An indirect approach to quad and hex mesh generation

Jean-François Remacle

Université catholique de Louvain (on leave).
Rice University.

http://www.geuz.org/gmsh

Paris, December 15th, 2014

Scope

- Indirect approaches to quad- and hex- meshing
  - The blossom algorithm of Edmunds and application to indirect quad meshing
  - Indirect hex meshing viewed as a clique problem
- The L^∞ distance in mesh generation
  - Delaunay and Voronoi in an L^∞ framework
  - Frame fields on manifolds
  - A frontal approach to quad- and hex-meshing

Remacle JF et. al Blossom-quad: a non-uniform quadrilateral mesh generator using a minimum cost perfect matching algorithm. *International Journal for Numerical Methods in Engineering* 2012.

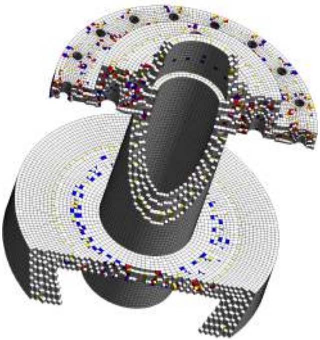
Remacle JF et al. A frontal Delaunay quad mesh generator using the L^∞ norm. *International Journal for Numerical Methods in Engineering* 2013.

Remacle JF et al. A frontal approach to hex-dominant mesh generation. *Advanced Modeling and Simulation in Engineering Sciences*, 1(1), 1-30, 2014.

Motivation for quad and hex meshes

- Quad and hex meshes are usually considered as superior to simplical meshes for finite element simulations.
- Discussions about if and why quadrilaterals are better than triangles are usually passionate in the finite element community.
- We will not try to argue about that thorny question here—but we assume that quad/hex meshes are indeed useful and in this paper we present a new way of generating such meshes.

Motivation for quad and hex meshes

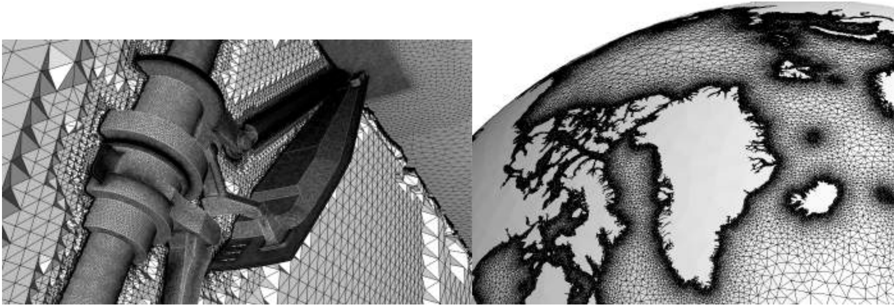


- Quad and hex meshes are usually considered as superior to simplical meshes for finite element simulations.
- Discussions about if and why quadrilaterals are better than triangles are usually passionate in the finite element community.
- We will not try to argue about that thorny question here—but we assume that quad/hex meshes are indeed useful and in this paper we present a new way of generating such meshes.

Motivation for quad and hex meshes

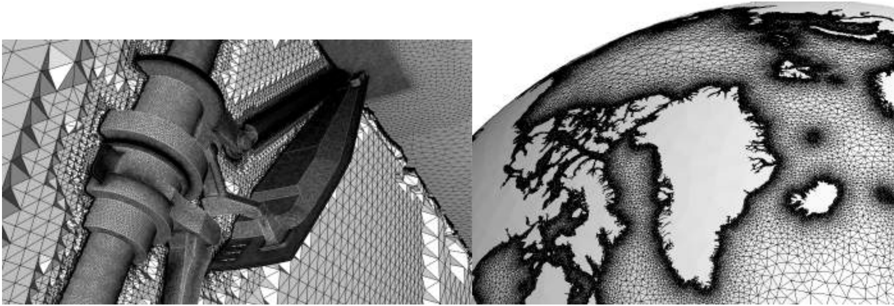
- Quad and hex meshes are usually considered as superior to simplical meshes for finite element simulations.
- Discussions about if and why quadrilaterals are better than triangles are usually passionate in the finite element community.
- We will not try to argue about that thorny question here—but we assume that quad/hex meshes are indeed useful and in this paper we present a new way of generating such meshes.

The indirect approach



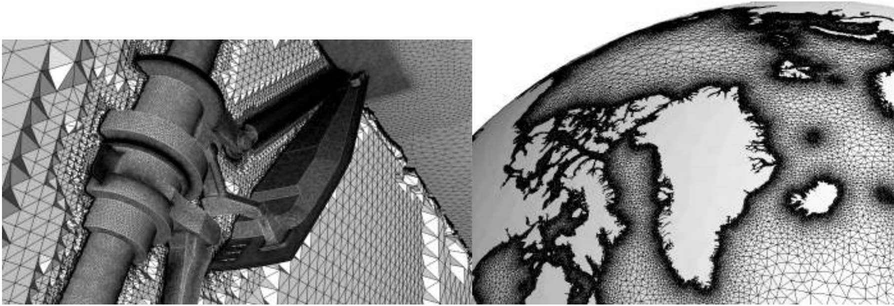
- Three dimensional mesh generation is a very hard problem, probably ill posed.
- We are now able to generate meshes made of triangles (2D) and tetrahedra in most of the cases ($> 99,9\%$!).
- It has taken us about 15 years of continuous efforts to reach that level of robustness.
- Start with a simplical mesh and transform it to a quad/hex mesh.

The indirect approach



- Three dimensional mesh generation is a very hard problem, probably ill posed.
- We are now able to generate meshes made of triangles (2D) and tetrahedra in most of the cases ($> 99,9\%$!).
- It has taken us about 15 years of continuous efforts to reach that level of robustness.
- Start with a simplical mesh and transform it to a quad/hex mesh.

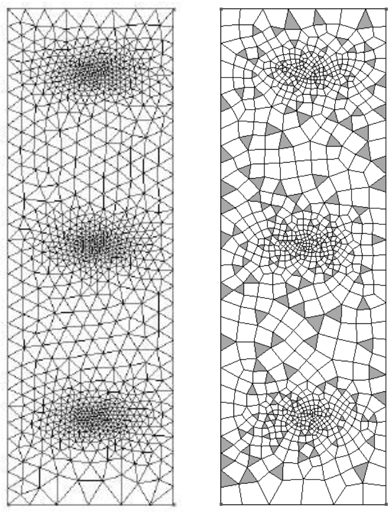
The indirect approach



- Three dimensional mesh generation is a very hard problem, probably ill posed.
- We are now able to generate meshes made of triangles (2D) and tetrahedra in most of the cases ($> 99,9\%$!).
- It has taken us about 15 years of continuous efforts to reach that level of robustness.
- Start with a simplical mesh and transform it to a quad/hex mesh.

