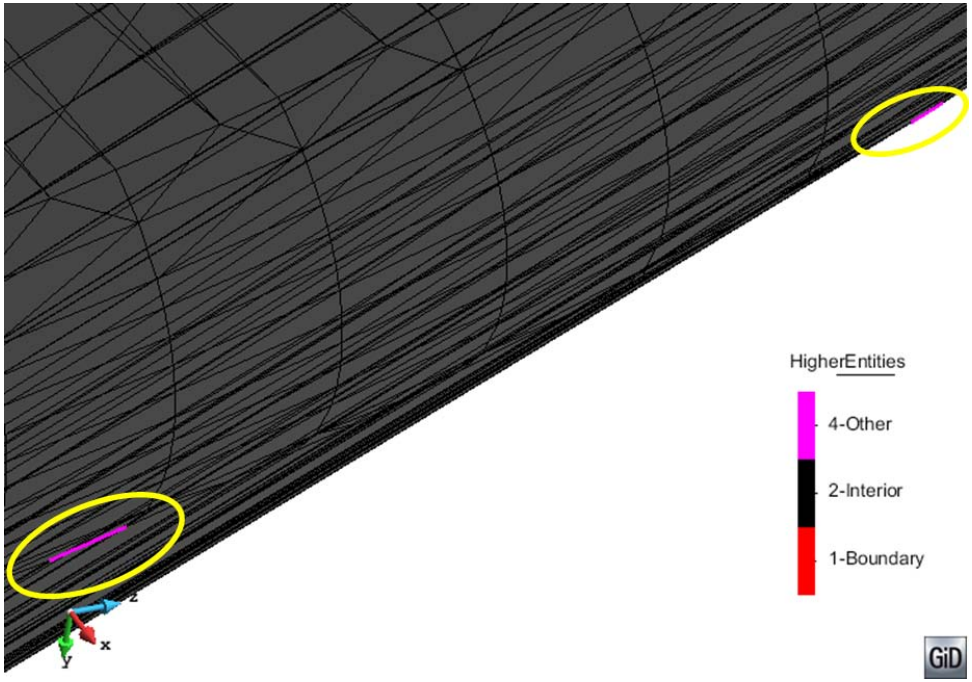
Validation examples

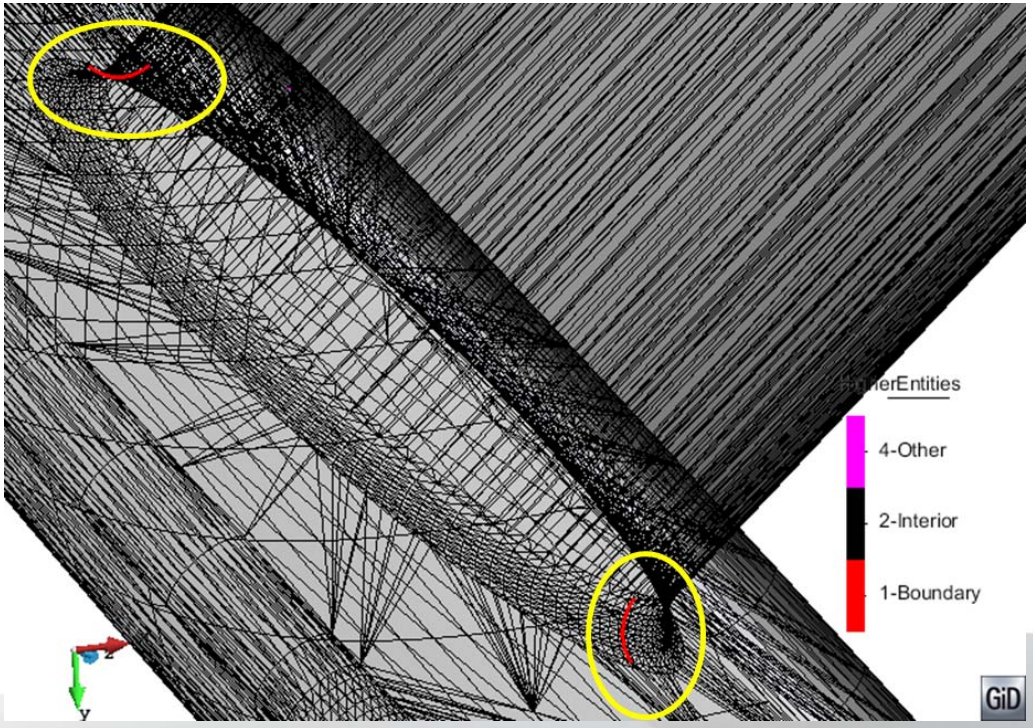
Mechanical piece in stl format



