AUTOMATIC PARALLEL GENERATION OF THREE-DIMENSIONAL UNSTRUCTURED GRIDS FOR COMPUTATIONAL MECHANICS

E. G. Ivanov
Competence Center for High Performance Computing and Visualization, Fraunhofer Institute for Industrial Mathematics, Kaiserslautern, Germany
e-mail: ivanov@itwm.fraunhofer.de

Представлен алгоритм параллельной генерации трехмерных неструктурированных сеток из поверхностного CAD описания. Акцент сделан на практических проблемах и реализации. Использован метод разложения расчетной области на подобласти. Параллельный генератор сеток был соединен с программой, решающей уравнения механики.

Introduction

Unstructured mesh techniques take an important place in grid generation. The main feature of unstructured grids consists, in contrast to structured grids, in a nearly absolute absence of any 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 solution variations are large can be resolved with a selective insertion of new points without unduly affecting the resolution in other parts of the physical domain.

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 widespread.

At the present time the methods of unstructured grid generation have reached the stage where three-dimensional domains with complex geometry can be successfully meshed. The most spectacular theoretical and practical achievements have been connected with the techniques for generating tetrahedral grids. There are at least two basic approaches that have been used to generate these meshes: 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 gives us some grounds to expect that the grid cells of a Delaunay triangulation are not too deformed [3].

The most famous Delaunay triangulation algorithms are the “Divide & Conquer” paradigm algorithm [4] and the incremental insertion algorithm [5].

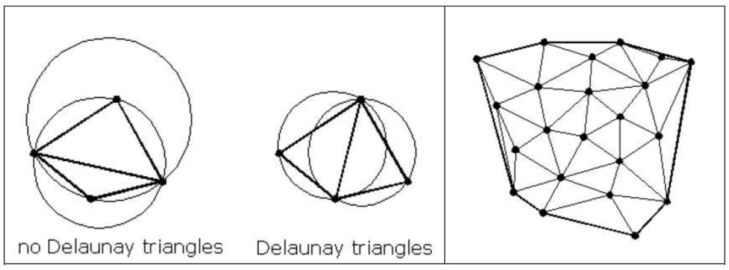
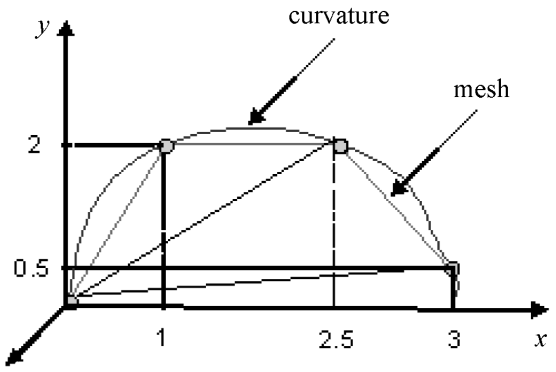


Fig. 1. Delaunay property and Delaunay triangulation.

A CAD object description is a set of points, curves, surfaces and solids that model the product. There are many different standards. Two of them are the most famous: IGES, which is popular in the US, and STEP, created by the International Standard Organization. We use a triangular mesh as an approximation, so we come to another format which represents surface triangulation: 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 shown on the left side of the picture. 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



```
facet normal 0.00 0.00 1.00
   outer loop
     vertex 0.00 0.00 0.00
     vertex 2.50 2.00 0.00
     vertex 1.00 2.00 0.00
   endloop
 endfacet
facet normal 0.00 0.00 1.00
   outer loop
     vertex 0.00 0.00 0.00
     vertex 3.00 0.50 0.00
     vertex 2.50 2.00 0.00
   endloop
 endfacet
```

```
4 2 0
0.00 0.00 0.00
3.00 0.50 0.00
2.50 2.00 0.00
1.00 2.00 0.00
1 2 3
3 4 1
```

Fig. 2. STL and OFF triangulation formats.

