DOMAIN DECOMPOSITION APPROACH FOR AUTOMATIC PARALLEL GENERATION OF 3D UNSTRUCTURED GRIDS

Evgeny G. Ivanov*, Heiko Andrä†, Alexey N. Kudryavtsev‡

*Fraunhofer ITWM, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
e-mail: ivanov@itwm.fhg.de

†Fraunhofer ITWM, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
e-mail: andrae@itwm.fhg.de

‡Institute of Theoretical and Applied Mechanics, Siberian Division of RAS, Institutskaya str. 4/1, 630090, Novosibirsk, Russia
e-mail: alex@itam.nsc.ru
Web page: http://www.itam.nsc.ru/users/alex/

Key words: Grid Generation, Unstructured Grid, Delaunay Triangulation, Parallel Programming, Domain Decomposition, Load Balancing

Abstract. The desire to simulate more and more geometrical and physical features of technical structures and the availability of parallel computers and parallel numerical solvers which can exploit the power of these machines have lead to a steady increase in the number of grid elements used. Memory requirements and computational time are too large for usual serial PCs. An a priori partitioning algorithm for the parallel generation of 3D nonoverlapping compatible unstructured meshes based on a CAD surface description is presented in this paper. Emphasis is given to practical issues and implementation rather than to theoretical complexity. To achieve robustness of the algorithm with respect to the geometrical shape of the structure authors propose to have several or many but relatively simple algorithmic steps. The geometrical domain decomposition approach has been applied. It allows us to use classic 2D and 3D high-quality Delaunay mesh generators for independent and simultaneous volume meshing. Different aspects of load balancing methods are also explored in the paper. The MPI library and SPMD model are used for parallel grid generator implementation. Several 3D examples are shown.

1 INTRODUCTION

Unstructured mesh techniques take an important place in grid generation. The main feature of unstructured grids consists, in contrast to structured grids, in almost complete absence of
restrictions on grid cells, grid organization, or grid structure. It allows placing the grid nodes locally irrespective of any coordinate direction, so that complex geometries with curved boundaries can be meshed easily and local regions in which variations of the solution are large or the accurate solution is of interest can be resolved with a selective insertion of new points without unduly affecting the resolution in other parts of the physical domain. Local adaptive mesh refinement can be easily done.

Unstructured grid methods were originally developed in solid mechanics. Nowadays these methods influence many other fields of applications beyond solid modeling, in particular computational fluid dynamics where they are becoming widely adopted.

At the present time the methods of unstructured grid generation have reached the stage where three-dimensional domains with complex geometry can be automatically meshed. The most spectacular theoretical and practical achievements with respect to automation have been connected with the techniques for generating tetrahedral grids. There are at least two basic approaches that have been used to generate these computational meshes: The Delaunay [1] and advancing front [2]. In this paper we are dealing with Delaunay approaches only. Delaunay property means that the hypersphere of each n-dimensional simplex defined by n+1 points is void of any other points of the triangulation (Fig. 1). This empty circum-circle property is the reason why the grid cells of a Delaunay triangulation are without small or large angles [3].

The most well-known or widely-used Delaunay triangulation algorithms are the “Divide & Conquer” algorithm [4] and the incremental insertion algorithm [5].

A CAD object description is a set of points, curves, surfaces and solids that model the object. There are many different standards. Two well-known ones are IGES, which is popular in the US, and STEP, created by the International Standard Organization. We use a triangular surface mesh as an approximation. Standard formats for surface triangulations are STL (stereolithography format) and OFF (object file format). In Fig. 2 two triangles approximating curved surface are written in these formats. The STL format is

![Figure 1: left – triangles which satisfy Delaunay circum-circle property; middle- triangles which do not satisfy Delaunay property; right - Delaunay triangulation, all triangles are Delaunay](image1)

![Figure 2: Two well-known surface triangulation formats: STL (stereolithography format) in the left column and OFF (object file) in the right column](image2)
shown on the left side of Fig. 2. It specifies triangular surfaces with normals. The OFF format is shown on the right side and specifies vertices coordinates and their incidents.