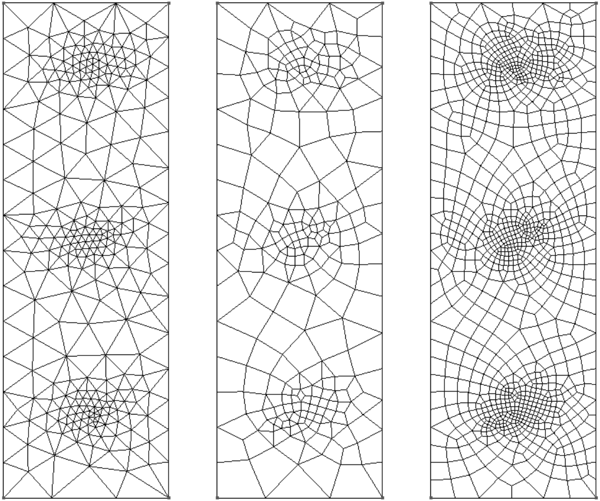
Indirect approach: a non-optimal matching algorithm

Mesh size $h(x,y) = 0.1 + 0.08\sin(3x)\cos(6y)$, 836 quads and 240 triangles



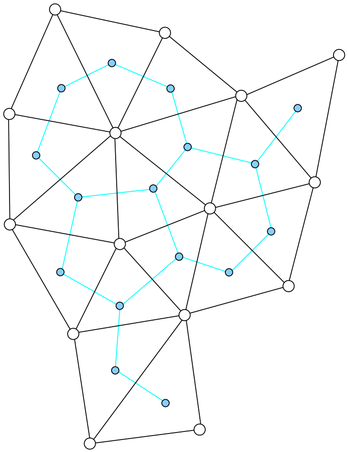
Indirect approach: a full-quad approach

$h(x, y) = 0.1 + 0.08 \sin(3x) \cos(6y)$, only quads



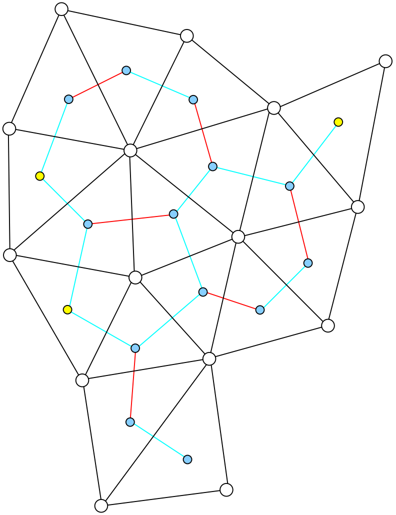
Indirect approach: the Blossom-Quad approach

We build $G(V, E, c)$ an undirected weighted graph. Here, V is the set of n_V vertices, E is the set of n_E undirected edges and $c(E) = \sum c(e_{ij})$ is an edge-based cost function, i.e., the sum of all weights associated to every edge $e_{ij} \in E$ of the graph.



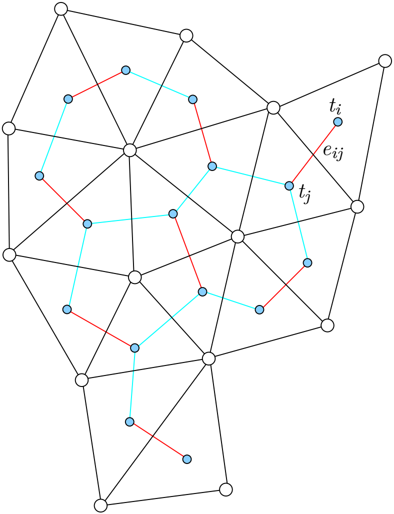
Indirect approach: the Blossom-Quad approach

A *matching* is a subset $E' \subseteq E$ such that each node of V has at most one incident edge in E'.



Indirect approach: the Blossom-Quad approach

A matching is perfect if each node of V has exactly one incident edge in E'. A perfect matching is optimum if $c(E')$ is minimum among all possible perfect matchings.



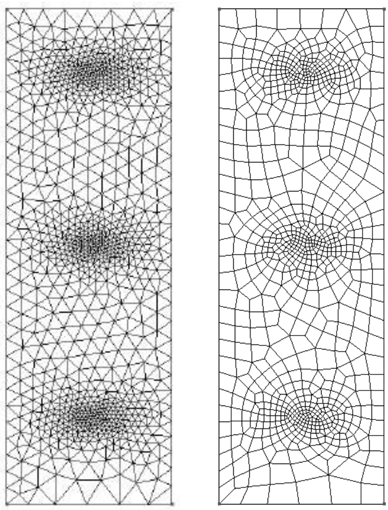
Indirect approach: the Blossom-Quad approach

- In 1965, Edmonds invented the *Blossom algorithm* that solves the problem of optimum perfect matching in polynomial time. A straightforward implementation of Edmonds's algorithm requires $\mathcal{O}(n_V^2 n_E)$ operations.
- Since then, the worst-case complexity of the Blossom algorithm has been steadily improving. Both Lawler and Gabow achieved a running time of $\mathcal{O}(n_V^3)$. Galil, Micali and Gabow improved it to $\mathcal{O}(n_V n_E \log(n_V))$. The current best known result in terms of n_V and n_E is $\mathcal{O}(n_V(n_E + \log n_V))$.
- There is also a long history of computer implementations of the Blossom algorithm, starting with the Blossom I code of Edmonds, Johnson and Lockhart. In this paper, our implementation makes use of the Blossom IV code of Cook and Rohe[1] that has been considered for several years as the fastest available implementation of the Blossom algorithm. Gmsh redistributes this piece of code.

[1] http://www2.isye.gatech.edu/?wcook/blossom4/.

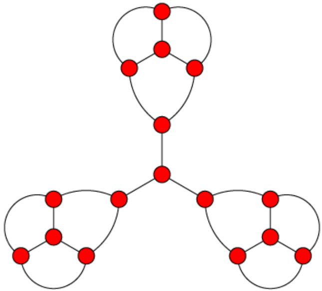
Indirect approach: Blossom-Quad

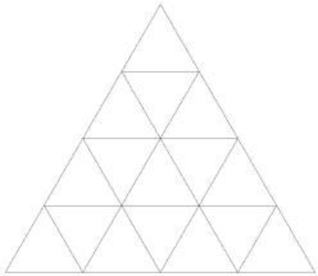
$h(x,y) = 0.1 + 0.08\sin(3x)\cos(6y)$, only quads



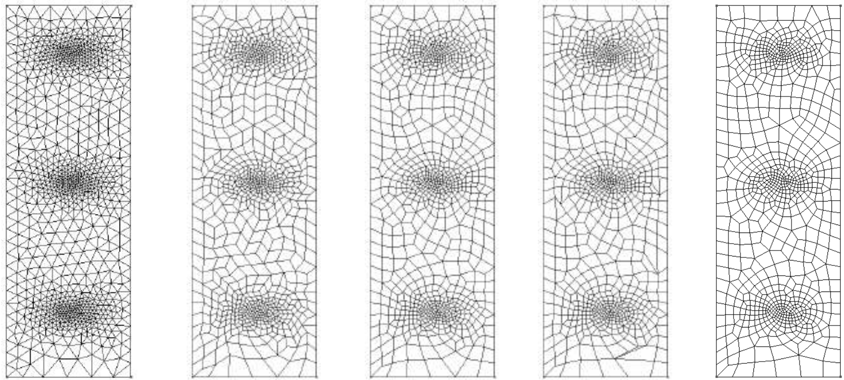
Existence of perfect matchings

- There is no guarantee that even one single perfect matching exists in a given graph.
- Tutte: A graph, $G = (V, E)$, has a perfect matching if and only if for every subset U of V, the subgraph induced by $V - U$ has at most $|U|$ connected components with an odd number of vertices
- Petersen: Every cubic graph with no bridges has a perfect matching.





