EFFICIENTLY HEX-MESHING THINGS WITH TOPOLOGY, IN PRACTICE

Jean-Christophe WEILL
CEA, DAM, DIF, F-91297 Arpajon, France

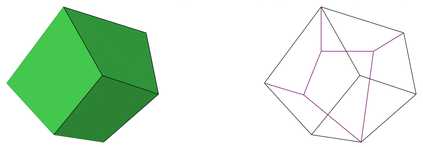
Context

In [1], Jeff ERICKSON describes a constructive method to extend a topological quadrilateral mesh of a connected surface Q in $\mathbb{R}^3$ to a topological hexahedral mesh of the interior domain Ω. Q should have an even number of quadrilaterals and no odd cycle in Q should bound a surface inside Ω.
If Q is given as a polyhedron in $\mathbb{R}^3$ with quadrilateral facets, a *topological* hexahedral mesh of the polyhedron can be constructed in polynomial times.

Our aim was to produce a C++ program using our mesh data structure GMDS[2] that implements this algorithm.

This poster describes some implementation points and difficulties of this algorithm.

This poster describes the algorithm as given in [1], step by step, with an illustration on the octagon spindle.



The octagon spindle is a polyhedron in $\mathbb{R}^3$ defined with 8 quadrilateral facets. Before this work, the best solution (that is minimizing the number of hexahedra) to this problem was given by Carlos CARBONERA and Jason SHEPHERD [3] and consists in a mesh of 38440 hexahedra.

Outline of the Algorithm

- Extend Q to a buffer layer B of hexahedra,
- Compute a triangulation T of the interior $\Omega\backslash\mathrm{B}$,
- Refine tetrahedra in T into cubes,
- Refine B to match inner and outer boundaries.

Construction of a temporary buffer hexahedral zone

The first step is a creation of a buffer layer B of topological hexahedra (cubes) that separates the boundary of Ω from its interior, obtained by joining the input quad mesh Q to a parallel copy of Q , we name it Q', just inside $\delta\Omega$.

This buffer zone is in the case of the octagon spindle constituted of 8 hexahedra.

Splitting Q' in triangles

The second step is also trivial and it consists in splitting each quadrilateral of Q^* into 2 triangles, this give an induced triangulation we name it ∂T.

This accepts many different solutions, at most 256 different triangulations can be derived for the octagon (if we do not consider the fact that the octagon has a lot of symmetries and rotations that keep it unchanged).

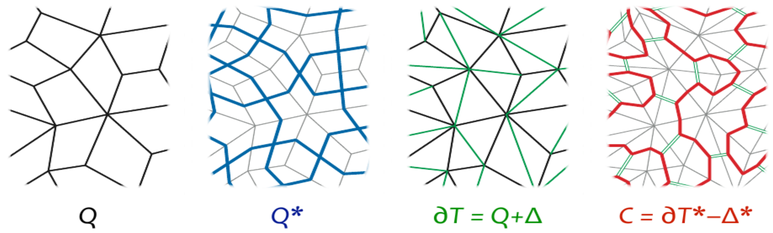
The triangulation is then used as a frontier of a tetrahedral mesh T (use your favorite algorithm here, personally I use Tetgen with parameter “Yq1.2pa0.1” as recommended by one colleague).

The best triangulation is then, one that gives a tetrahedral mesh with a minimum number of tetrahedra.

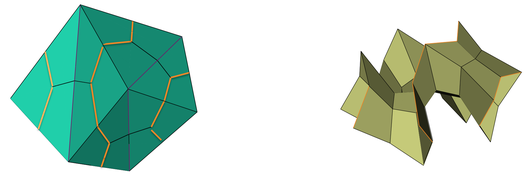
Refining the Interior Triangulation

For the octagon spindle, the minimum tetrahedral mesh consists in 9 tetrahedra and its boundary is given by the triangle mesh given below.

The next step consists in refining the interior triangulation by first splitting each tetrahedron in T into four hexahedra meeting at that tetrahedron’s centroid, in the usual manner depicted upper right, which gives for the octagon spindle a 36 hexahedral elements mesh.