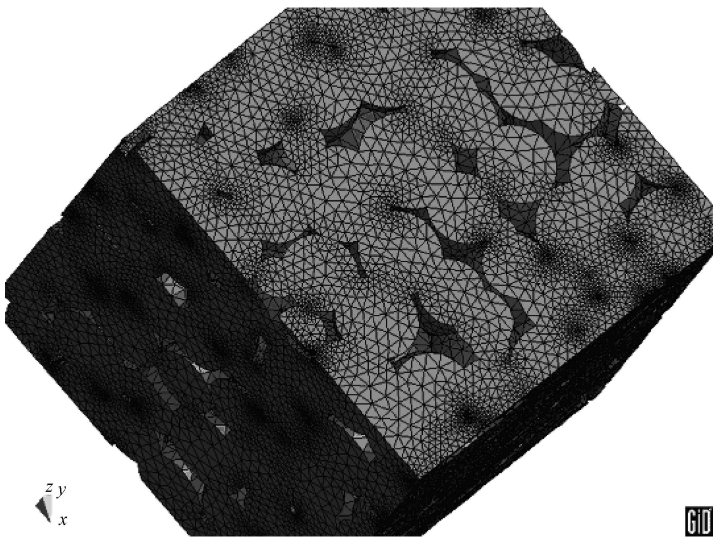


Fig. 3. A foam microstructure of sinter material.

grids in excess of 10^8–10^9 elements will be required [13]. As mesh sizes becomes 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, 23–42].

For instance, Cignoni [37, 38] 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 operates 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 was used for computer graphics applications.

Recently, the author [31] introduced an algorithm for parallel generation of three-dimensional unstructured grids using domain decomposition approach. This paper provides an extensive description of the algorithm stressing practical issues and implementation. The author prefers to have several or many relatively simple algorithmic steps rather than complex algorithms such as mentioned above.

The paper is organized as follows: in section 1 the problem is formulated, the section 2 gives an extensive description of the algorithm and explores load balancing and mesh optimization, in section 3 the parallelization strategy and its implementation are discussed and analyzed, the section 4 is devoted to coupling the parallel grid generator with the parallel FEM solver, and section 5 summarizes and concludes the paper.

1. Problem formulation

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

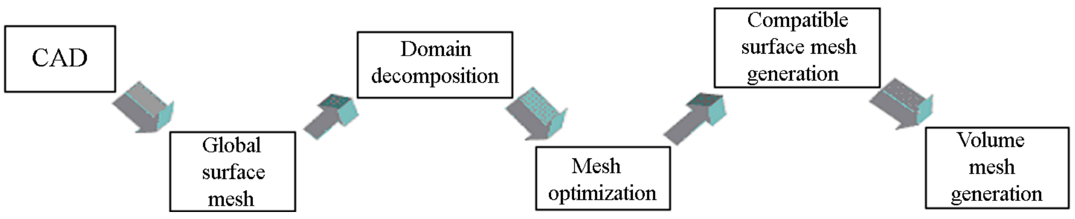


Fig. 4. Scheme of implementation steps.

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. We have a CAD boundary representation of the object which we approximate with triangular mesh. So we have a global surface mesh and we want to use the domain decomposition approach. After dividing the domain into subdomains it is necessary to perform a mesh optimization. Then we construct a compatible surface mesh for each subdomain and, if good load-balance is achieved, perform independent and parallel volume mesh generation within each subdomain.

2. Parallel generation algorithm

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

1. Load balanced decomposition of an object into open subdomains by using a set of cutting planes.

2. Construction of 2D constrained Delaunay triangulation on each interface and of closed and compatible surface mesh for each subdomain.

3. Performing surface mesh optimization.

4. Independent and parallel volume meshing within each subdomain based on its surface mesh.

In fig. 5 the major steps of the algorithm are shown in two dimensions. Equality of volumes is chosen as splitting criteria in order to achieve good load-balancing. The goal of the parallel generation algorithm is to perform simultaneous construction of three-dimensional grid in each subdomain. This is the heaviest computations with running time $O(N^3)$, where N is a number of mesh nodes.