The introduction of scalable parallel computers is enabling ever-larger problems to be solved in such areas as Computational Mechanics (CM), Computational Fluid Dynamics (CFD) and Computational Electro Magnetics (CEM). Grids in excess of $10^7$ elements have become common for production runs in CFD [6-10] and CEM [11,12]. The expectation is that in the near future grids in excess of $10^8 – 10^9$ elements will be required [13]. As mesh cell numbers become as large as this (Fig. 3), the process of mesh generation on a serial computer becomes problematic both in terms of time and memory requirements. For applications where remeshing is an integral part of simulations, e.g. problems with moving bodies [14-20] or changing topologies [21,22], the time required for mesh regeneration can easily consume more than 50% of the total time required to solve the problem [13]. Faced with that problem, a number of efforts have been reported on parallel grid generation [13,23-44].

Starting in two dimensions, Verhoeven et al. [44] demonstrated the ability to produce parallel unstructured Delaunay meshes across a network of workstations. Topping et al. [45], Laemer et al. [46], Loehner et al. [23], amongst others, have parallelized the advancing front algorithm. Moving to three dimensions, the task becomes more complicated. Chew et al. [28], Chrisochoides et al. [47], Okunsanya et al. [29] have parallelized the Delaunay algorithm. Loehner [25] has demonstrated the extension of the advancing front algorithm to produce tetrahedral elements on parallel platforms. Said et al. [30] have shown parallel mesh generation using initial coarse meshing and decomposition.

There are two methods to parallelize a mesh generator; parallelize the algorithm directly or decompose the problem. The latter is based on domain decomposition and can be classified into a priori and a posteriori partitioning algorithms [48].

Cignoni et al. [37,38], for instance, investigated algorithms for the parallelization of Delaunay triangulation. Different solutions were designed and evaluated. The first one, which is a parallel implementation of the “Divide & Conquer” paradigm, was faster but showed limited scalability. The second one performs a regular geometric partition of the dataset and subdivides the load among $m$ independent asynchronous processors, using on each node an incremental construction algorithm (InCoDe); this solution is algorithmically quite simple and
allows sufficiently good scalability. It is used for computer graphics applications.

Chetverushkin et al. [39] suggested another algorithm based on initial a posteriori partitioning, where the volumetric mesh generation procedure includes three main stages. The first one consists of surface meshing using the initial geometric model. The constructed surface mesh forms a base for the subsequent domain splitting into a set of large tetrahedrons. This is a stage of primary volumetric triangulation and the number of these tetrahedrons usually is moderate. At the third stage of the process the mesh of primary tetrahedrons is refined to the necessary resolution and the resulting 3D mesh is smoothed and optimized if necessary.

Recently, Ivanov [31,43] introduced and developed an a priori partitioning algorithm for the parallel generation of three-dimensional unstructured grids using the domain decomposition approach, i.e. decomposing the problem. This paper provides an extensive description of the new algorithm stressing practical issues and implementation. To achieve robustness of the algorithm the authors prefer to have several or many relatively simple algorithmic steps rather than complex algorithms such as mentioned above.

The paper is organized as follows: Section 2 formulates the problem, gives an extensive description of the algorithm and explores load balancing and partitioning methods. In Section 3, the results are discussed and a complex geometry example is shown. Section 4 summarizes and concludes the paper.

2 PARALLEL GENERATION ALGORITHM

The goal of the work is to create a parallel grid generator for high-quality unstructured volume tetrahedral grids with good properties for solving PDEs (e.g. Delaunay property). It should be fully automatic, adaptive (via coupling with the solver) and, of course, be able to generate large meshes. The input data is a CAD surface description of an object. In fig. 4 the main steps of implementation are shown.

![Figure 4: Scheme of major implementation steps of the parallel grid generator.](image)

The domain decomposition approach is used for a parallel grid generation. The algorithm consists of several major steps:

1. Load balanced recursive decomposition of an object into open subdomains.
   a. Center of mass and inertia matrix calculation for setting up the cutting plane.
   b. Smoothing of a cross-section contour for projection.
   c. Projection of the contour nodes to the plane for further interface triangulation.
2. Construction of 2D constrained Delaunay triangulation on the projected interface.
3. Mapping back the contour nodes of the triangulation to original surface positions.
4. Construction of closed and compatible surface mesh for each subdomain.
5. Independent parallel volume meshing (without communication) within each subdomain based on its surface mesh description.

In Fig. 5 the major steps of the algorithm are shown. The smallest principal inertia axis is chosen as splitting criterion in order to achieve good load-balancing and minimize cross-section interface area. The goal of the parallel generation algorithm is to perform simultaneous construction of three-dimensional grid in each subdomain. This is computationally most expensive step of the algorithm.