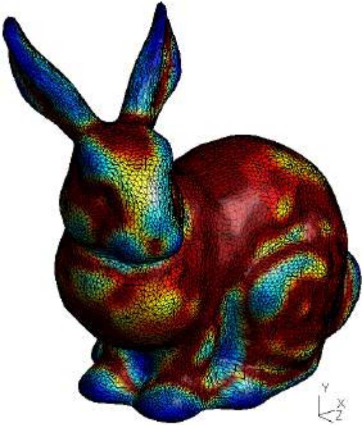
The Blossom-Quad algorithm



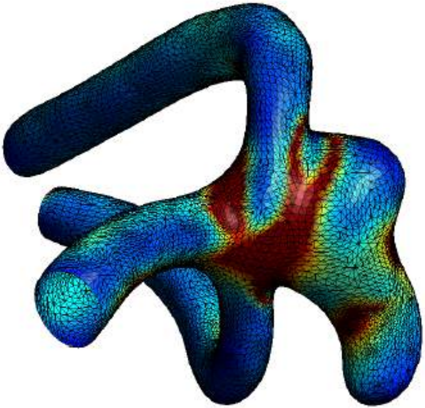
Initial triangulation | Raw Blossom application | Vertex smoothing | Topological optimization | Final mesh

Quad mesh generation applied to STL models

$h(\vec{x}) = \frac{2\pi R(\vec{x})}{N_p}$, with $R(\vec{x}) = \frac{1}{\kappa(\vec{x})}, N_P = 50$

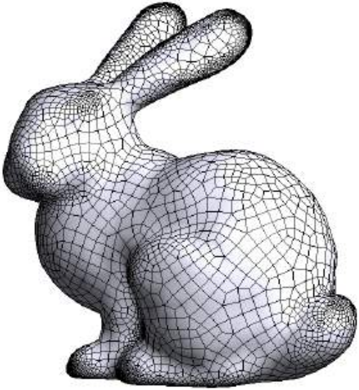


Stanford Bunny

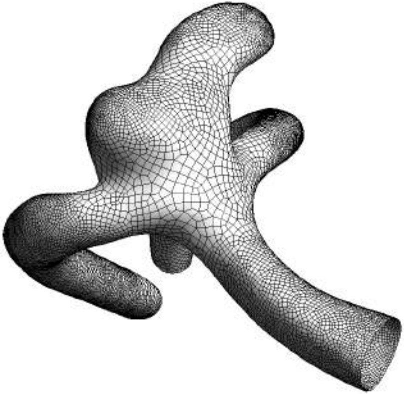


Aneurysm

Quad mesh generation applied to STL models



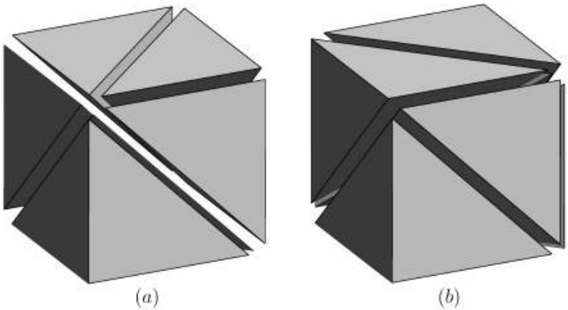
$\tau = 0.842549$



$\tau = 0.856247$

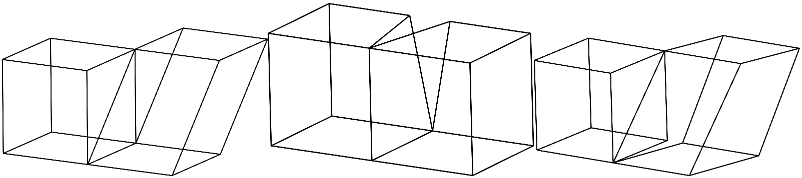
- The overall remeshing procedure for both STL examples takes only $20s$ ($5s$ for the Blossom-Quad).
- The quad-dominant meshing algorithm of [Levy et al.] takes $271s$ for the remeshing of the Stanford bunny.

Extension to 3D



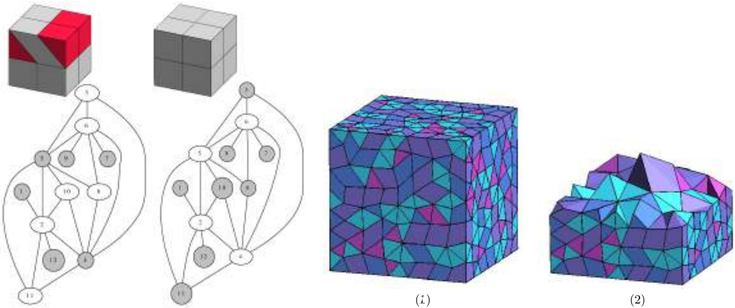
- A single hex can be made of 5,6 or 7 tets.
- A (finite) subtle set of non-compatibilities exists.
- The problem can be written as an independant set problem: a $(\mathcal{O})(n \log n)$ greedy algorithm is presently used.
- The rest of the domain is filled with prisms, pyramids and some tets subsist.
- Full hex mesh can be obtained by splitting. Yet, how to split a pyramid into hexes is an open problem.

Extension to 3D



- A single hex can be made of 5,6 or 7 tets.
- A (finite) subtle set of non-compatibilities exists.
- The problem can be written as an independant set problem: a $(\mathcal{O})(n \log n)$ greedy algorithm is presently used.
- The rest of the domain is filled with prisms, pyramids and some tets subsist.
- Full hex mesh can be obtained by splitting. Yet, how to split a pyramid into hexes is an open problem.

Extension to 3D



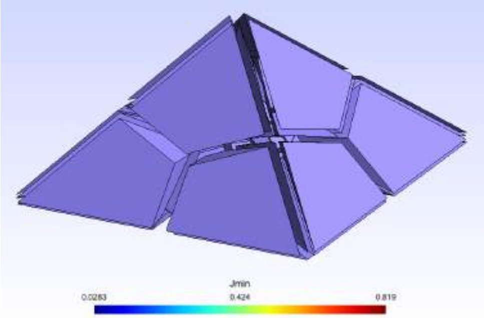
(1) (2)

- A single hex can be made of 5,6 or 7 tets.
- A (finite) subtle set of non-compatibilities exists.
- The problem can be written as an independant set problem: a $(\mathcal{O})(n \log n)$ greedy algorithm is presently used.
- The rest of the domain is filled with prisms, pyramids and some tets subsist.
- Full hex mesh can be obtained by splitting. Yet, how to split a pyramid into hexes is an open problem.

Extension to 3D

- A single hex can be made of 5,6 or 7 tets.
- A (finite) subtle set of non-compatibilities exists.
- The problem can be written as an independant set problem: a $(\mathcal{O})(n \log n)$ greedy algorithm is presently used.
- The rest of the domain is filled with prisms, pyramids and some tets subsist.
- Full hex mesh can be obtained by splitting. Yet, how to split a pyramid into hexes is an open problem.