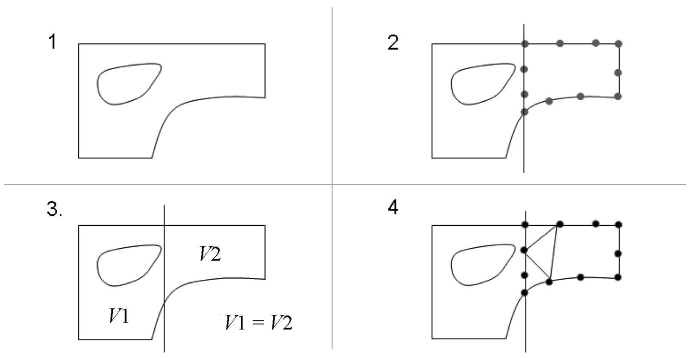


Fig. 5. Major steps of the algorithm. Two dimensions.

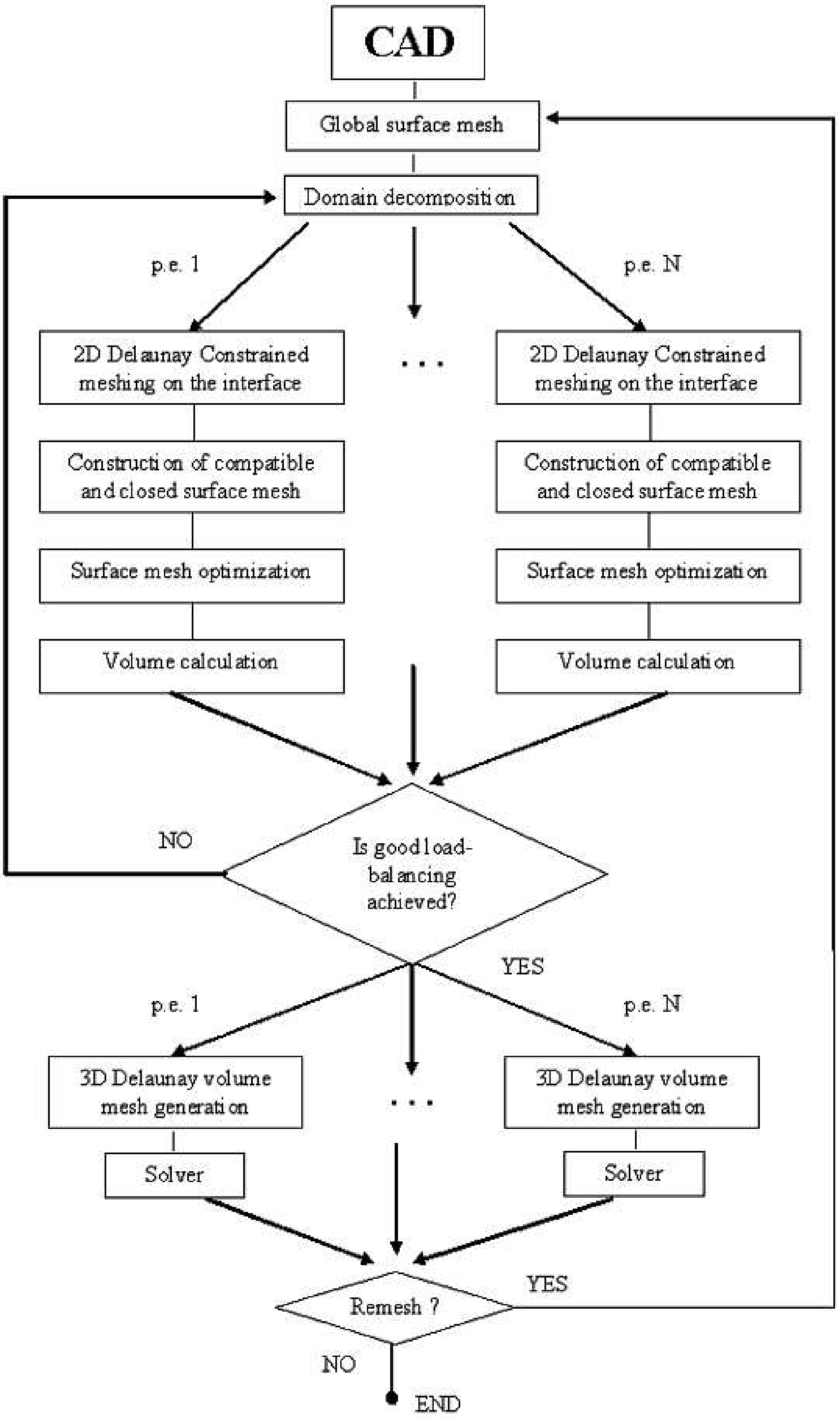

CAD
Global surface mesh
Domain decomposition
p.e. 1
p.e. N
2D Delaunay Constrained meshing on the interface
Construction of compatible and closed surface mesh
Surface mesh optimization
Volume calculation
2D Delaunay Constrained meshing on the interface
Construction of compatible and closed surface mesh
Surface mesh optimization
Volume calculation
NO
Is good load-balancing achieved?
YES
p.e. 1
p.e. N
3D Delaunay volume mesh generation
Solver
3D Delaunay volume mesh generation
Solver
Remesh ?
YES
NO
END

Fig. 6. Flow chart of the parallel grid generator.

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 [43].

The programs TRIANGLE from Shewchuk [44] and TetMesh-GHS3D [45] from Simulog and INRIA have been used for 2D and 3D triangulation.

The disadvantage of this method is that domain decomposition is not always efficient and therefore needs to be continuously improved.

The flow chart of the parallel grid generator is shown in fig. 6.

Having described the general procedure, details will be given in the following paragraphs.

2.1. Forming the interface and compatible surface mesh for each subdomain

The intersections of the cutting planes with the surface triangles are calculated. Then the intersection points are sorted such, that they form a closed and continuous cross-section contour. These points and contour are used for the construction of two-dimensional Delaunay triangulation of the cross-section surface. TRAINGLE 1.5 [44] has been employed for generating high-quality Delaunay triangulation with constraints on minimal triangle angle and maximum triangle area. The intersected triangles are split into three other triangles so the domain can be decomposed into subdomains along the contours (see fig. 7).

2.2. Load balancing

The load balancing always has been a big issue for a parallel applications algorithm. Several well known techniques and criteria are considered in the paper.

