Parallel mesh generation for large scale applications - Using hex-dominant meshes

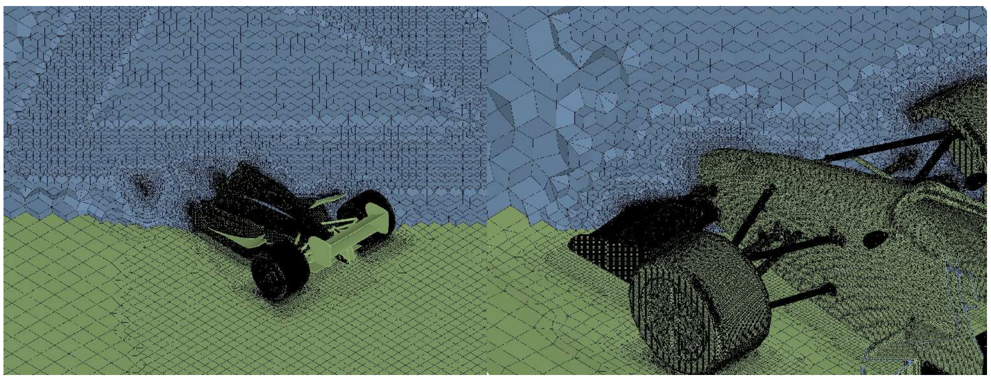
Dipl.- Ing. Ferdinand Kickinger
Ferdinand.Kickinger@meshing.at
CAE Software Solutions
Wolfkersbühelstr. 23, A-3730 Eggenburg

Abstract:

The industrial application of numerical simulation often reaches the limits, when complex CAD geometries are involved. These geometries are usually full of gaps, holes and other data errors. Furthermore they contain lots of unneeded details, like the screws, when the geometry is a whole car. In the following, a mesh generation algorithm, resolving all the mentioned problems, will be presented. This method also runs in parallel to take advantage of available memory located on different machines to attack problems of interest on the original geometry without simplifying it.

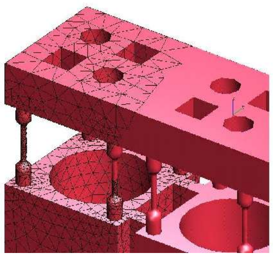
1 Introduction

In commercial applications, when dealing with really interesting geometries, one of the most time consuming parts it the CAD preparation (cleanup). The goal is, to achieve a "nice" simplified model for the simulation. In case the geometry is a car and the simulation should be about the cooling behaviour, the point of interests is the flow around the car and under the hood. Using standard meshing methods the CAD cleanup can last up to several weeks until we get a valid "clean" representation of the car from which we can start the meshing. To avoid this CAD cleanup we request a mesh generator, which is able to deal with everything.

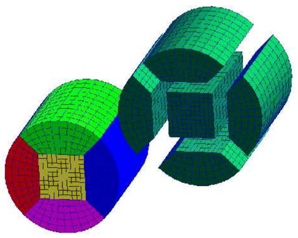


Picture 1, 2: Mesh of F1 racing-car

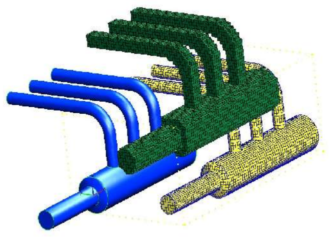
Standard commercial meshing tools always start from a “nice” geometry. The surface is then remeshed with triangles, and the interior is filled up with tetrahedrons created by advancing front, or Delauny –typed algorithms. The advantage of these algorithms is that they work fast and almost automatic when a “nice” surface description is available. The problem is that such algorithms are very sensitive on errors in the CAD data. For geometries like a whole car, often the geometry has to be rebuilt within the CAD to meet the demands of the mesh generator.


Picture 3: Advancing front surface mesh

Another method is block-structured meshes, often also called “hand-made” meshes. Here hexahedral blocks are merged together and projected on the geometry. The advantage is that almost every CAD problem can be overcome because all projections are done per hand and interactive. As one can imagine this consumes a lot of time and takes weeks to build up a mesh for sophisticated geometries.

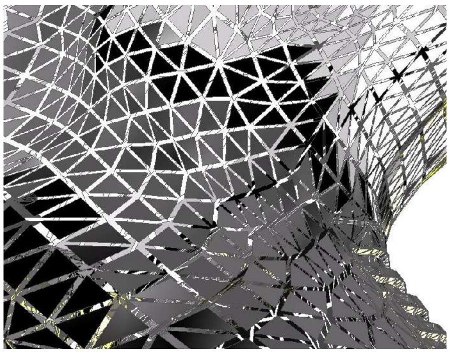

Picture 4: Block structured mesh

The next class of algorithms are “grid based methods”. Here we take a so called “background mesh” which overlaps the domain and “cut” off the cells which are outside the domain. Here we have a lot of freedom in choosing the background mesh and in cutting off the outer cells. The algorithm presented in these pages is of this type.