Validation examples

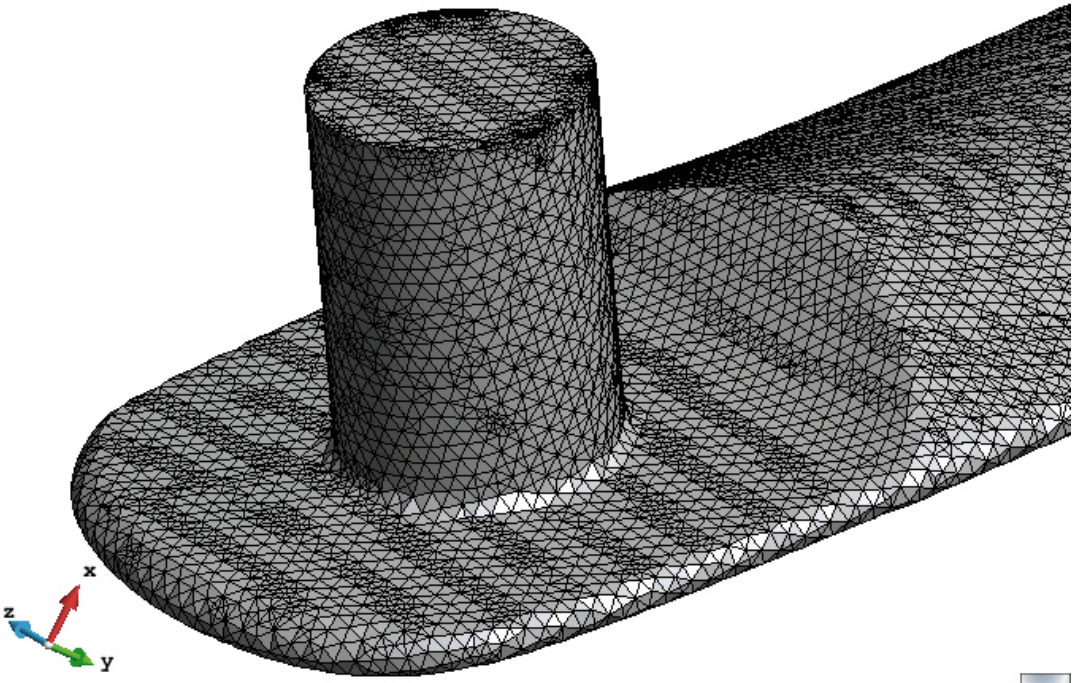
Non-watertight model

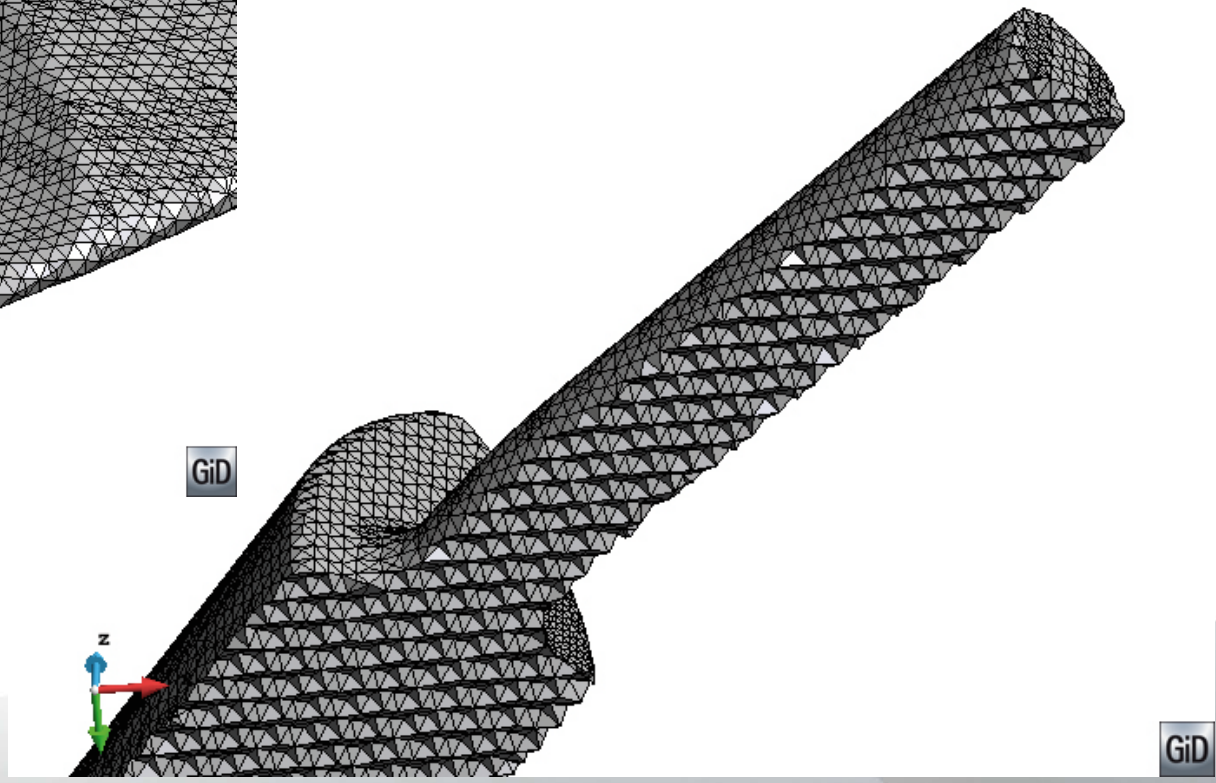