The advantage of this algorithm is that it allows us to use sequential classic 2D and 3D Delaunay triangulators, which are capable of producing high-quality Delaunay meshes with different conditions and constraints. They are widely available [49]. The programs Triangle from Shewchuk [50] and TetGen from Hang Si [51] have been used for 2D and 3D triangulation.

The disadvantage of this method is that domain decomposition is not always effective and depends on the shape of an object. Therefore it needs to be continuously improved and

Figure 6: Flow chart of the parallel grid generator
extended for different shapes (non-convex, long, and thin).

The detailed flow chart of the parallel grid generator is shown in Fig. 6.
Details of the algorithm are given in the following paragraphs.

2.1 Setting the cutting planes up and load balancing

Load balancing always has been a big issue for parallel applications. Several techniques and criteria are considered in the paper.

**Prepartition along the same direction.** The object is partitioned along several parallel partitioning planes. A partitioning of an object into N subdomains would require N-1 parallel tasks. The partitioning can be done in parallel.

**Recursive prepartitioning.** An object is cut in two. Then for each part a new cutting plane is determined and the parts are cut in two and so on. Note, that first task is sequential, second task involves two parallel tasks, and the k th step involves $2^{k-1}$ tasks.

**Overdecomposition.** An object is decomposed into many subdomains. The number of subdomains is much larger than the number of processors. Then in the case of load imbalance the master process gives the task (subdomain) to idle processor.

The partitioning criteria for the object decomposition could be volume, number of boundary facets, number of nodes, moment of inertia.

Going through developing stages the parallel grid generator had different partitioning techniques and splitting criteria (see Fig. 7). Axis-aligned equidistant planes can result in

---

**Figure 7:** Evolution of the parallel grid generator. 1 – equidistant axis-aligned cutting. 2 – axis-aligned cutting with volume comparison. 3 – recursive cutting with moment of inertia equality.
imbalanced partitioning depending on the shape of the object because the scheme it is not sensitive to the object shape and can not be applied to arbitrary geometries. The axis-aligned cutting with volume comparison criterion produces balanced partitioning, but it also can result in a large interface area, which is not optimal for further interface triangulation and crucial for a parallel solver, because of communication overhead between subdomains.

Here the center of gravity along with moment of inertia criterion is used. Each object is cut perpendicular to its smallest principal inertia axis. It means that for each part with the set of nodes \( V \), the inertia matrix is computed by eq. (1), where \( (x_G, y_G, z_G) \) is the coordinate of the center of gravity which is calculated by assigning unit mass to each node of the mesh. Thus grid resolution is also taken into account. Then (one of) the eigenvector(s) with the smallest eigenvalue is selected.

\[
\begin{align*}
I &= \sum_p \left( (y_G - y_p)^2 + (z_G - z_p)^2 \right) - \sum_p (x_G - x_p)(y_G - y_p) - \sum_p (x_G - x_p)(z_G - z_p) \\
&\quad - \sum_p (x_G - x_p)(y_G - y_p) + \sum_p ((x_G - x_p)^2 + (z_G - z_p)^2) - \sum_p (y_G - y_p)(z_G - z_p) \\
&\quad - \sum_p (x_G - x_p)(z_G - z_p) + \sum_p ((x_G - x_p)^2 + (y_G - y_p)^2)
\end{align*}
\]

This procedure defines planes perpendicular to the smallest principal inertia axis. The actual cutting plane is chosen to go through the center of gravity. This partitioning technique is sensitive to the object shape and grid resolution and can minimize the interface area.

Nevertheless it turns out to be hard to find a reasonable criterion for predicting a good load balancing in advance. Even if the number of tetrahedra is approximately the same for each subdomain, the CPU time spent for the volume meshing of each part can be quite different [32].

### 2.2 Construction of the splitting contour

Once the cutting plane is defined, we can construct a cross-section contour where 2D constrained Delaunay triangulation will be performed.

It is very important to partition a mesh in such a way that it does not deteriorate significantly the grid quality. In [43] we employed the following division method: The intersected triangle was divided into three other smaller triangles. Additional nodes were inserted, so that the final contour consisted of the intersection lines of the plane with boundary facets (Fig. 7). Obviously, this splitting can cause “bad triangles” – triangles with very acute angle or small area. Proposed in [52] sophisticated mesh optimization technique was used in order to overcome this problem and improve the quality of the mesh.