Extension to 3D

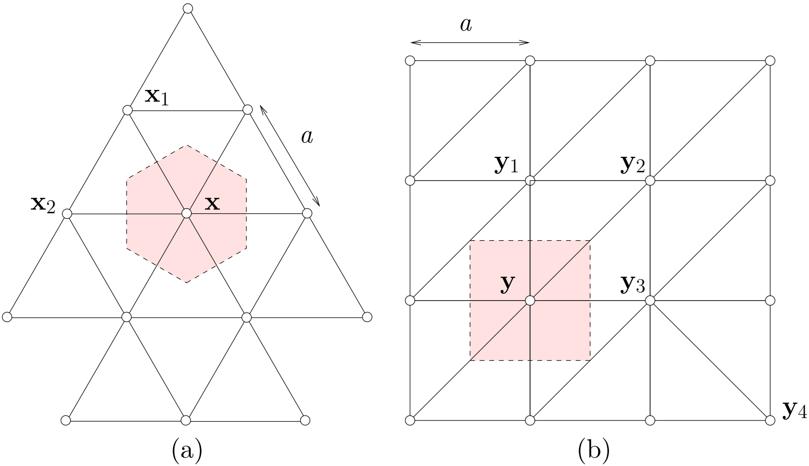


- A single hex can be made of 5,6 or 7 tets.
- A (finite) subtle set of non-compatibilities exists.
- The problem can be written as an independant set problem: a $(\mathcal{O})(n \log n)$ greedy algorithm is presently used.
- The rest of the domain is filled with prisms, pyramids and some tets subsist.
- Full hex mesh can be obtained by splitting. Yet, how to split a pyramid into hexes is an open problem.

Problems

- Simplical meshes are essentially isotropic.
- Recombination rate is low (about 20 % in volume).
- Quad-hex meshes should contain a minimum number of singularities and follow directions (frame fields).
- Live example ...

Indirect approach: a distance problem



Voronoi cells of one vertex that belongs either to mesh of a equilateral triangles (a) or of right triangles (b).

Indirect approach: an obvious problem

- Consider a uniform mesh of R^2 made of equilateral triangles of size a.
- The Voronoi cell relative to each vertex of this mesh is an hexagon of area $a^2\sqrt{3}/2$. The number $2/(a^2\sqrt{3})$, is the number of points per unit of surface of this mesh.
- Assume now a uniform mesh of R^2 made of squares of size a.
- The Voronoi cell relative to each vertex of this mesh is a square of area a^2.
- This means that filling R^2 with equilateral triangles requires $2/\sqrt{3}$ times more vertices than filling the same space with squares.
- So, even though it is always possible to build a mesh made of quadrangles by recombining triangles, a good triangular mesh made of equilateral triangles contains about $2/\sqrt{3}$ times too many to make a good quadrilateral mesh.

Indirect approach: an obvious problem

- Consider a uniform mesh of R^2 made of equilateral triangles of size a.
- The Voronoi cell relative to each vertex of this mesh is an hexagon of area $a^2\sqrt{3}/2$. The number $2/(a^2\sqrt{3})$, is the number of points per unit of surface of this mesh.
- Assume now a uniform mesh of R^2 made of squares of size a.
- The Voronoi cell relative to each vertex of this mesh is a square of area a^2.
- This means that filling R^2 with equilateral triangles requires $2/\sqrt{3}$ times more vertices than filling the same space with squares.
- So, even though it is always possible to build a mesh made of quadrangles by recombining triangles, a good triangular mesh made of equilateral triangles contains about $2/\sqrt{3}$ times too many to make a good quadrilateral mesh.

Indirect approach: an obvious problem

- Consider a uniform mesh of R^2 made of equilateral triangles of size a.
- The Voronoi cell relative to each vertex of this mesh is an hexagon of area $a^2\sqrt{3}/2$. The number $2/(a^2\sqrt{3})$, is the number of points per unit of surface of this mesh.
- Assume now a uniform mesh of R^2 made of squares of size a.
- The Voronoi cell relative to each vertex of this mesh is a square of area a^2.
- This means that filling R^2 with equilateral triangles requires $2/\sqrt{3}$ times more vertices than filling the same space with squares.
- So, even though it is always possible to build a mesh made of quadrangles by recombining triangles, a good triangular mesh made of equilateral triangles contains about $2/\sqrt{3}$ times too many to make a good quadrilateral mesh.

Indirect approach: an obvious problem

- Consider a uniform mesh of R^2 made of equilateral triangles of size a.
- The Voronoi cell relative to each vertex of this mesh is an hexagon of area $a^2\sqrt{3}/2$. The number $2/(a^2\sqrt{3})$, is the number of points per unit of surface of this mesh.
- Assume now a uniform mesh of R^2 made of squares of size a.
- The Voronoi cell relative to each vertex of this mesh is a square of area a^2.
- This means that filling R^2 with equilateral triangles requires $2/\sqrt{3}$ times more vertices than filling the same space with squares.
- So, even though it is always possible to build a mesh made of quadrangles by recombining triangles, a good triangular mesh made of equilateral triangles contains about $2/\sqrt{3}$ times too many to make a good quadrilateral mesh.

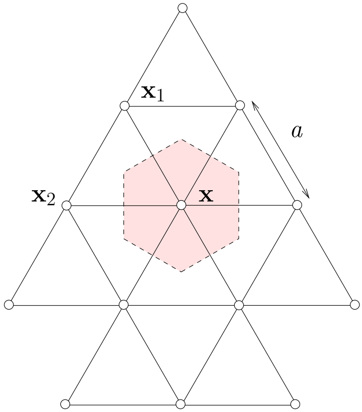
Indirect approach: an obvious problem

- Consider a uniform mesh of R^2 made of equilateral triangles of size a.
- The Voronoi cell relative to each vertex of this mesh is an hexagon of area $a^2\sqrt{3}/2$. The number $2/(a^2\sqrt{3})$, is the number of points per unit of surface of this mesh.
- Assume now a uniform mesh of R^2 made of squares of size a.
- The Voronoi cell relative to each vertex of this mesh is a square of area a^2.
- This means that filling R^2 with equilateral triangles requires $2/\sqrt{3}$ times more vertices than filling the same space with squares.
- So, even though it is always possible to build a mesh made of quadrangles by recombining triangles, a good triangular mesh made of equilateral triangles contains about $2/\sqrt{3}$ times too many to make a good quadrilateral mesh.

Indirect approach: an obvious problem

- Consider a uniform mesh of R^2 made of equilateral triangles of size a.
- The Voronoi cell relative to each vertex of this mesh is an hexagon of area $a^2\sqrt{3}/2$. The number $2/(a^2\sqrt{3})$, is the number of points per unit of surface of this mesh.
- Assume now a uniform mesh of R^2 made of squares of size a.
- The Voronoi cell relative to each vertex of this mesh is a square of area a^2.
- This means that filling R^2 with equilateral triangles requires $2/\sqrt{3}$ times more vertices than filling the same space with squares.
- So, even though it is always possible to build a mesh made of quadrangles by recombining triangles, a good triangular mesh made of equilateral triangles contains about $2/\sqrt{3}$ times too many to make a good quadrilateral mesh.