Validation examples

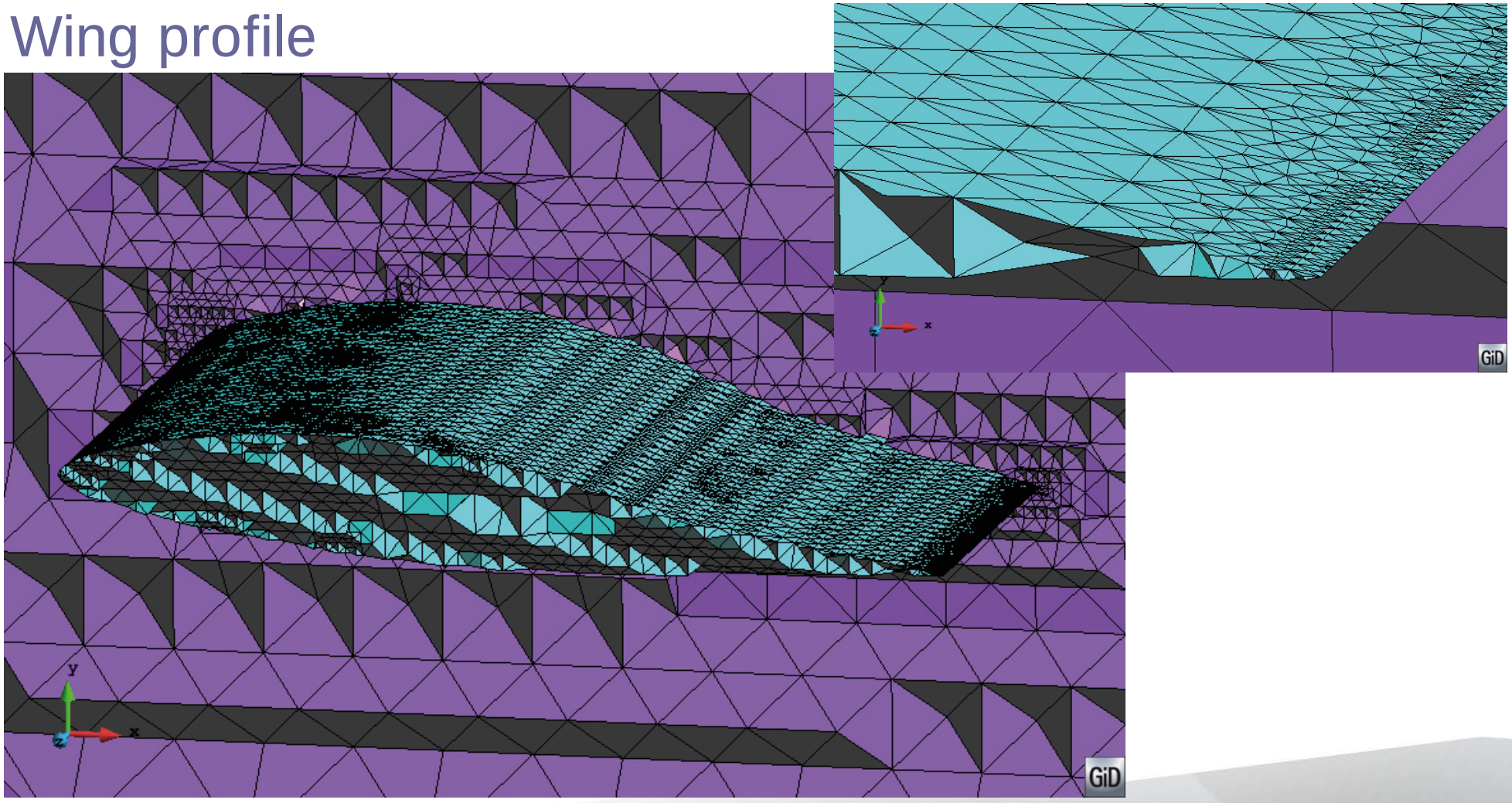
Final mesh





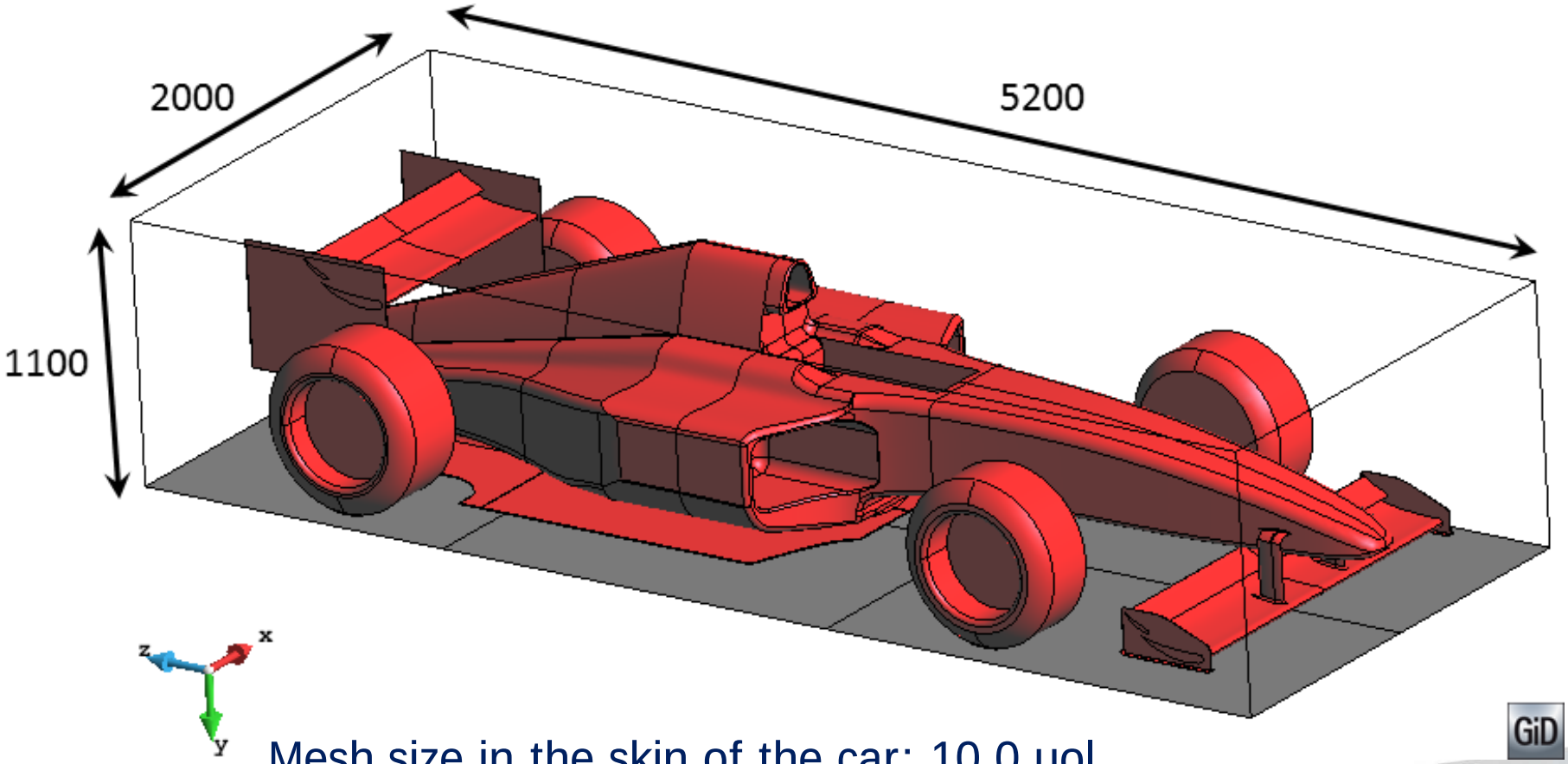
