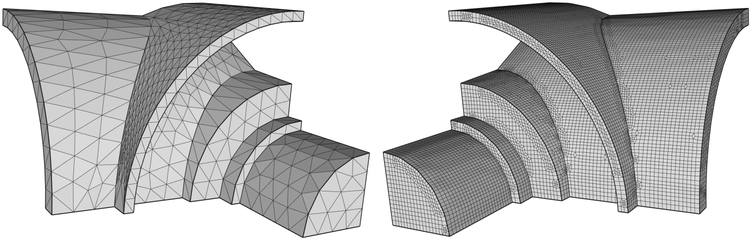
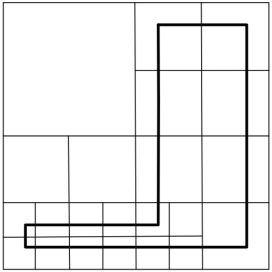
Hexotic
an automated hexahedral mesh generator

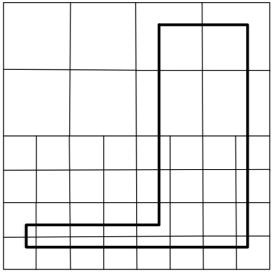


Loïc Maréchal Gamma3 / INRIA Saclay, France

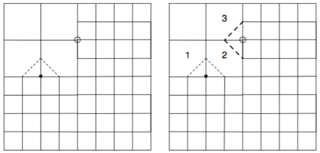
Basics of octree meshing

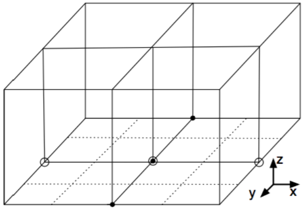
- set of subdivision criteria: geometry's thickness, curvature, a posteriori error estimate,
- balancing rule: a cell should not be more than two time bigger or smaller than it neighbors,
- Pairing rule: if a cell is to be subdivided, then it "brothers" should be subdivided too.

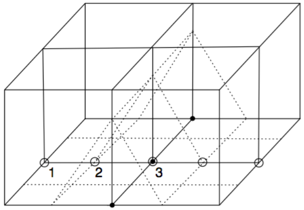


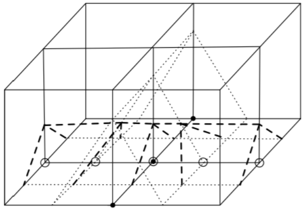


Octree conforming using polyhedral cutting

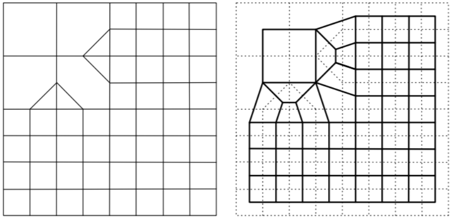




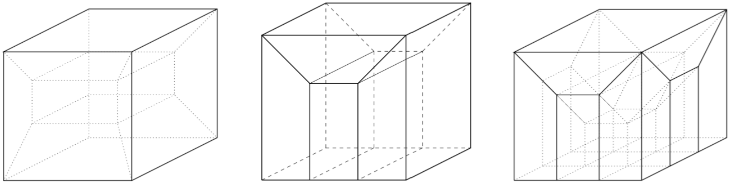




Dual mesh

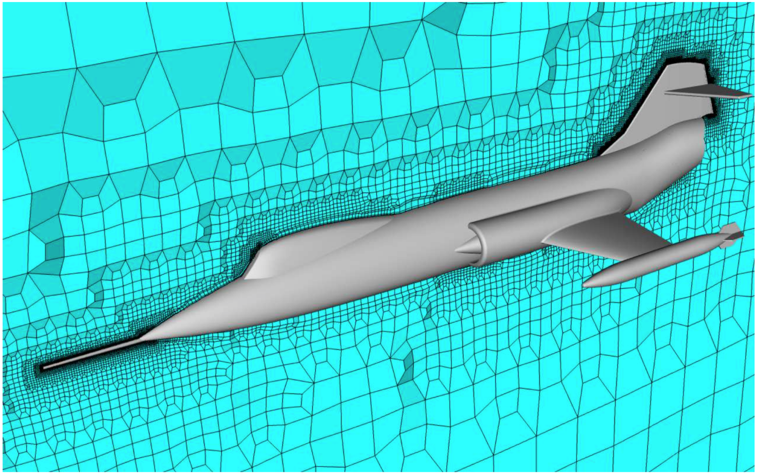


2D case, element $\Longleftrightarrow$ node, edge $\Longleftrightarrow$ edge



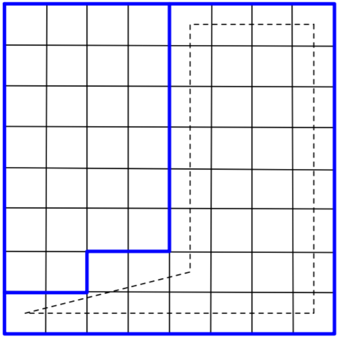
3D case, element $\Longleftrightarrow$ node, face $\Longleftrightarrow$ edge