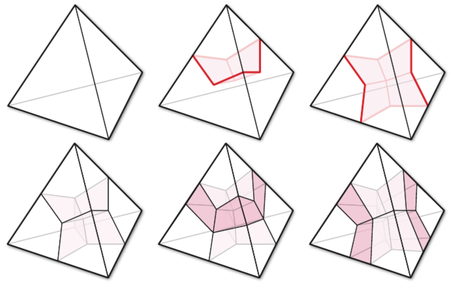


Find curves C in ∂T^* that are homologous to Q^*. *We then define* Σ *to be any 2-chain in* T^* such that $\partial\Sigma = C$. (such a 2-chain exists). Σ is an embedded surface in T^*.

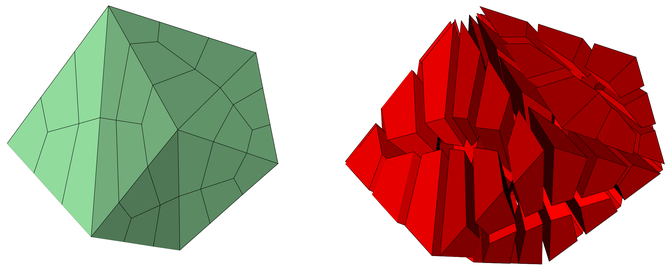


We can compute easily Σ by considering that 1) for each edge e in C, there is an odd number of facets from T^* in Σ that share the edge e and 2) that each other edge in Σ has an even number of facets incident to it. This can be obtained by solving a linear system in $\frac{\mathbb{Z}}{2\mathbb{Z}}$.

T is then refined into a hex mesh Y by splitting each tetrahedron into either, four seven or eight hexahedra, depending on whether the tetrahedron intersects zero, three of four facets of Σ as shown below.



Refining the Interior Triangulation

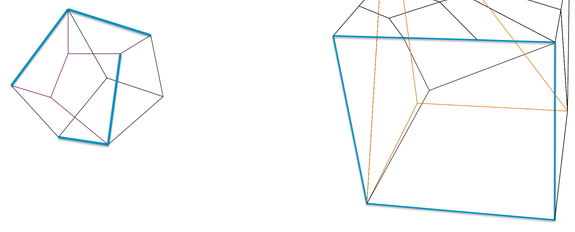


In the case of the spindle octagon, each tetrahedron is split in 8 hexahedra. So it gives the mesh just shown upper.

Refining the Buffer Cubes

It remains only to refine the buffer cubes in B to conform to the boundary of the hexahedral mesh Y. Each buffer cube has an *outer* facet in Q, an *inner* facet on the boundary of $\Omega\backslash\mathrm{B}$, and four *transition facets*. By construction, the inner facet of each buffer cube is subdivided into ten quadrilaterals.

Since Q^* contains a perfect matching[4], we compute it, and it gives us a partition of Q into different pairs of quads. In the left figure below, the facet that shared a blue edge are paired.



We then subdivide each transition facet of B into either 2 or 3 quadrilaterals, depending on whether that facet is bounded by an edge of R or not. Each boundary cube is refined into 20 quads. To finish the algorithm, it only remains to mesh into hexahedra the quadrilateral mesh shown in the right part in the figure upper.

In [1] as in [5], the problem of constructing the hexahedral solution to these cells of quadrilaterals is left open to the reader. Up to now, we did not achieve in solving it, and we showed that with the help of computers that it requires at least 12 hexahedra. Using the solution of [3], we constructed a solution with 76881 hexahedra.

So this method gives us a mesh of the octagon spindle with 615120 hexahedra. More generally if we can mesh the buffer cube with n hexahedra, this methods gives us a mesh with $72 + 8n$ hexahedra. So every mesh with less than 4804 hexahedra for the buffer cube should give a better solution than the one provided in [3]. Whether this topological solution is geometrically valid will be an other problem.

References

[1] Erickson, J. Efficiently Hex-Meshing Things with Topology. *Discrete & Computational Geometry 2014*; **52** (3):527-449.
[2] Ledoux F., Bertrand Y. and Weill J.-Ch. Generic Mesh Data Structure in HPC Context, *Computation Technologies and Innovation Series,* volume 26, Chapter 3, pp 49-80, Saxe-Coburg Publications, 2010
[3] Carbonera, C. and Shepherd J., A constructive approach to constrained hexahedral mesh generation, *Engineering with Computers 2010,* **26**:341-350
[4] Erickson, J., *Personal Communication,* 2016
[5] Eppstein, D. Linear complexity hexahedral mesh generation, in: 12th ACM symposium on computational geometry ACM, 1999 pp 58-67