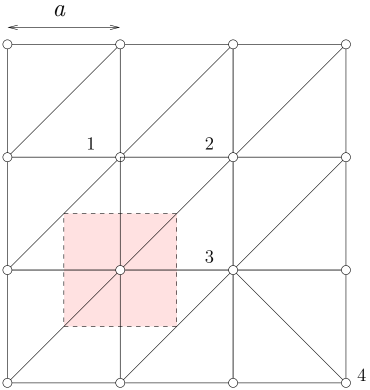
Distances and Norms



- The equilateral mesh has all its edges of size a in the Euclidian norm.
- The mesh made of right triangles has "long" edges of size $a\sqrt{2}$.
- Find out a way to measure distances in a way that all edges of the right triangles are of size a.
- One could think of using standard metric techniques. Yet, this is not the right way to go: edges at 45 degree and -45 degree should have the same size.
- Use the L^∞-norm distance:

$$\begin{aligned} \|\mathbf{x}_2 - \mathbf{x}_1\|_\infty &= \lim_{p \to \infty} \|\mathbf{x}_2 - \mathbf{x}_1\|_p \\ &= \max(|x_2 - x_1|, |y_2 - y_1|) \end{aligned}$$

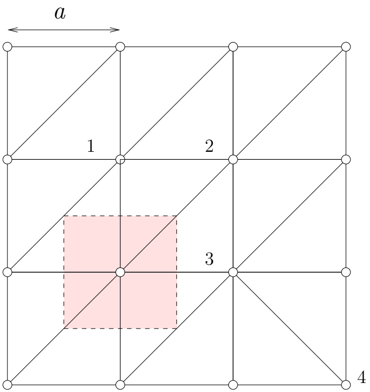
Distances and Norms



- The equilateral mesh has all its edges of size a in the Euclidian norm.
- The mesh made of right triangles has "long" edges of size $a\sqrt{2}$.
- Find out a way to measure distances in a way that all edges of the right triangles are of size a.
- One could think of using standard metric techniques. Yet, this is not the right way to go: edges at 45 degree and -45 degree should have the same size.
- Use the L^∞-norm distance:

$$\begin{aligned} \|\mathbf{x}_2 - \mathbf{x}_1\|_\infty &= \lim_{p\to\infty} \|\mathbf{x}_2 - \mathbf{x}_1\|_p \\ &= \max\left(|x_2 - x_1|, |y_2 - y_1|\right) \end{aligned}$$

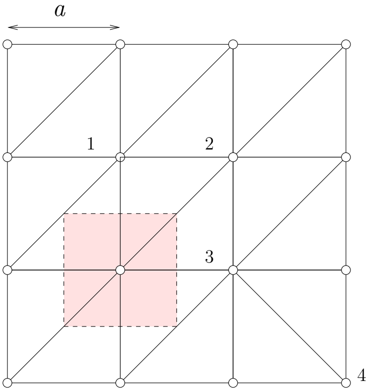
Distances and Norms



- The equilateral mesh has all its edges of size a in the Euclidian norm.
- The mesh made of right triangles has "long" edges of size $a\sqrt{2}$.
- Find out a way to measure distances in a way that all edges of the right triangles are of size a.
- One could think of using standard metric techniques. Yet, this is not the right way to go: edges at 45 degree and -45 degree should have the same size.
- Use the L^∞-norm distance:

$$
\begin{aligned}
\|\mathbf{x}_2 - \mathbf{x}_1\|_\infty &= \lim_{p\to\infty} \|\mathbf{x}_2 - \mathbf{x}_1\|_p \\
&= \max\left(|x_2 - x_1|, |y_2 - y_1|\right)
\end{aligned}
$$

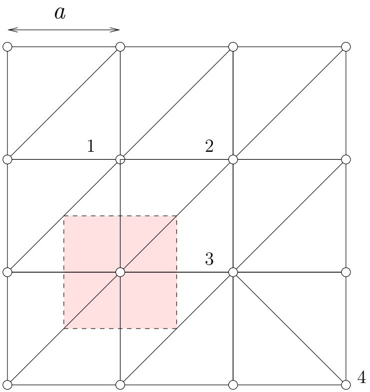
Distances and Norms



- The equilateral mesh has all its edges of size a in the Euclidian norm.
- The mesh made of right triangles has "long" edges of size $a\sqrt{2}$.
- Find out a way to measure distances in a way that all edges of the right triangles are of size a.
- One could think of using standard metric techniques. Yet, this is not the right way to go: edges at 45 degree and -45 degree should have the same size.
- Use the L^∞-norm distance:

$$\begin{aligned} \|\mathbf{x}_2 - \mathbf{x}_1\|_\infty &= \lim_{p \to \infty} \|\mathbf{x}_2 - \mathbf{x}_1\|_p \\ &= \max\left(|x_2 - x_1|, |y_2 - y_1|\right) \end{aligned}$$

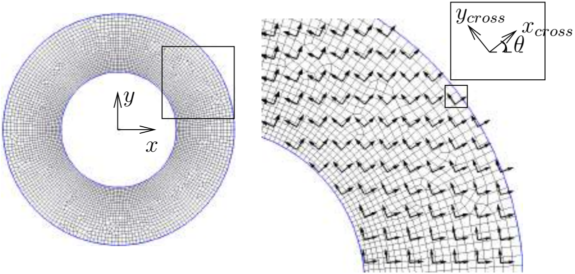
Distances and Norms



- The equilateral mesh has all its edges of size a in the Euclidian norm.
- The mesh made of right triangles has "long" edges of size $a\sqrt{2}$.
- Find out a way to measure distances in a way that all edges of the right triangles are of size a.
- One could think of using standard metric techniques. Yet, this is not the right way to go: edges at 45 degree and -45 degree should have the same size.
- Use the L^∞-norm distance:

$$\begin{aligned} \|\mathbf{x}_2 - \mathbf{x}_1\|_\infty &= \lim_{p\to\infty} \|\mathbf{x}_2 - \mathbf{x}_1\|_p \\ &= \max\left(|x_2 - x_1|, |y_2 - y_1|\right) \end{aligned}$$

Cross Fields



- The L^{∞} norm is not invariant by rotation → quads should be oriented
- A cross field (in 2D) is a scalar field $\theta(\mathbf{x})$ that gives the orientations of a local system of axis at point $\mathbf{x}$.
- The local value of the cross field $\theta(x, y)$ is defined only up to rotations by $\pi/2$, we choose to propagate

$$\alpha(x, y) = a(x, y) + ib(x, y) = e^{4i\theta(u,v)}$$

through a harmonic map.

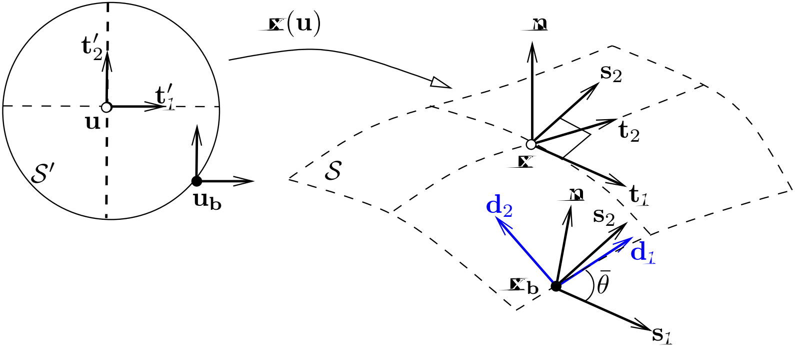


Figure : Surface parametrization and construction of the frame field (in blue) .