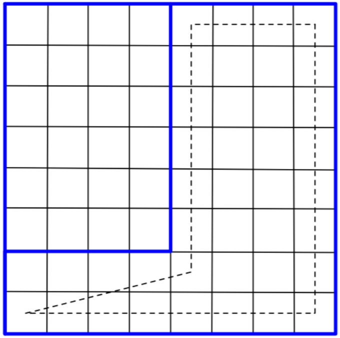
Dual mesh: airplane



Subdomains recovering

- basic inside/outside algorithm,
- It can handle internal surfaces (non-manifold geometries),
- domains must be two-element thick in any direction.





Surface meshing

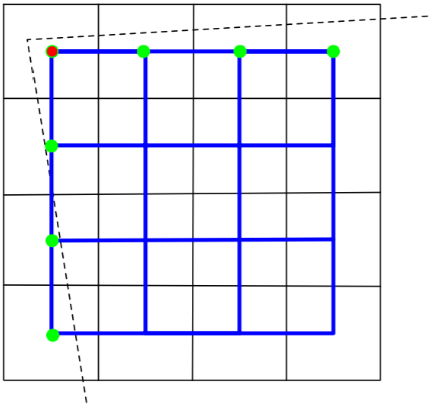
Analyzing groups of edges (triangles) intersecting each octant:

2D case:

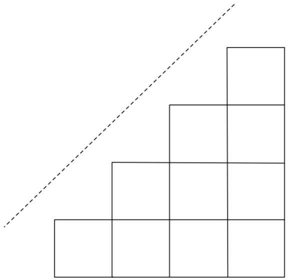
1. median line,
2. two intersecting lines making a sharp angle,
3. too complex.

3D case:

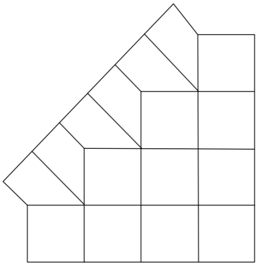
1. median plane,
2. two intersecting planes making a sharp edge,
3. three intersecting planes making a sharp corner,
4. too complex.



Surface projection

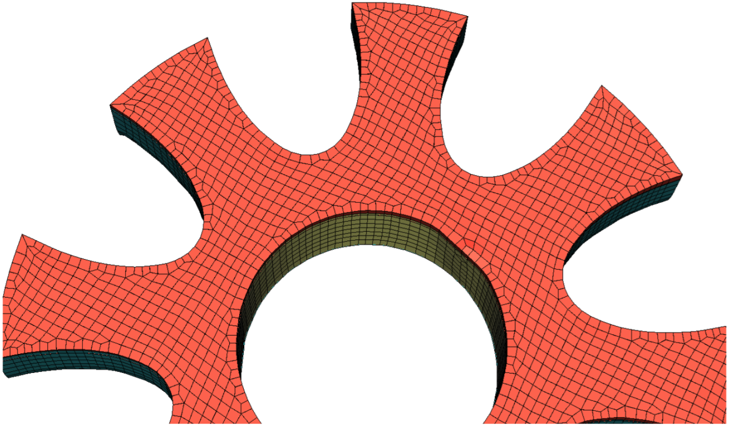


Problem: the octree generates a so called "staircase mesh". Hexes may have two or three faces to be projected on the same plane. Surface quad may have two edges to be projected on the same sharp line.



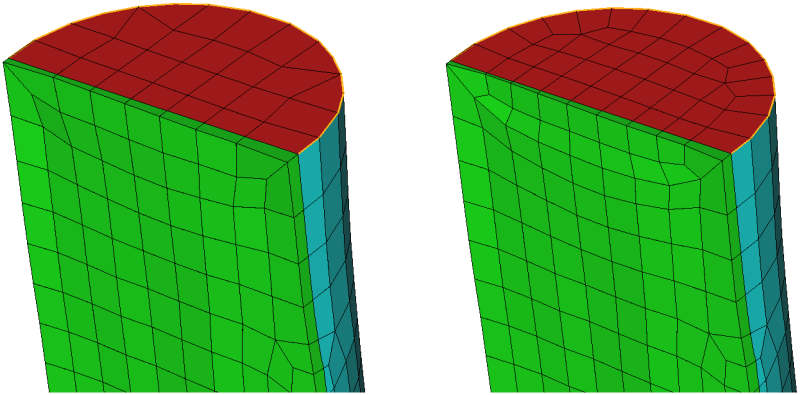
Solution: buffer-layer insertion. A first layer of hexes is inserted around the staircase mesh so that new boundary elements have only one face to be projected on the real geometry. Likewise, a second layer of element is inserted around sharp edges.