Validation examples

Wing profile



Racing car

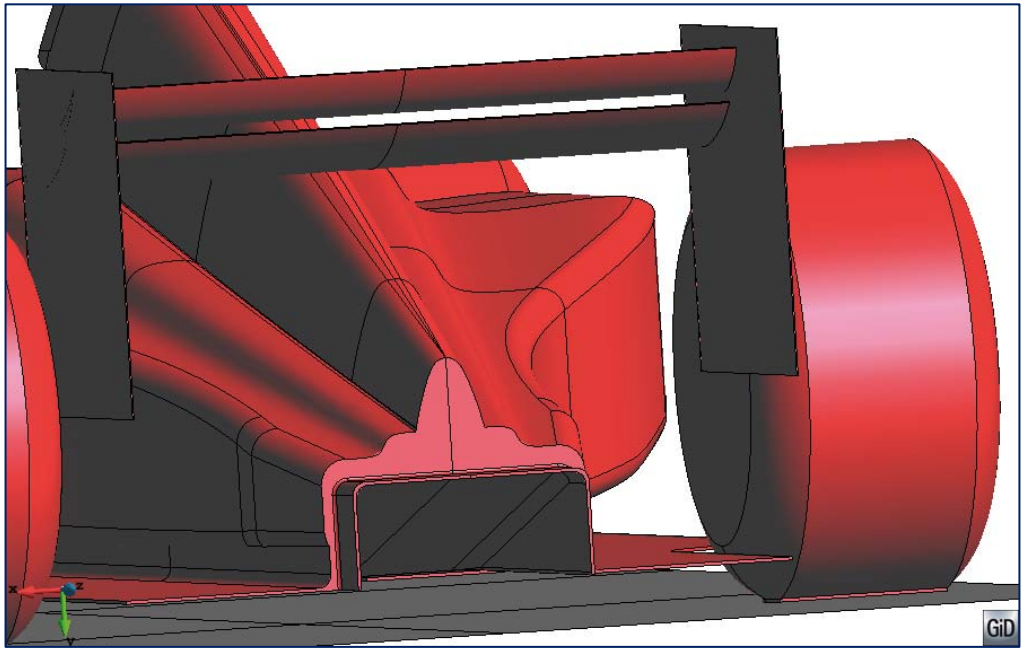
Body-fitted mesher: model of a racing car