Here we present more advanced technique. The construction of the contour consists of the following steps:

1. Extract all intersected edges of the surface triangulation.
2. Remove all edges with “hanging” node.
3. Sort the edges into a closed loop.
4. Smooth the contour.

![Figure 8: Steps of the contour construction. a – extract all intersected edges. b – remove edges with “hanging” nodes. c – replace two edges in the loop with third one for triangles which have three nodes in the path. d – smoothed contour of edges.](image)

The smoothing step requires more detailed explanation. We consider all triangles attached to the path of edges and for those triangles, which have two edges in the path we replace them with third edge. The smoothing phase is required for better further projection on the cutting plane and construction of 2D triangulation of interface. Next section is devoted to that problem.

2.3 Construction of interface and 2D constrained Delaunay triangulation

Two steps are required before the 2D triangulation of the interface can be done:

1. Projection of the contour nodes on the cutting plane.

2. Rotation into X-Y plane of the coordinates for 2D triangulation.

The program Triangle by Shewchuk [50] is used for triangulation of the interface.

In three dimensions a coordinate rotation can be described by a 3x3 matrix $M$, which rotates a coordinate by an angle $\theta$ around a unit vector $v$,

$$
M (v, \theta) = \begin{bmatrix}
\cos \theta + (1 - \cos \theta) x^2 & (1 - \cos \theta) y x + (\sin \theta) z & (1 - \cos \theta) z x + (\sin \theta) y \\
(1 - \cos \theta) y x + (\sin \theta) z & \cos \theta + (1 - \cos \theta) y^2 & (1 - \cos \theta) y z - (\sin \theta) x \\
(1 - \cos \theta) z x + (\sin \theta) y & (1 - \cos \theta) z y + (\sin \theta) x & \cos \theta + (1 - \cos \theta) z^2
\end{bmatrix}.
$$

After triangulation of the interface with certain constraints on minimal angle and maximum triangle area, the coordinates are reversed back and the contour nodes are mapped back on their original surface positions (fig. 9).
2.4 Splitting the mesh along the path of edges

It is not that obvious how to split the mesh along the path of edges. For the triangles which are not intersected it is clear. One just has to check whether this triangle on the right or on the left side of the cutting plane. Another situation is with intersected triangles. Here we should take into account the smoothness of the contour.

Let us recall, that after smoothing phase, there are no triangles which have two edges in the path, since they were replaced with the third edge. So the intersected triangles have two vertices in the edge path and one free vertex on a left or right side of the path. Hence, triangle belongs to that part, where this free vertex is located. Fig. 10 explains the implemented technique.

2.5 Parallel volume mesh construction

When balanced partitioning is done and a closed and compatible surface mesh is constructed for each subdomain the volume meshes are constructed in parallel. TetGen - a
quality tetrahedral mesh generator and three-dimensional Delaunay triangulator from Hang Si [51] has been used for volume Delaunay tetrahedralization with certain quality bound (radius-edge ratio), a maximum volume bound, a maximum area bound on a facet, a maximum edge length on a segment. Fig. 11 shows an example of partitioning and final tetrahedralization inside of the subdomains on 4 CPUs.

3 RESULTS AND DISCUSSION

The generation time of the volume mesh for the whole computational domain is that time, which is spent on generation of a volume mesh in one subdomain with the highest computational effort. The flip-based algorithm TetGen uses is from Edelsbrunner and Shah [53]. The complexity of the algorithm is $O(n^2)$ in worst case. In practice this algorithm has a nearly linear complexity $O(n \log n)$. In Fig. 12 a speed-up graph is shown for the mesh of Fig. 11. The speed-up here is better than linear. It is “super-linear” due to the fact that complexity of the algorithm is higher than $O(n)$. When we divide our problem into subproblems and solve them in parallel we get, of course, a super-linear scaling. Note, that there is no any communication overhead, since construction of the volume mesh is performed absolutely independent and does not require any data exchange.

![Figure 12: Speed-up of volume mesh generation time (shown in red) without prepartitioning time](image)

3.1 Numerical test example

In Fig. 13 numerical example of knee prosthesis component is shown [55]. It was partitioned by using developed partitioning algorithm for 4 CPUs (see Fig. 14).
3.2 Surface and volume mesh quality

It was already mentioned that the mesh quality is an important property to pay attention to. The partitioning algorithm should not adversely affect the quality of the surface mesh. The most undesirable triangles for FEM calculations are those with very acute angles. Figure 15 shows how our partitioning algorithm affects mesh quality. “Bad triangles” (shown in red) with angle less than 30° are shown before and after partitioning.