Boundary after projection



A buffer layer of elements is wrapped around the staircase mesh.

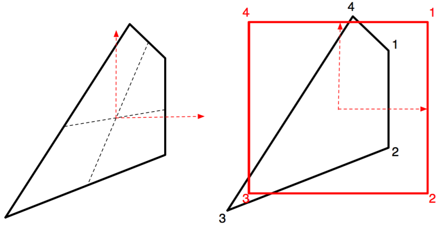
Ridge buffer-layer



Local buffer-layers are inserted alongside ridges.

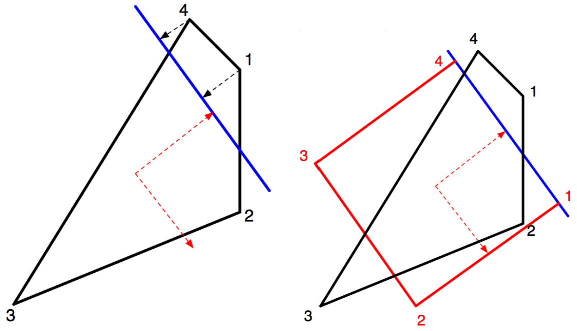
Quality optimization via node smoothing

Each solver has its own quality criterion and there is only one common ground: each hex must have a positive volume or jacobian. So we can only try to get as close as possible to the perfect cube.
The optimizing process finds the closest perfect cube from each hex and adds the contributions to a new set of nodes' coordinates which eventually will result in a better mesh.
This step has been multi-threaded with the help of the LP3lib and achieves a 11x speed-up on a 12-core Xeon E5.
A version has also been ported to OpenCL with the GM2lib and is 54 times faster than the CPU serial version on an AMD FirePro D700 GPU.



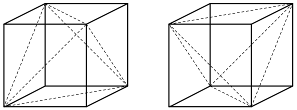
Boundary constrained optimization scheme

Boundary elements could be treated the same way as others and surface node could be projected on the geometry afterward: smoothing may push the nodes in one direction and projection may move them back !
Geometry should be part of the smoothing scheme: the perfect cube is rotated and pushed away so that its surface face matches the geometry it should represent.



Quality criterion

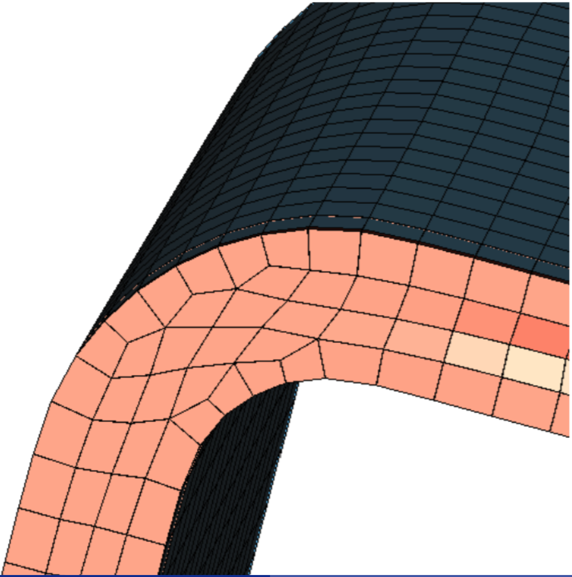
There as many quality criteria as there are solvers !
The only common ground is the jacobian measure which is good but not sufficient as it controls the volumes but not the face twisting.
Hence, I added two internal control tetrahedra.