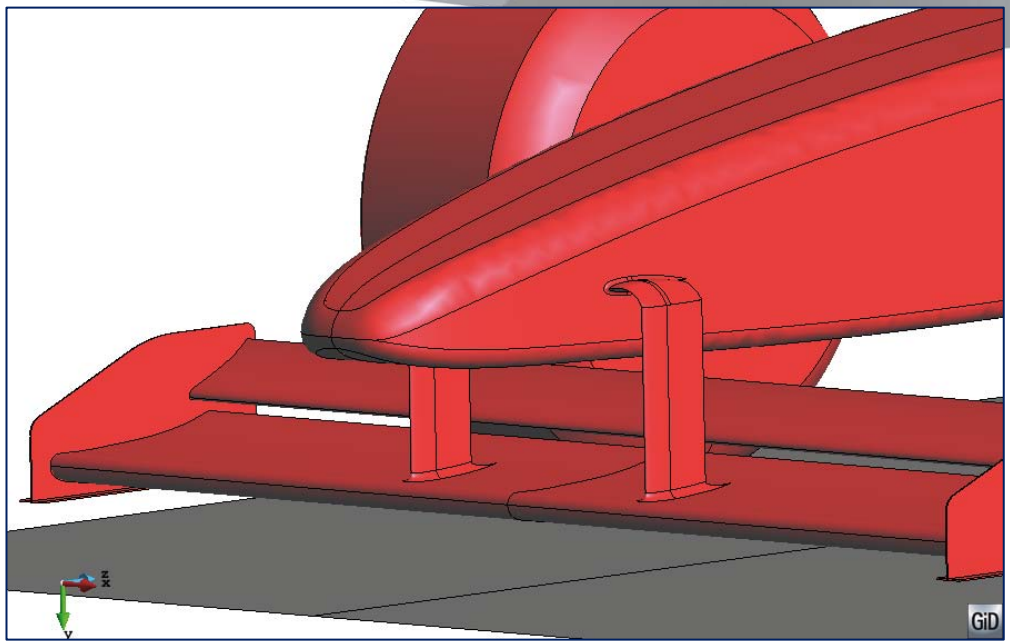


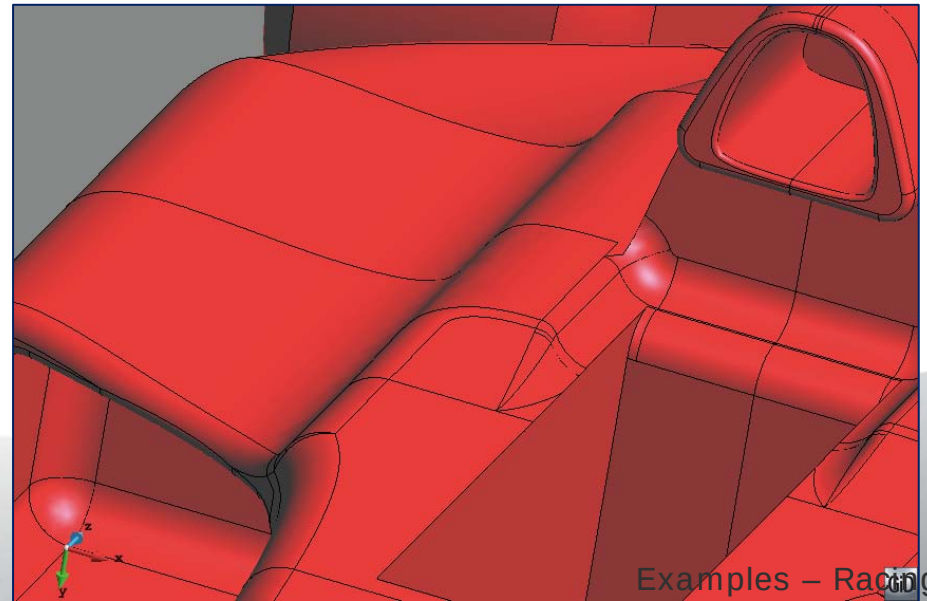
Mesh size in the skin of the car: 10.0 uol
Mesh size in outlet surface: 20.0 uol
Transition factor: 0.6