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

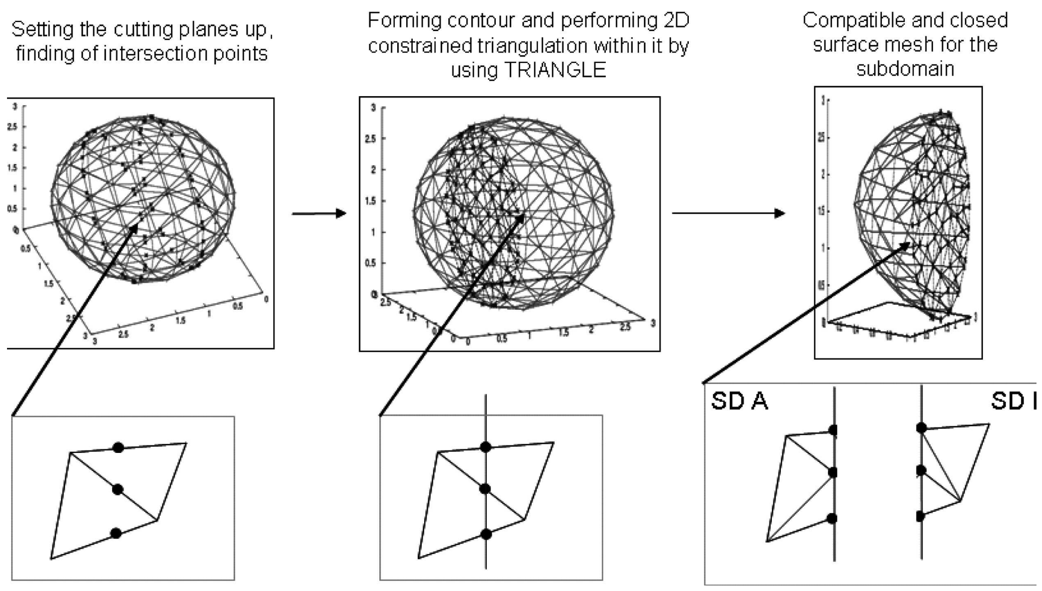

Setting the cutting planes up, finding of intersection points
Forming contour and performing 2D constrained triangulation within it by using TRIANGLE
Compatible and closed surface mesh for the subdomain
SD A
SD I

Fig. 7. Forming the interface and surface mesh for each subdomain.

— **Recursive prepartitioning.** An object cut in two. Then for each of the parts the cutting plane is chosen and the parts are cut in two and so on. Note, that first task is sequential, second task involves two parallel tasks, and the kth step involves 2^{k-1} ones.

— **Overdecomposition.** An object is decomposed into many subdomains. Number of subdomains is much larger than processors exist. Then in the case of load imbalance the master process gives the task to idle processor.

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

The moment of inertia criterion requires more detailed explanation. Each object is cut perpendicular to its smaller principal inertia axis. It means that for each part with the set of nodes V, the inertia matrix should be computed, where (x_g, y_g, z_g) are the coordinates of the center of gravity V. Then one of the eigenvectors with the smallest eigenvalue is selected. This gives us family of planes.

$$\begin{bmatrix} \sum\limits_{v\in V}(y_v-y_g)^2+(z_v-z_g)^2 & -\sum\limits_{v\in V}(x_v-x_g)(y_v-y_g) & -\sum\limits_{v\in V}(x_v-x_g)(z_v-z_g) \\ -\sum\limits_{v\in V}(y_v-y_g)(x_v-x_g) & \sum\limits_{v\in V}(z_v-z_g)^2+(x_v-x_g)^2 & -\sum\limits_{v\in V}(y_v-y_g)(z_v-z_g) \\ -\sum\limits_{v\in V}(z_v-z_g)(x_v-x_g) & -\sum\limits_{v\in V}(z_v-z_g)(y_v-y_g) & \sum\limits_{v\in V}(x_v-x_g)^2+(y_v-y_g)^2 \end{bmatrix}$$

“Node inertia matrix”

Nevertheless it happened to be hard to find a reasonable criterion for predicting a good load balancing in advance. Even if the number of elements is approximately the same for each subdomain, the CPU time spent on each part could be quite different [32].

The volume criterion was chosen in present work. The cutting planes are located so that the volumes of the subdomains are the same. The initial cut is done by equidistant set of planes and then the planes are iteratively moved to achieve the equality of volumes.

2.2.1. Orientation of boundary facets

It is important to mention that after operations on the surface mesh such as cutting of triangles, insertion of new nodes, constructing of a new mesh on the interfaces, it could happen that boundary faces are not properly oriented. The Delaunay volume grid generator is sensitive to the orientation of the boundary faces. Also for volume comparison it is necessary to have properly oriented triangles.

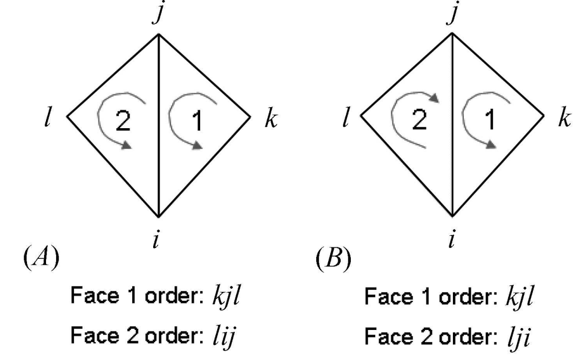


Fig. 8. Correct orientation of two adjacent triangles in (A).