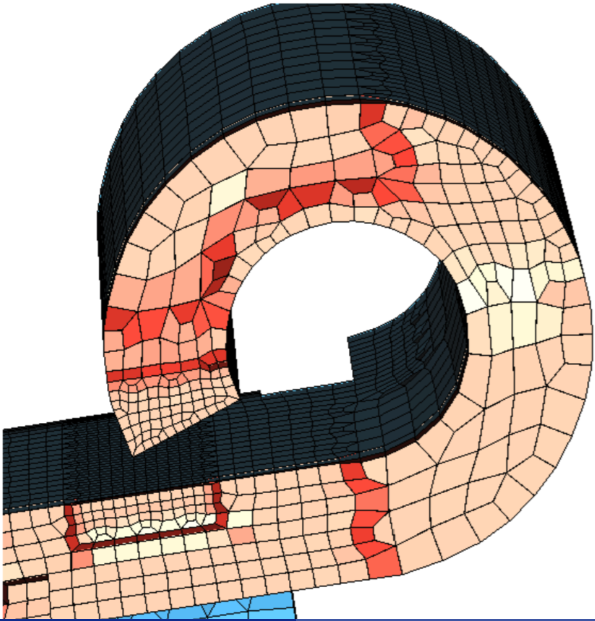
$$q = 24\sqrt{3}\frac{V_{min}}{(\sum\limits_{1 \leq i \leq 12} l_i)^{3/2}}$$

Where V_{min} is the minimum volume among the two sets of five tetrahedra cut from the hex, and l_i are the lengths of the twelve edges.

Strong limitation: the lack of topological operators on hex meshes

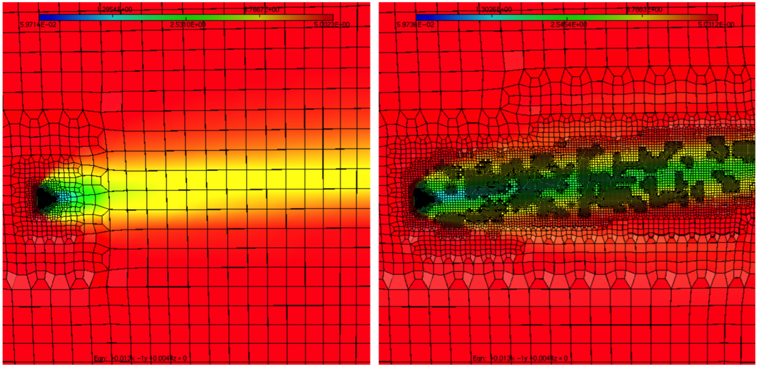


Strong limitation: no smooth size transition



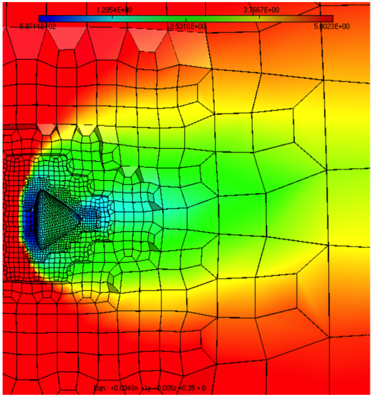
Mesh adaptation

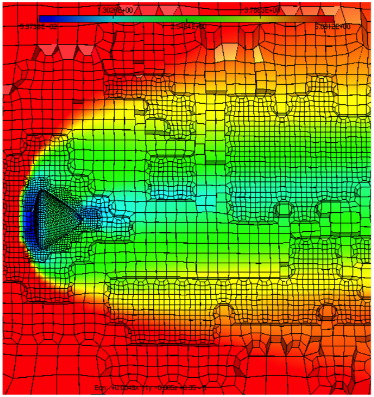
Only isotropic adaptation because of the octree method: no anisotropy like in tet meshes.
Twofold size ratio between neighboring elements : no smooth size transitions.



Pressure field around an Apollo capsule reentering the atmosphere.