Picture 5: Geometry, background mesh, mesh

2 The Basic Algorithm

The geometry given is defined via a set of triangles. This triangles need not to be connected. Within the set there can be duplicate triangles, and the orientation is not of importance.

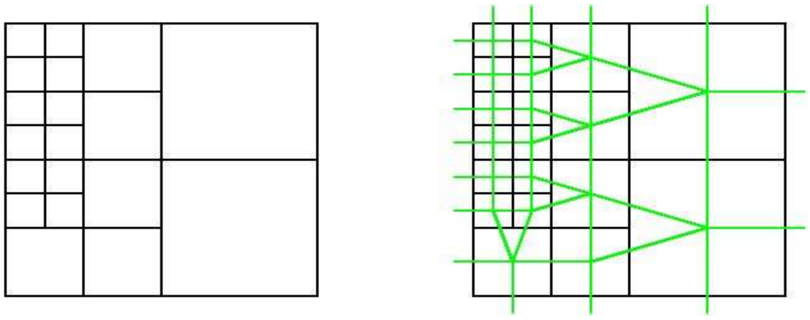


Picture 6: Possible data quality

Octree and Dual Mesh

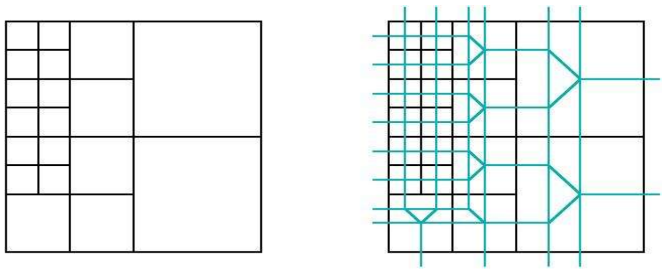
The basic data structure of the algorithm is an octree, which we need for a fast projection of a point near the geometry on the geometry. The octree is refined according to a priori information. This information is attached to subsets of the given triangles, or to additional geometries (bodies). This defines the local box size within the octree.

Now we have a local refined octree. This could be taken as hanging node background mesh (and some other methods do so), but we want to create a conform mesh. Therefore we take the dual mesh of the octree. This could simply be done by taking the box centres as vertices of the mesh, and all surrounding boxes of a corner in the octree define the cells.

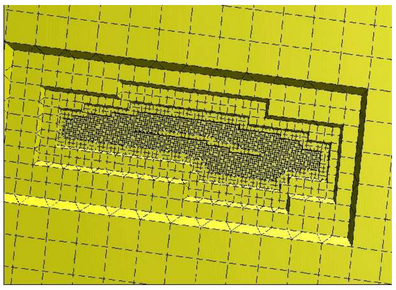


Picture 7: quad tree and quad tree with dual mesh

Picture 6 shows how this works in 2D. In 3D this leads us to non standard element types which would be no problem, but most available commercial simulation software can only deal with hexahedrons, prisms, tetrahedrons and pyramids. So the solution is here some kind of modified dual mesh.

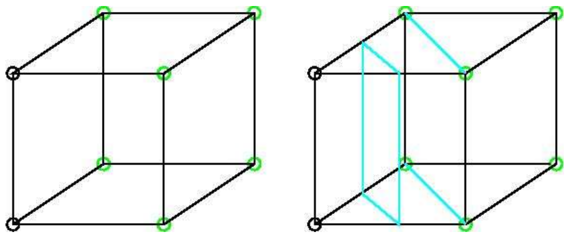

Picture 8: quad tree and quad tree with modified dual mesh.

This modification is simply by adding vertices when coarse boxes meet fine boxes. It can also be seen as inserting macro elements instead of the non standard cell types. A nice property of this method is that in 3D wee need only hexahedrons, pyramids and prisms (no tetrahedrons).

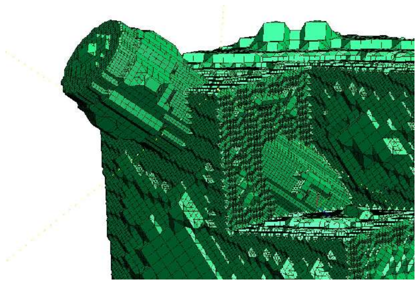

Picture 9: local refined modified dual mesh of an octree

Marking and Cutting

Now we are able to mark the different materials of the geometry. This is done by the definition of a starting point. Because of the construction with the dual mesh, the vertices of the mesh are the centres of the octree-boxes. If an octree-box contains triangles, we mark the according vertex in the mesh. This defines the border which is used for the marking of materials. The marking is done via the nodes of the mesh. This means, that we end up with all nodes marked as outer, material or boundary. Now we could just keep cells which fulfil some criteria and remove the rest (some methods do so), but we use here a splitting method. Instead of just removing cells, we cut the cells where nodes are marked as boundary and material. The cutting is done via bisecting an edge if the corners are marked different.