Cross Fields on surfaces

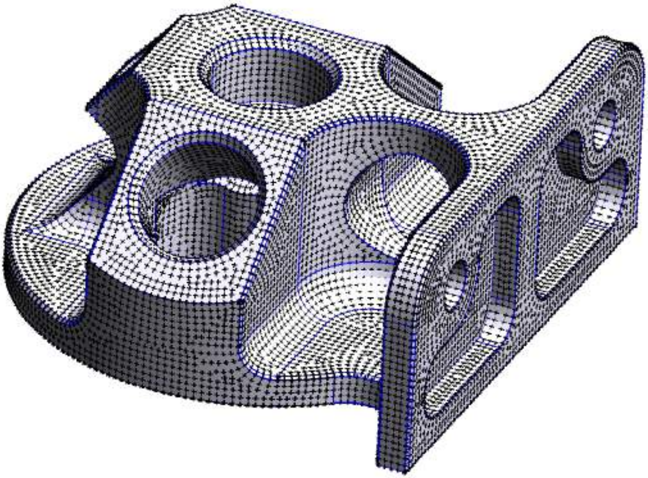
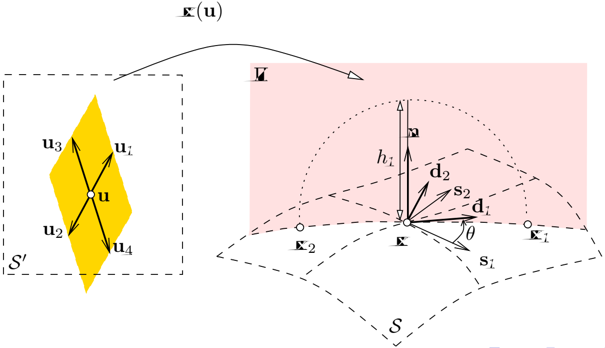
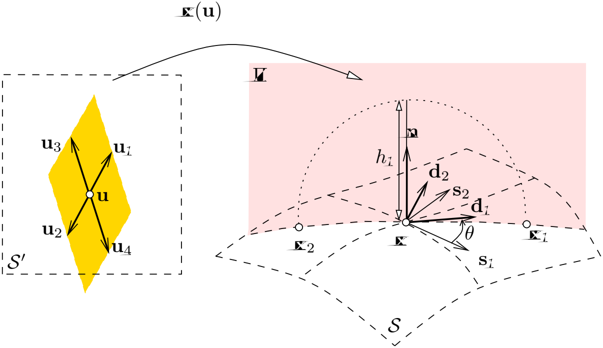


Figure : Cross fields on the surfaces of a mechanical part.

- In a perfect quad mesh, each vertex is connected to four neighboring vertices forming a cross parallel to the cross field.
- In our approach, four prospective points $\mathbf{x}_i$, $i=1,\ldots,4$ are constructed in the neighborhood of point $\mathbf{x}$ with the aim of generating the perfect situation.
- Points $\mathbf{x}_1$ and $\mathbf{x}_2$ are constructed as the intersection of the surface $\mathcal{S}$ with a circle of radius h_1, centered on $\mathbf{x}$ and situated in the plane Π of normal $\mathbf{d}_2$



- Each vertex of the boundary is inserted in a fifo queue.
- Vertex $\mathbf{x}$ at the head of the queue is removed and its four prospective neighbors $\mathbf{x}_i$ are computed.
- A new vertex $\mathbf{x}_i$ is inserted at the tail of the queue if the following conditions are satisfied: (i) vertex $\mathbf{x}_i$ is inside the domain and (ii) vertex $\mathbf{x}_i$ is not too close to any of the vertices that have already been inserted.



R-trees

R-trees are tree data structures used for spatial access methods, i.e., for indexing multi-dimensional information such as geographical coordinates, rectangles or polygons.

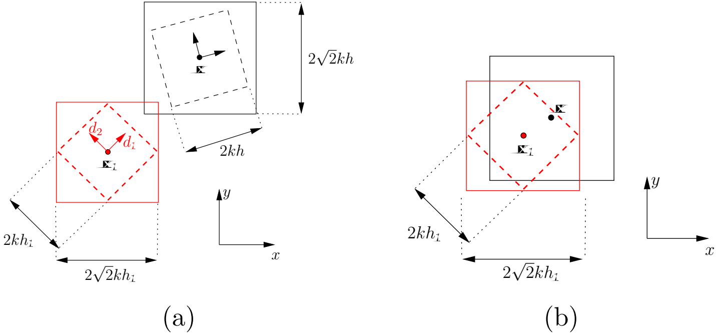


Figure : Frontal approach for vertex insertion for a 2D planar surface: a) the prospective vertex can be inserted because the infinite distance between the $\mathbf{x}_1$ and $\mathbf{x}$ is greater than kh_1 (x is outside the red dotted square), b) the prospective vertex is not inserted because the distance between the vertices is smaller than kh_1 (x is inside the red dotted square).

Point insertion

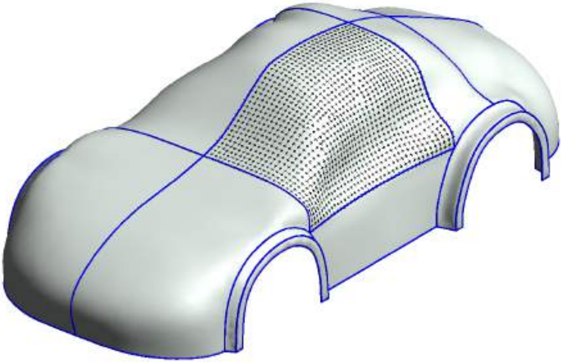


Figure : Different stages of the surface meshing process.

Point insertion

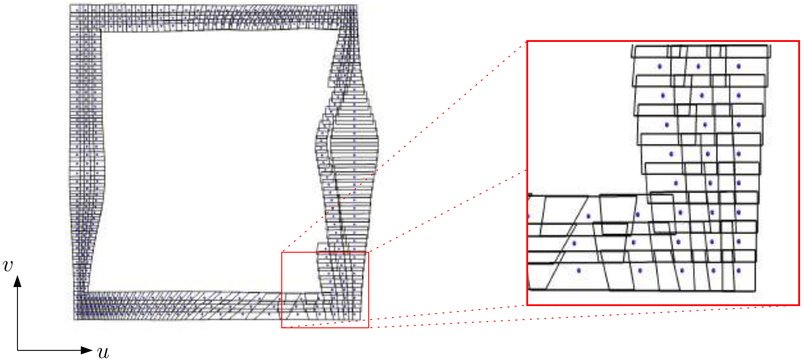


Figure : Different stages of the surface meshing process.

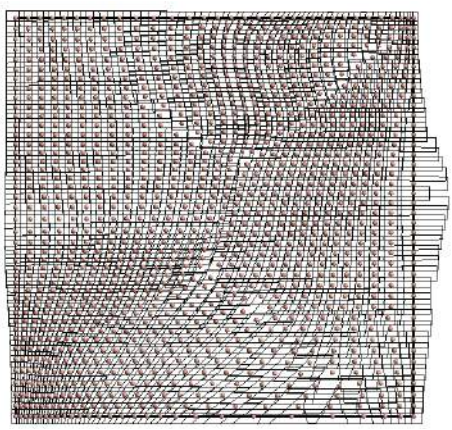


Figure : Different stages of the surface meshing process.