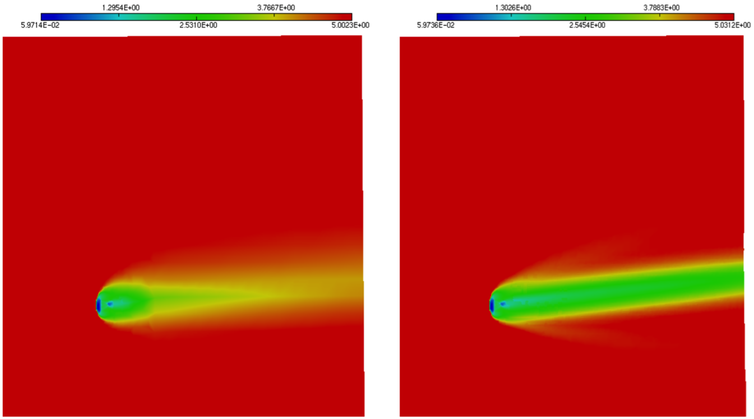
Mesh adaptation: the apollo capsule





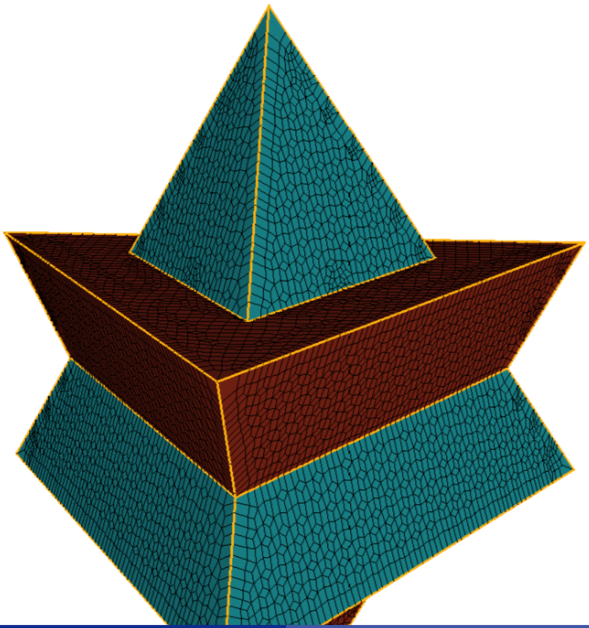
2:1 size transition may create unwanted adaptation artifacts.

Mesh adaptation: the apollo capsule

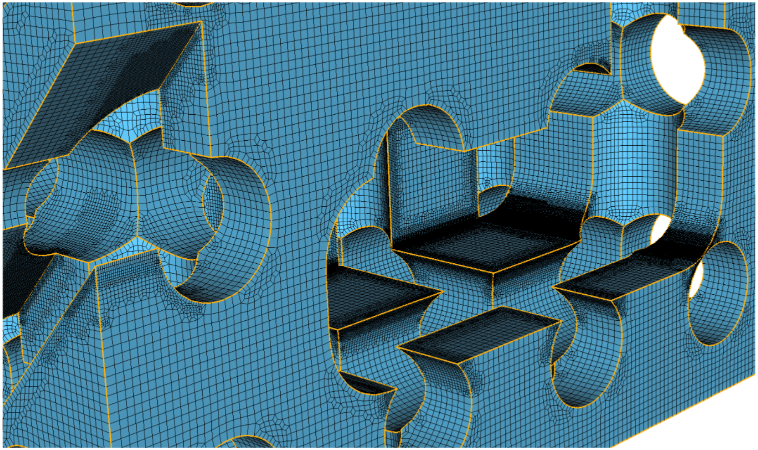


Left: original pressure field. Right: with adapted mesh.