Picture 10: Example of cell-cutting

Taking care that we only use hexahedrons, prisms, pyramids and tetrahedrons and that we keep the whole mesh conform, we end up with an almost smooth mesh surface which is a good approximation of the real geometry already.


Picture 11: cutted mesh

Smoothing Surface and Mesh

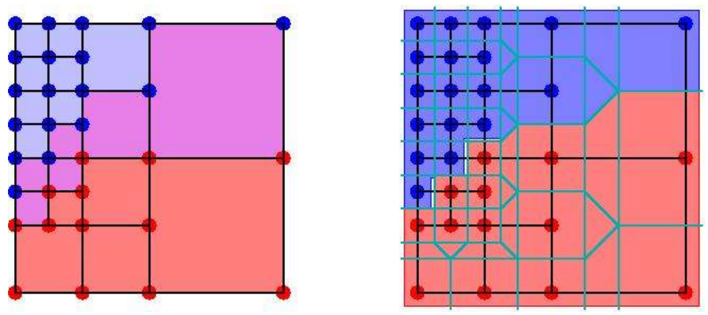
The only thing left to do is projecting and smoothing the surface of the mesh and finally smoothing the interior of the mesh. Both are done by minimizing a local energy functional. The surface is smoothed in combination with projection, so that iteration by iteration the surface "converges" to the real geometry. Important is, that the energy functional counts "shape" and "curvature" of the local environment.

Smoothing the volume mesh is the same, but we use Laplace typed smoothers as acceleration in advance. The energy functional is again based on the deformation energy known from elasticity.

To improve the mesh quality we also add one ore more boundary layers. This gives at least three cell layers everywhere and for CFD applications this is necessary for the k-e turbulence model.

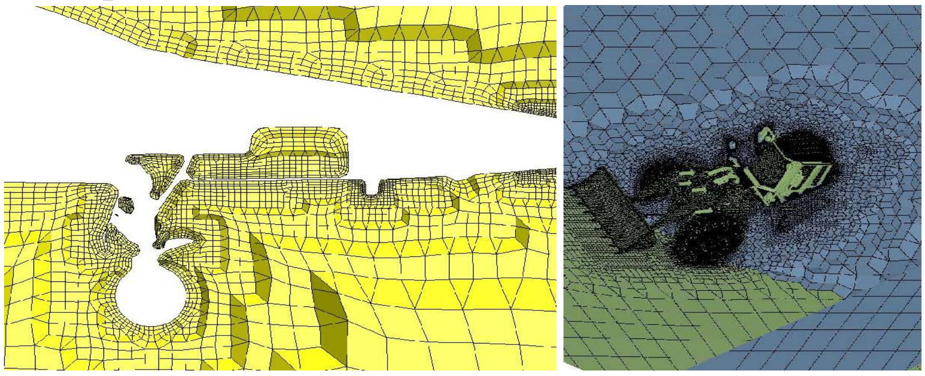
3 Parallel Aspects

The domain decomposition of the octree is done via one overlapping box layer. This corresponds to the dual mesh with distributed cells. This allows performing the refinement without communication.


Picture 12: Tree decomposition and according dual mesh decomposition

Also the dual mesh generation can be done without communication. The marking step of the nodes needs communication because it spreads out until nothings left. The algorithm is here, that we mark on each processor and treat the interfaces as boundary. Then in a communication step, we link the materials from the different processors together. The splitting can then again be done fully parallel. For smoothing of surface and volume mesh there is again communication needed, but this is like solving a PDE. So finally we end up with a distributed mesh on each processor.
One difficult point is missing. Of course there is dynamic load balancing needed, because we cannot predict the refinement behavior, especially when the refinement is based on criteria like surface curvature. The plan is to use the hierarchy of the octree for this dynamic load distribution.

4 Examples


Picture 13: Examples

Formula 1 Car	46 000 000 cells	6.0h cpu (4procs)
Airbus Interior	33 000 000 cells	4.5h cpu (4procs)
Loader	4 700 000 cells	.7h cpu (2procs)

5 Conclusion

Finally we end up with a method, which allows handling any type of input geometry creating any size of meshes. This speeds up the time from CAD to mesh dramatically. The method produces meshes consisting of hexahedrons (60%), prisms (20%), pyramids (10%) and tetrahedrons (10%). The algorithm is also able to suppress features if the local mesh resolution is too small to resolve them.

6 Bibliography

[1] Kickinger, F.: Multiscale Problems: Meshes and Solvers. ZAMM vol 78, (1997).

[2] Tatschl, R; Riedinger, H;, Künsberg Sarre, C; Putz, N; Kickinger ,F .:Rapid Meshing and Advanced Physical Modeling for Gasoline DI Engine Applications. SAE Congress Detroit, (2000).