Point insertion

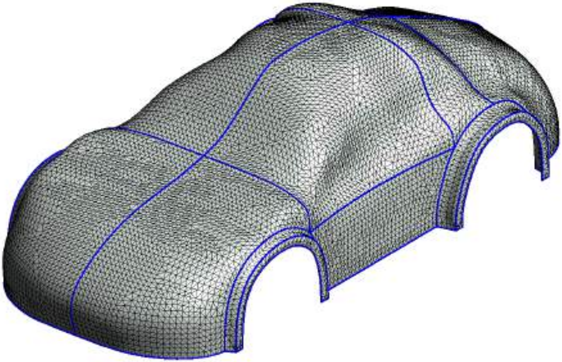
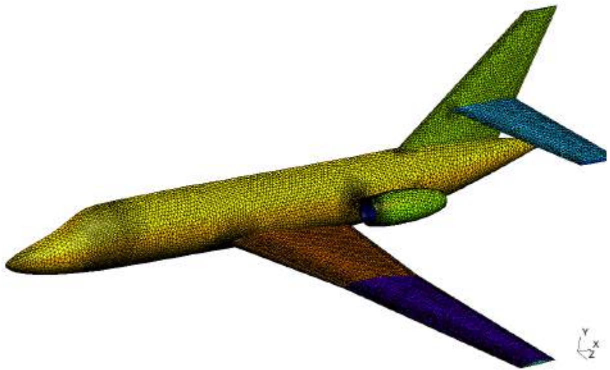


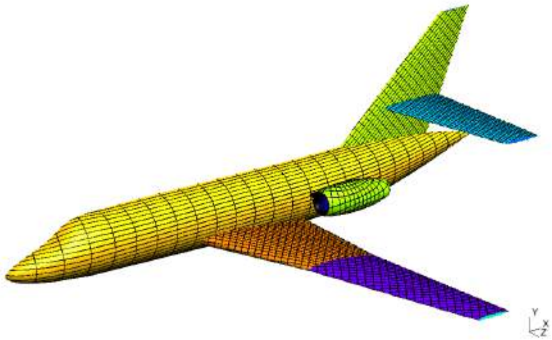
Figure : Different stages of the surface meshing process.

Falcon

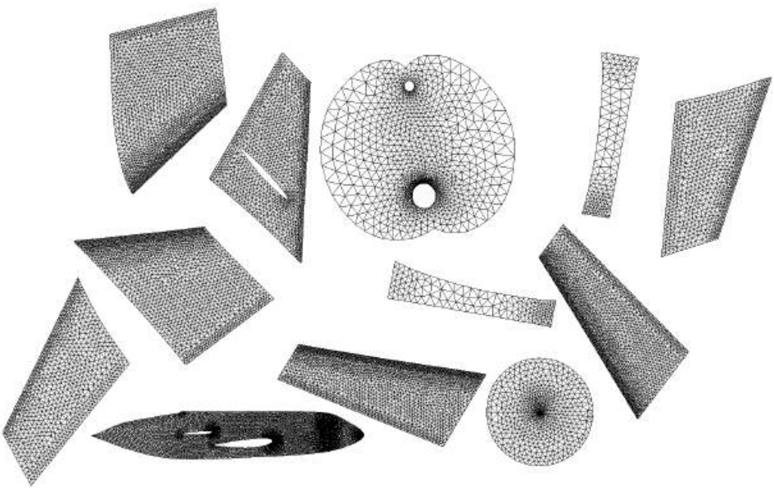


Surface triangular mesh of a Falcon aircraft (left) and contour lines of the conformal parametrizations (right). Colors are indicative of the different surfaces of the model.

Falcon

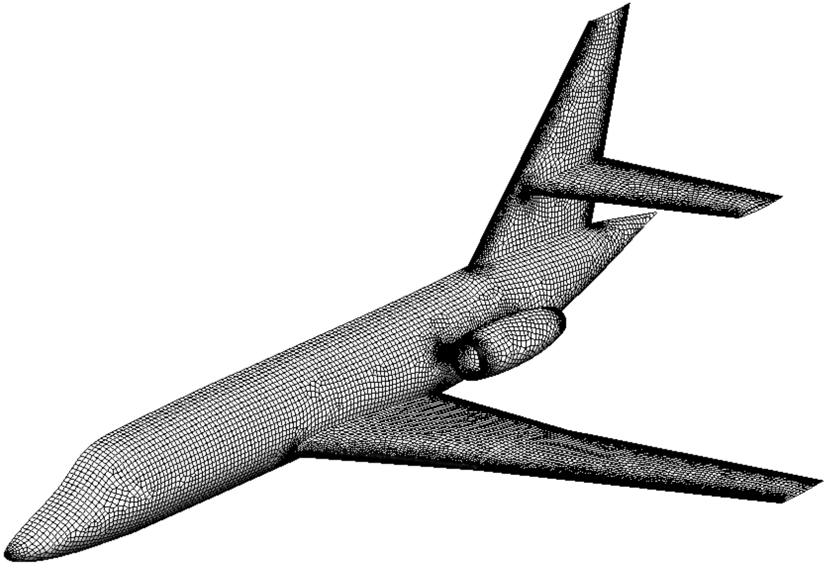


Surface triangular mesh of a Falcon aircraft (left) and contour lines of the conformal parametrizations (right). Colors are indicative of the different surfaces of the model.

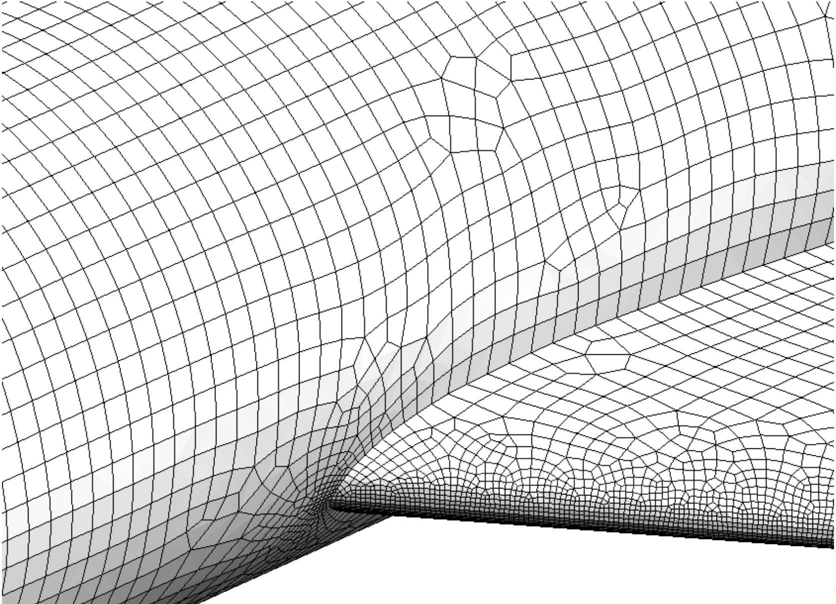


Atlas the Falcon aircraft in the $\{u, v\}$ plane.