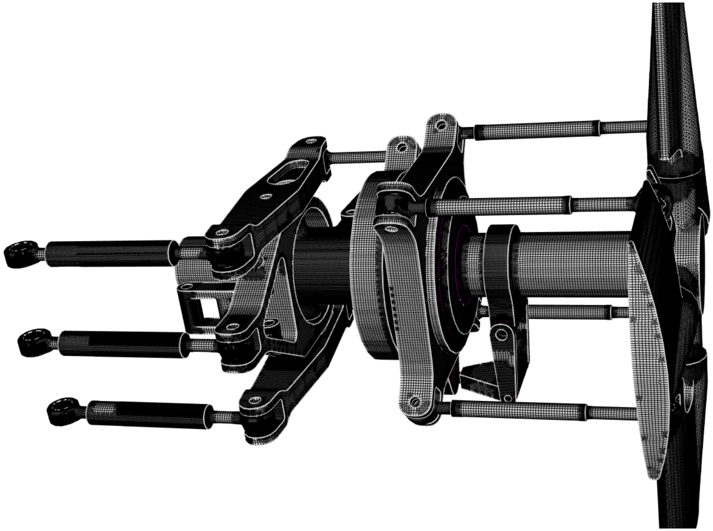
Academic examples



Academic examples

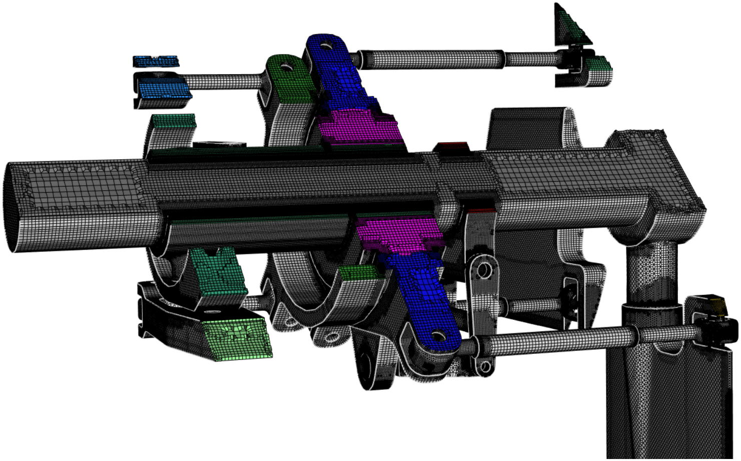

1.3 million elements, mean quality: 0.58, minimal: 0.01

Industrial examples

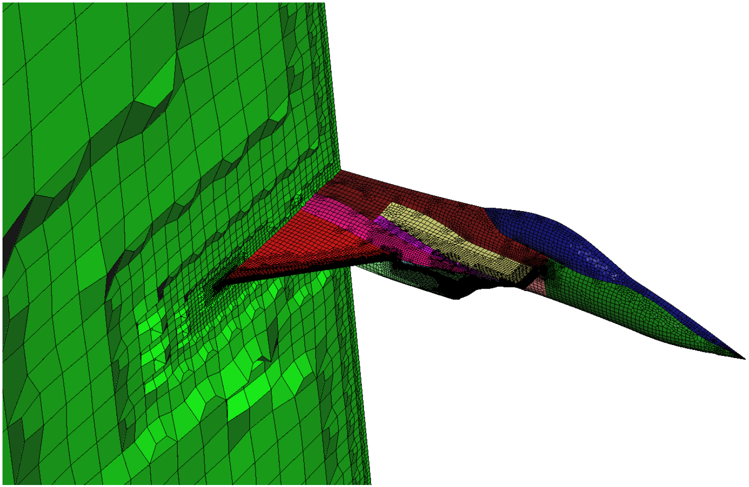


Helicopter's rotor: 5.7 millions hexes.

Industrial examples


Cut through the 34 non-manifold subdomains.

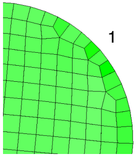
Industrial examples

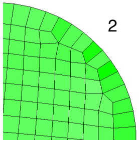


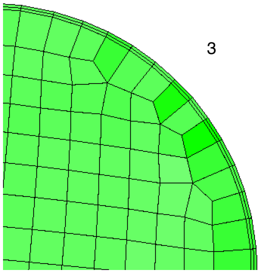
F15 jet fighter: the high refinement level near sharp features generated more than 2 million elements.

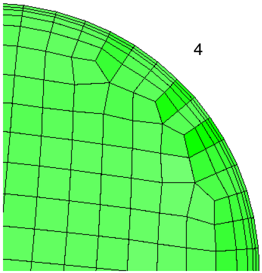
Boundary layers: the basics

Boundary layers can be inserted for CFD simulation purposes.
The user has control over the number, size, growth factor as well as the stem surface references, that is, the triangles references from which the layers will be extruded.
Additional blending layers may be inserted between the first set of physical layers and the free hex mesh in order to mitigate the size transition so that no element is more than two times bigger than its neighbors.





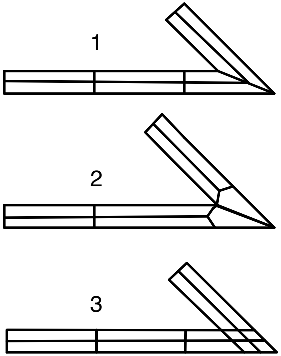




Boundary layers: concavities

How to grow good boundary layers inside a sharp concave angle ?

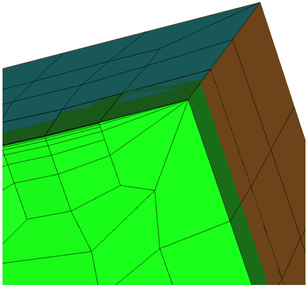
1. regular type: poor surface orthogonality, wide angles inside the layers,
2. wedge shaped ending: looks good, but what the solvers think about it ?
3. cross imprinting: seems to be the natural solution.

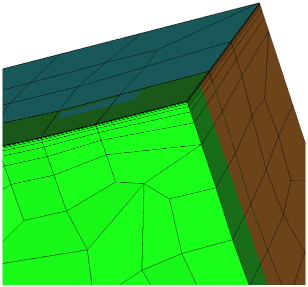


Should this choice be user driven, or automatically imposed by the geometry ?