For the volume tetrahedral mesh there are several measures available in the literature. Here the quality measure used in TetGen will be described [54]. For high
accuracy in the FEM, it is generally necessary that the shapes of tetrahedra have bounded aspect ratio. The aspect ratio of a tetrahedron is the ratio of the maximum side length to the minimum height. For a quality mesh, this value should be as small as possible. For example “thin and flat” tetrahedra tend to have a large aspect ratio.

A similar but weaker quality measure is radius-edge ratio. The radius-edge ratio $Q$ is the ratio of the circumsphere radius $R$ to the length of the shortest edge $L$, defined by

$$Q = \frac{R}{L} \quad (3)$$

For all well-shaped tetrahedra, this value is small, while for most of badly-shaped tetrahedra, this value is large [56] (see Fig. 16).

A special type of badly-shaped tetrahedron is called “sliver”, which is very flat and nearly degenerate. Slivers can have radius-edge ratio as small as $\sqrt{2}/2 \approx 0.707$. The radius-edge ratio is not a proper measure for slivers. TetGen does a simplified sliver removal step. Slivers are removed by local flip operations and peeling off from the boundary.

After tetrahedralization automatic mesh quality evaluation is performed and a mesh quality report on the smallest and largest volume, the shortest and longest edge, the smallest and largest dihedral angle, radius-edge ratio histogram, aspect ratio histogram, dihedral angle histogram is printed.
Figure 16: The radius-edge ratio for some well-shaped and badly-shaped tetrahedra. a – the radius-edge ratio for some well-shaped tetrahedra. b – the radius-edge ratios for some badly-shaped tetrahedra. c – sliver (special type of badly-shaped tetrahedron)

4 SUMMARY AND CONCLUSIONS

A method to generate 3D unstructured nonoverlapping meshes in parallel has been demonstrated. An a priori algorithm based on domain decomposition has been used. This problem allows us to use standard sequential 2D and 3D triangulators in parallel. The programs Triangle [50] and TetGen [51] were employed for construction of 2D and 3D high-quality Delaunay meshes. Different aspects of load balancing methods and criteria such as prepartitioning along the same direction and recursive partitioning based on moment of inertia splitting criterion are also explored in the paper. The partitioning algorithm is demonstrated for a knee prosthesis component with emphasis on mesh quality. The parallelization strategy is based on SPMD computational model and employs the MPI library for the implementation of the parallel grid generator. Hence, the most expensive computations, the generation of the volume mesh inside each subdomain is completely independent and performed in parallel. A superlinear speed-up of volume mesh construction time has been observed. This fact is due to complexity of Delaunay mesh construction which is higher than $O(n)$.

The parallel grid generator has two important benefits over traditional Delaunay generators, the time required to generate a mesh is shorter than that from a sequential generator, and the memory required for each CPU to generate a mesh is lower in comparison to that of a sequential mesh generator. This enables us to work with larger meshes than it would be possible with sequential generators.

An a priori partitioning algorithm is clearly advantageous in terms of computational time and memory usage compare to an a posteriori method used by mesh partitioning libraries such as METIS [57], since we perform first partitioning and then volume meshing, while an a posteriori method partition already constructed volume mesh.
Further work on testing, creating of programming interface with CAD, handling of examples with more complex geometries and larger meshes and performing local adaptive mesh refinement is in progress.

5 ACKNOWLEDGEMENTS

This work is supported by the DAAD, TU Kaiserslautern, Fraunhofer Institute for Industrial Mathematics and Institute of Theoretical and Applied Mechanics SD RAS. The authors are grateful to Dimitar Stoyanov and Robert Zillich for their contribution to that work and corrections of the paper. The authors wish to thank Academician S.K. Godunov for his interest to this work and many comments on subject.

REFERENCES


triangulator," *Proceedings first workshop on Applied Computational Geometry*, 124-133
(1996)

[51] H. Si, K. Gaertner, “Meshing Piecewise Linear Complexes by Constrained Delaunay
Tetrahedralizations”, *Proceeding of the 14th International Meshing Roundtable*, (2005)


Triangulations”, *Algorithmica* 15, 223-241 (1996)

the Fourteenth Annual Symposium on Computational Geometry*, 86-95 (1998)

[55] [http://www.lima.it/english/prostheses/products/multigen.html](http://www.lima.it/english/prostheses/products/multigen.html)