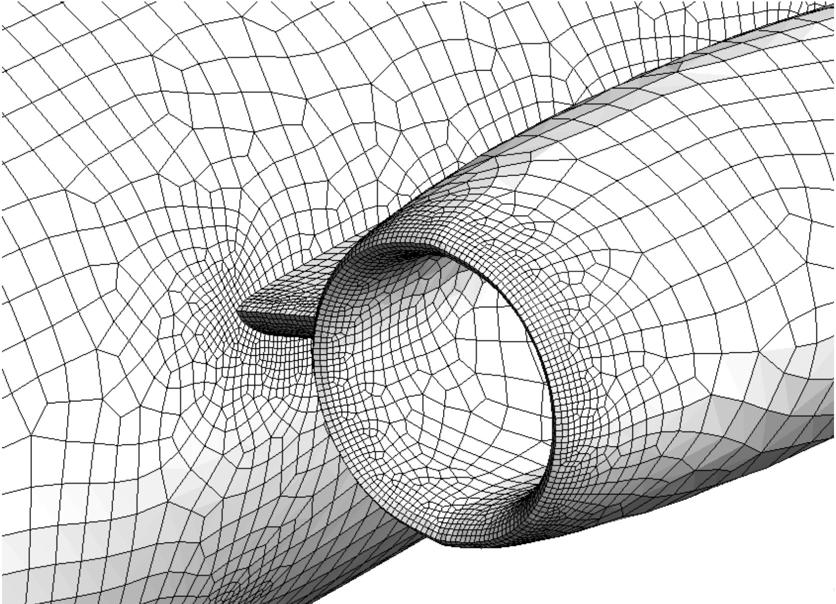
Falcon



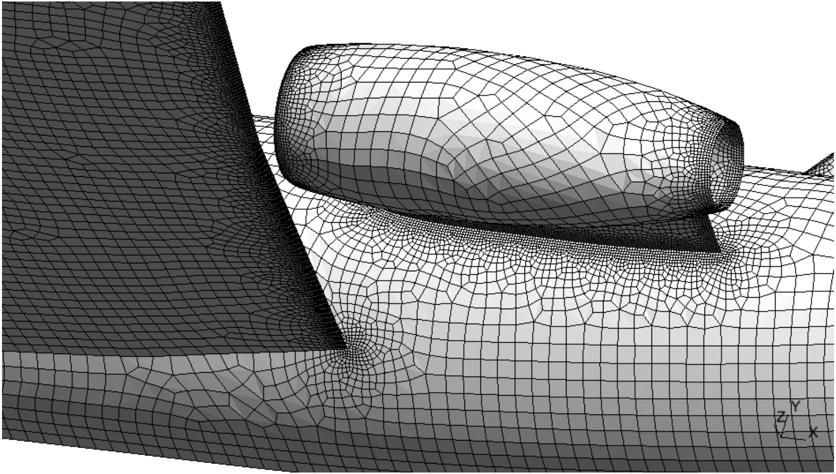
Falcon



Falcon

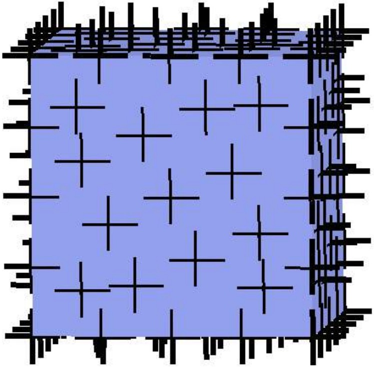


Falcon



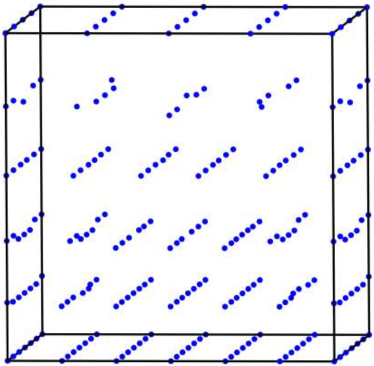
Going to 3D

- Start from a 2D triangular mesh that has been created using surfacic frame fields. Boundary mesh vertices are initially pushed into a queue.
- Each vertex popped out of the queue attempts to create six neighboring vertices.
- A tetrahedral mesh is created...
- and subsequently recombined.



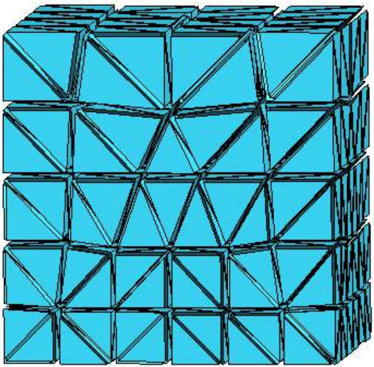
Going to 3D

- Start from a 2D triangular mesh that has been created using surfacic frame fields. Boundary mesh vertices are initially pushed into a queue.
- Each vertex popped out of the queue attempts to create six neighboring vertices.
- A tetrahedral mesh is created...
- and subsequently recombined.



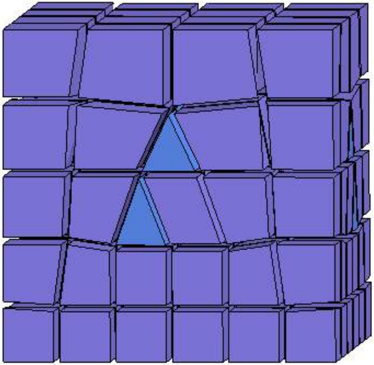
Going to 3D

- Start from a 2D triangular mesh that has been created using surfacic frame fields. Boundary mesh vertices are initially pushed into a queue.
- Each vertex popped out of the queue attempts to create six neighboring vertices.
- A tetrahedral mesh is created...
- and subsequently recombined.

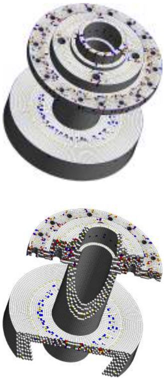
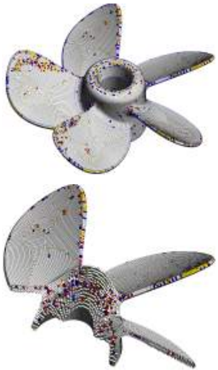
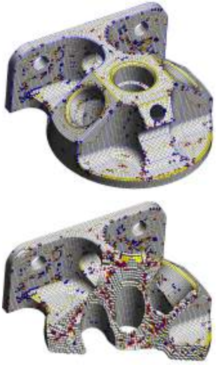


Going to 3D

- Start from a 2D triangular mesh that has been created using surfacic frame fields. Boundary mesh vertices are initially pushed into a queue.
- Each vertex popped out of the queue attempts to create six neighboring vertices.
- A tetrahedral mesh is created...
- and subsequently recombined.



Examples

CUBO	BLADES	CV745
		

Name	#Vertices	%Hex	%Prism	% Pyr	% Tet	cpu(s)
CUBO	133,436	89.74	4.02	4,20	2.02	247
BLADES	133,678	83.65	5.62	6.75	3.98	268
CV745	102,946	78.55	7.63	8.81	5.91	225

Conclusions

Work done:

- Indirect Hex-dominant mesh generation algorithm available
- Appropriate for FE simulations
- Extension to 3D

Ongoing work:

- Thin domains, anisotropy
- Non conforming Hexes
- All-Hexes

Wiki and doc

The automatic procedure is implemented within the open source code Gmsh: `http://www.geuz.org/gmsh`. Examples of how to use it can be found on the wiki: `https://geuz.org/trac/gmsh`. Access the wiki with username *gmsh* and password *gmsh*.

Thank you for your attention ...

jean-francois.remacle@uclouvain.be

Questions?