Racing car

Detail views of the model

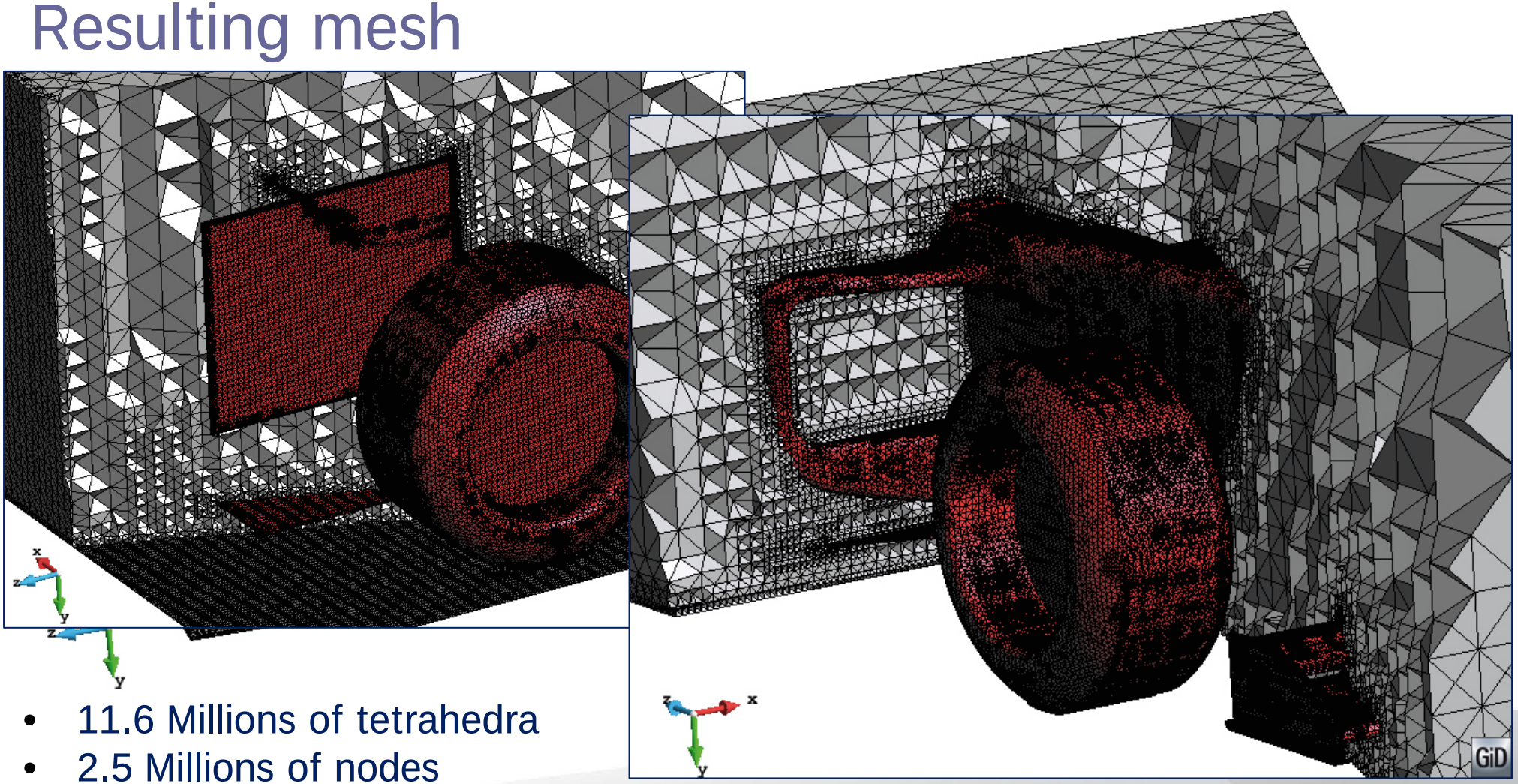






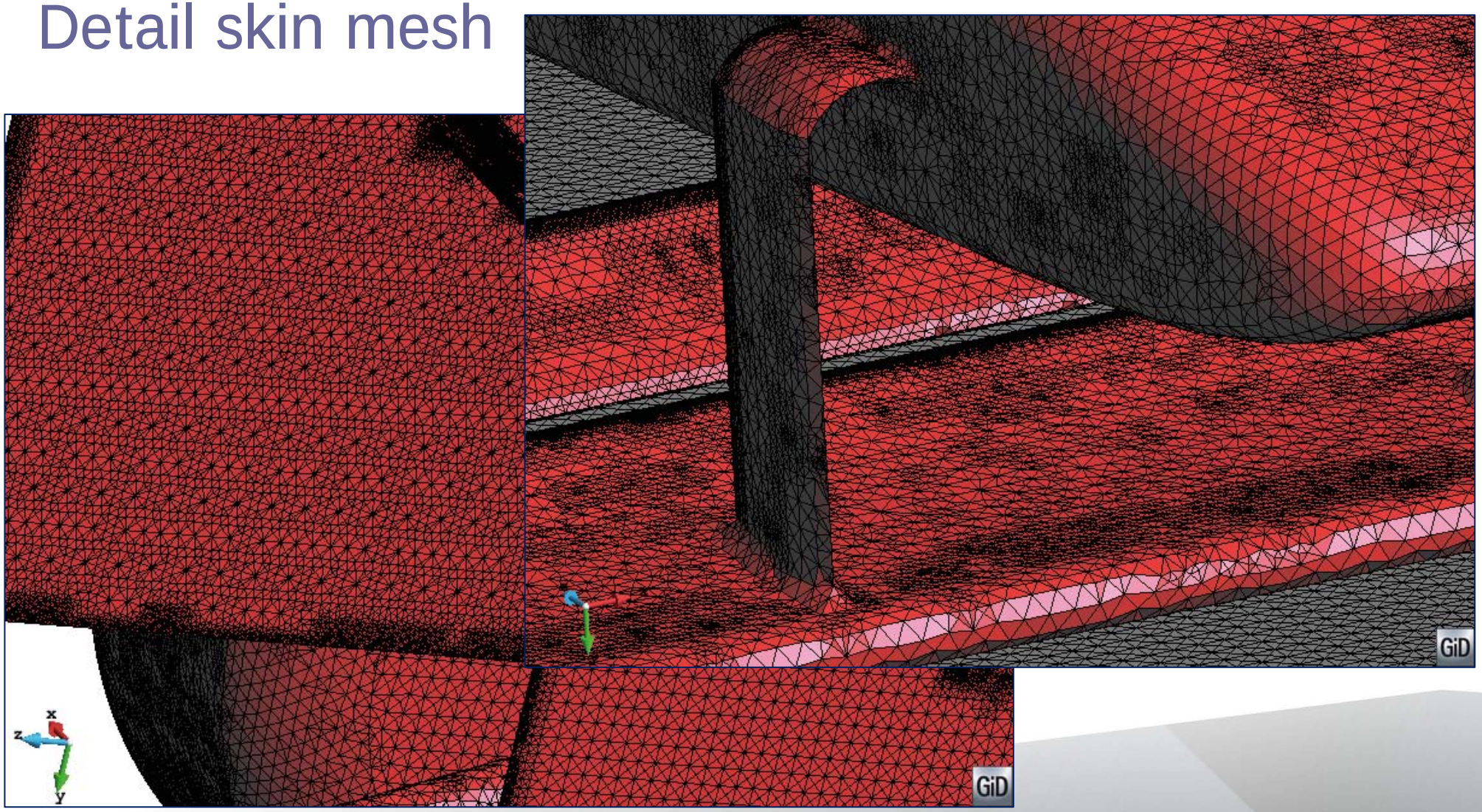
- Very thin parts (5 uol)
- Smooth surfaces

Resulting mesh



- 11.6 Millions of tetrahedra
- 2.5 Millions of nodes
- 4 elements with minimum dihedral angle <10º