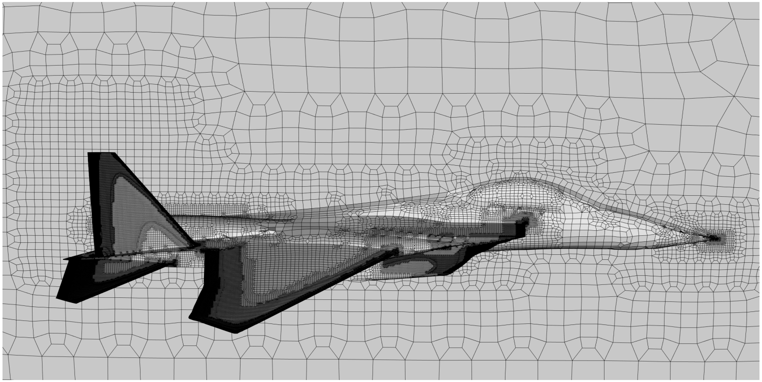
Boundary layers: imprinting

One set of layers may be specified for each surface references.
A pair of layered elements end up as a wedge like degenerate hex.
Transition between source surfaces and others are treated with either wedges or imprinting.



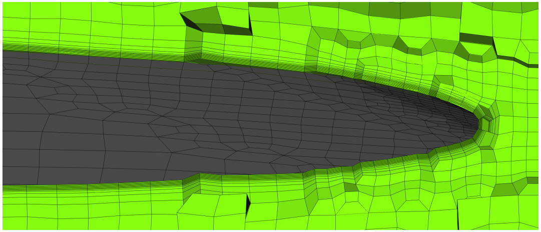


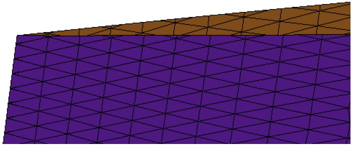
Boundary layers: F15

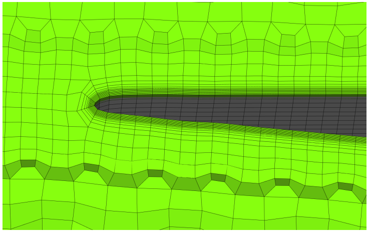


F15: 6.5 millions of hexes, 2.3m in the free volume, 4.2m in the boundary layers. Mean quality: 0.81, minimal: 0.013, CPU time: 35s, RAM: 3.4 GB

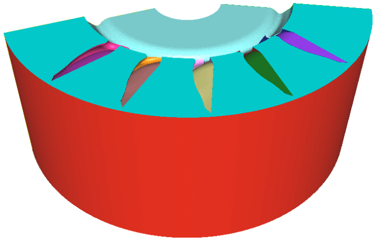
Boundary layers: F15 leading and trailing edges