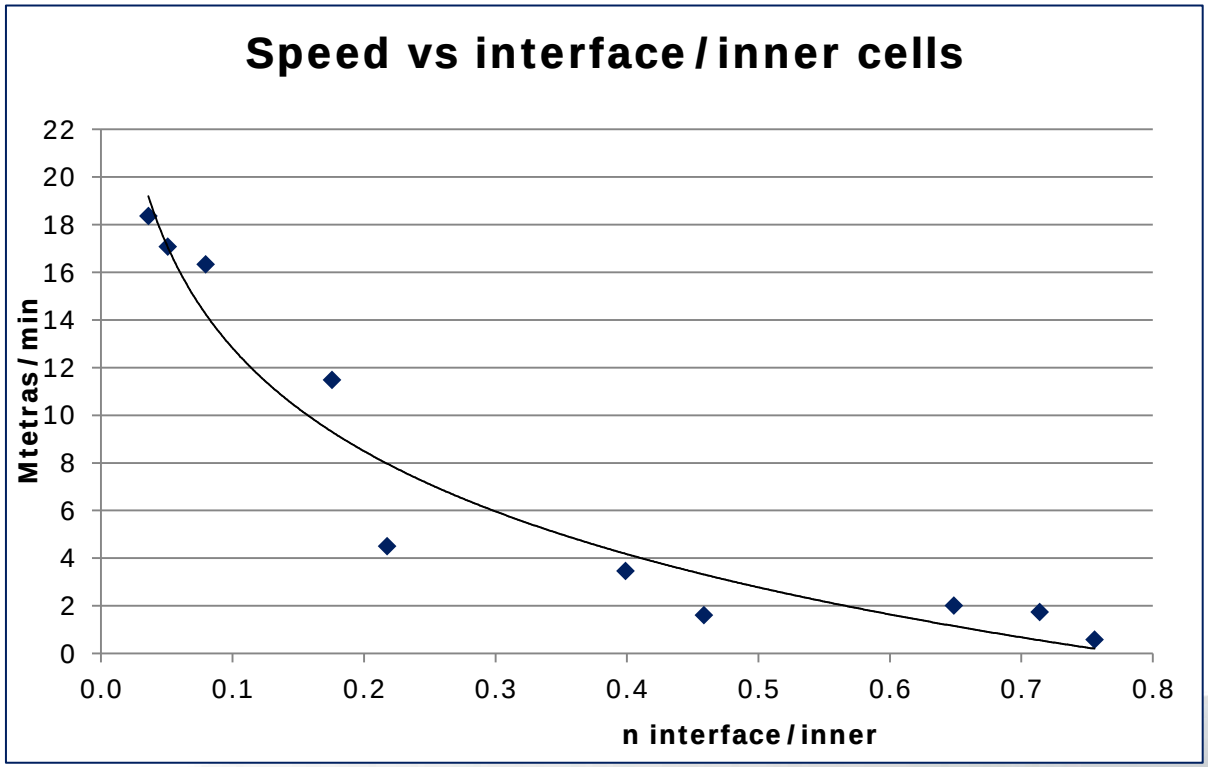
Detail skin mesh



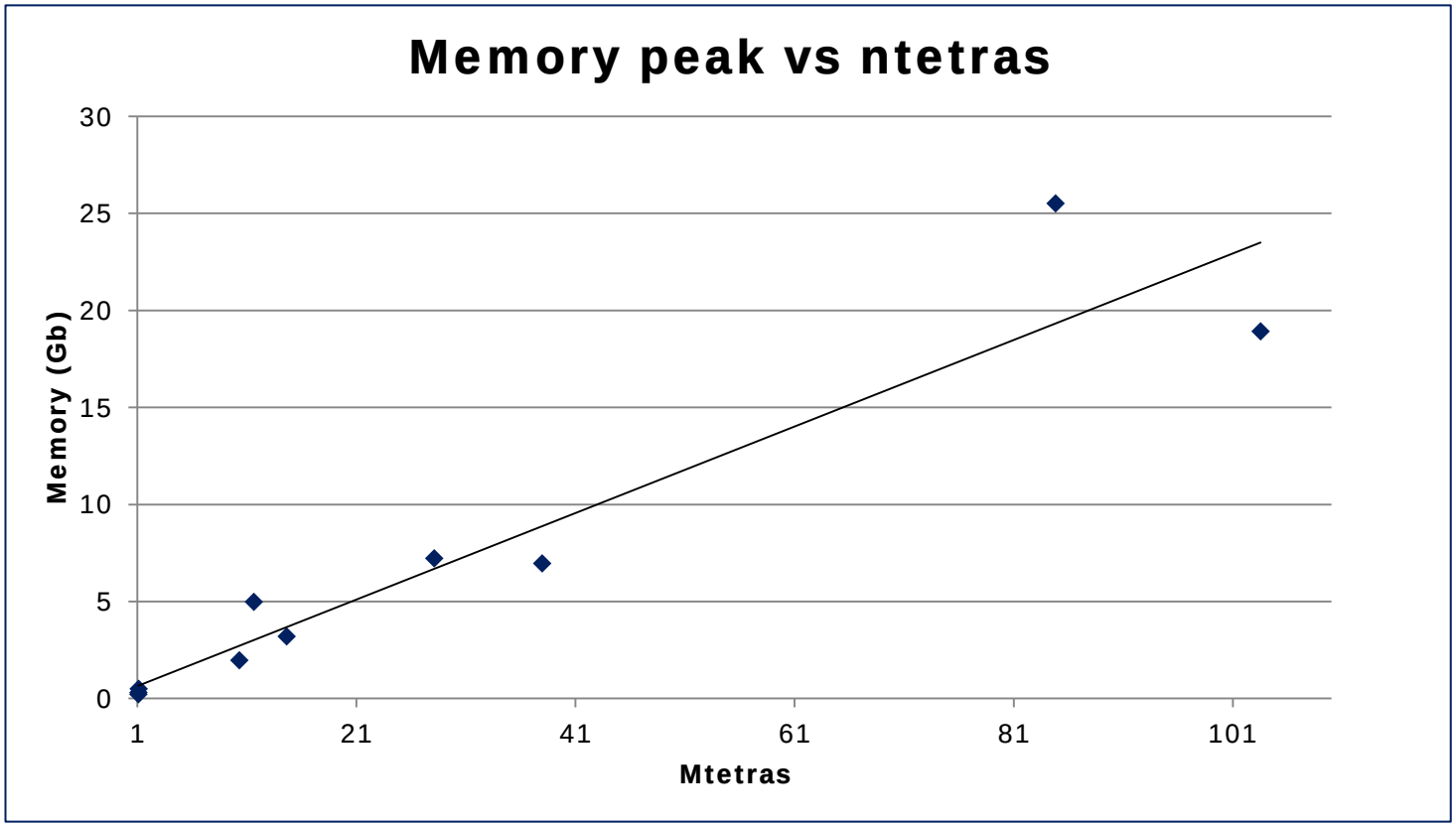
Performance:

Speeds from 0.6 to 18 Mtetrahedra per minute

Speed-up of 1.6 using 4 threads



Memory consumed

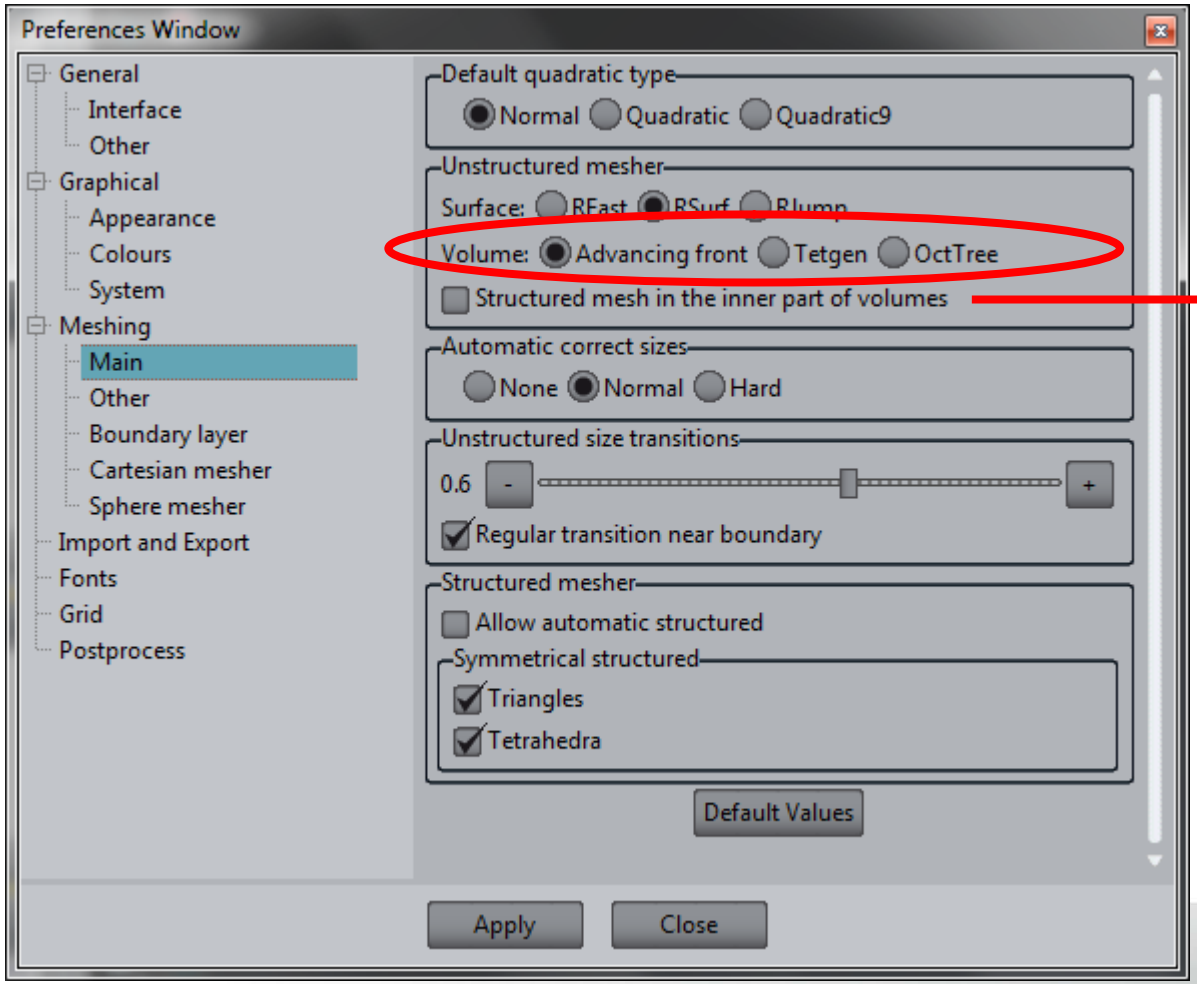


0.25 Gb of memory peak per Mtetrahedra

Outline

- Introduction
- Mesher characteristics
- Meshing algorithm step by step
- Examples
- How to use it inside GiD

Available as another unstructured mesher



Option to use the octree pattern only in the inner part of the volumes

How to use it inside GiD

- The mesher is not used if some entity has structured information
- Lines and points are skipped following Rjump criteria:
  - Their parents are tangent enough
  - They have the same properties as the parents
  - They are in the same group as the parents
- Non-watertight geometries are able to be meshed using MeshFromBoundary option (right buttons)
  - Tolerance of gaps and sharp edges can be entered

Further developments

- Make the GiDMeshLibrary public.
  - Accessible as a product external to GiD

- Further developments to use it with GiD:
  - Allow to define non-watertight geometries in GiD
  - Improvement in the render mesh
  - Improvement in the make-up and smoothing operations
  - Improvement in sharp edges detection for stl models

Thanks for your attention