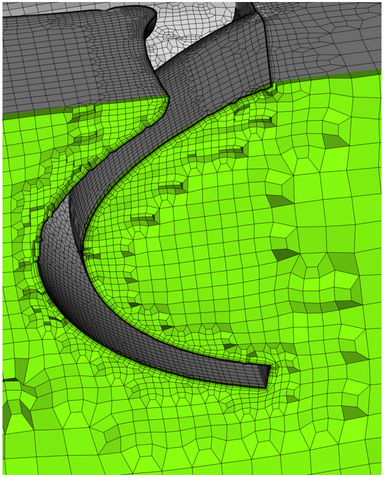




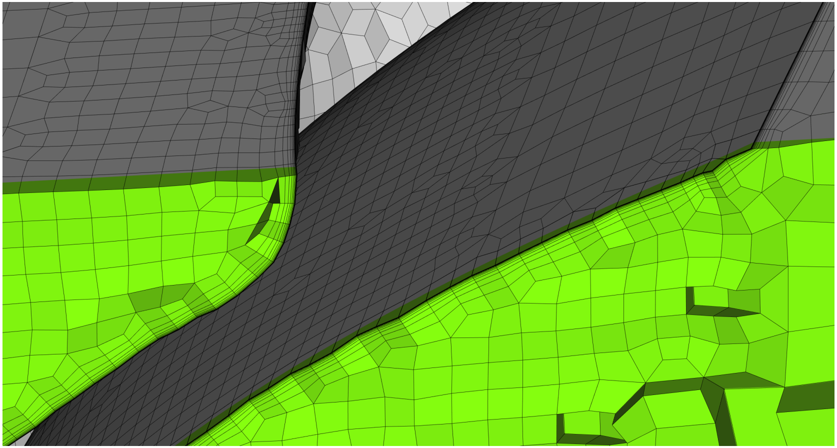
Boundary layers: rotor



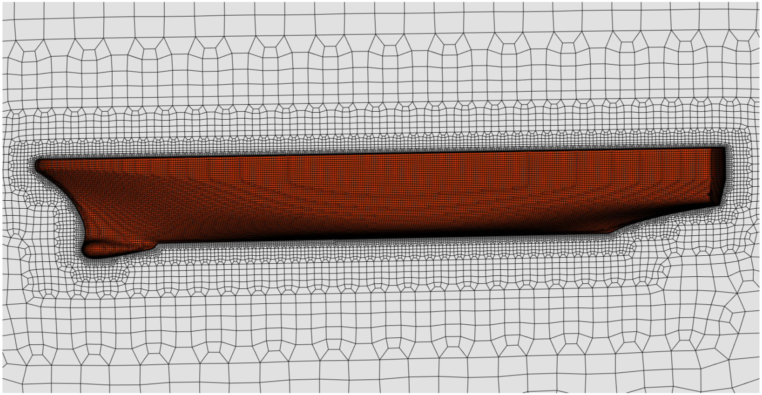
Rotor: 3.4 million hexes, free mesh: 1.1 millions, boundary layers: 2.3 millions, mean quality: 0.75, minimal quality: 0.002, meshing time: 20s, RAM: 1.8 GB



Boundary layers: rotor imprinting

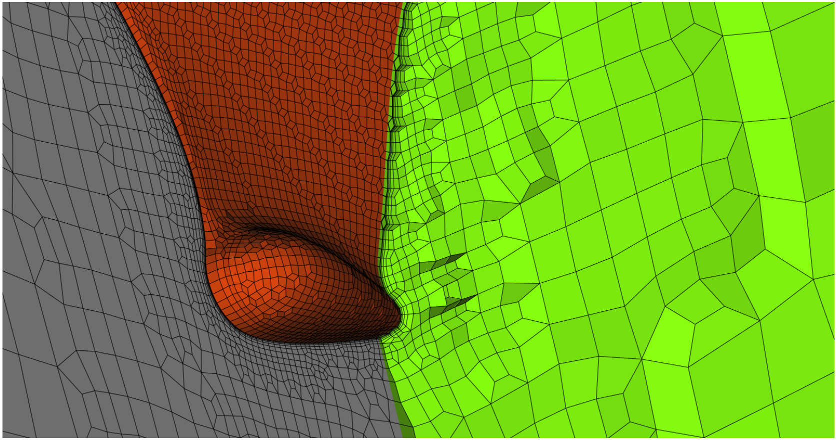


Boundary layers: DTMB ship hull

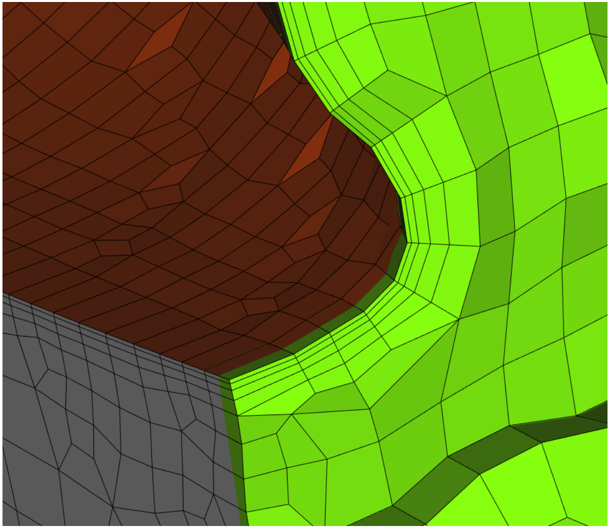


DTMB: 340,000 hexes, 200,000 in the free volume and 140,000 in the boundary layers. Mean quality: 0.72, minimal: 0.03, compute time 3s, RAM: 230 MB

Boundary layers: DTMB ship hull



Boundary layers: DTMB ship hull



Strengths and weaknesses of the method

Strengths:

- robust: it always produces a result, but it may not be the meshed you dreamed of...
- fast: 8.000.000 elements per minute on this laptop,
- simple: command-line program requiring few arguments like min & max sizes or sharp angle threshold,
- 100 % hexahedral and conformal meshes, no pyramids, prisms or hanging nodes.

Weaknesses:

- generates too many elements when it comes to thin geometries like blades, wings, etc...
- isotropic meshing only,
- Angles sharper than 60° are smoothed out: otherwise, hexes gets too distorted,
- unstructured meshes, although large parts of the mesh are grid-like.

Work under way

- reducing the number of elements with sheet removal at the expense of the grid structure,
- symmetrical meshes,
- cross imprinting boundary layers,
- high order elements,
- improving the capture of non-manifold edges.

Conclusion: there is still a long way to go to reach the "holly grail" of hex meshing !