A technique described in [30] has been used. Two adjacent triangles are considered to have the same orientation if the shared edge exists on order ij on the first triangle, and the same edge ordered ji on the other triangle. In fig. 8 two adjacent triangles are oriented correctly in (A). Original boundary faces always point in the correct direction, because their orientation is preserved. So the algorithm starts from original boundary face and change the order of vertices which are oriented differently.

2.2.2. Volume calculation

Gauss's Theorem relates the divergence of a vector field within a volume to the flux of a vector field through a closed surface by:

$$\iiint_V \nabla F dv = \iint_S F da, \tag{1}$$

where the surface S encloses the volume V, ∇ is defined by:

$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right). \tag{2}$$

When F is set to:

$$F(x, y, z) = (x, 0, 0). \tag{3}$$

Then, $\nabla F = 1$ and equation (1) becomes the volume

$$\iiint_V 1 dv = \iint_S F da. \tag{4}$$

Equation (4) tells us that the flux of the vector field F through the closed surface S is equal to the volume enclosed by S. When S is defined by a set of triangles, the enclosed volume is equal to the sum of the flux of F through every triangle. A straightforward parameterization of the triangle $\{V_1, V_2, V_3\}$ is given by:

$$e_1 = v_2 - v_1, \quad e_2 = v_3 - v_1, \quad s(u, v) = v_1 + ue_1 + ve_2, \tag{5}$$

where the evaluation of $s(u, v)$ yields any point on the triangle for $u \in [0, 1]$ and $v \in [0, 1 - u]$. Under this parameterization the surface element da from equation (4) can be written as:

$$da = (e_1 \times e_2)\ dvdu. \tag{6}$$

The flux of F through the triangle $\{V_1, V_2, V_3\}$ becomes

$$\Phi = \int\limits_0^1 \int\limits_0^1 F\left(s(u, v)\right)\left(e_1 \times e_2\right)\ dvdu; \tag{7}$$

$$\Phi = \int\limits_0^1 \int\limits_0^1 \left(v_{1x} + ue_{1x} + ve_{2x}\right)\left(e_{1y}e_{2z} - e_{1z}e_{2y}\right)\ dvdu. \tag{8}$$

The solution to equation (8) can be written as:

$$\Phi = \frac{1}{6}\left[\left(v_{2y} - v_{1y}\right)\left(v_{3z} - v_{1z}\right) - \left(v_{2z} - v_{1z}\right)\left(v_{3y} - v_{1y}\right)\right]\left(v_{1x} + v_{2x} + v_{3x}\right). \tag{9}$$

The right hand side of equation (9) yields the flux of F through the triangle $\{V_1, V_2, V_3\}$. Given a closed and clockwise wound triangle set $\{V_{i1}, V_{i2}, V_{i3}\}$ for $i = \{0, 1, 2, ..., n\}$ the enclosed volume is computed by evaluating the sum:

$$volume = \frac{1}{6} \sum_i \left[(v_{i2y} - v_{i1y})(v_{i3z} - v_{i1z}) - (v_{i2z} - v_{i1z})(v_{i3y} - v_{i1y}) \right] (v_{i1x} + v_{i2x} + v_{i3x}) \,. \quad (10)$$

2.3. Mesh optimization

As it is seen in fig. 9 the cutting of the surface mesh can produce "bad triangles" — triangles with small angles. One of the ways to get rid of them is to perform mesh optimization. An optimization technique described in [46] has been used for improving the mesh.

Given a set of data points scattered in three dimensions and an initial triangular mesh M_0, the task is to produce a mesh M, of the same topological type as M_0, that fits the data well and has a small number of vertices.

Here it is important to distinguish connectivity of the vertices and their geometric positions.

Formally a mesh M is a pair (K, V), where K is a simplicial complex representing the connectivity of the vertices, edges and faces, thus determining the topological type of the mesh and $V = \{v_1, \ldots, v_m\}$, $v_i \in R^3$ is a set of vertex positions defining the shape of the mesh in R^3 (its geometric realization).

We find a simplicial complex K and a set of vertex positions V defining a mesh $M = (K, V)$ that minimizes the energy function

$$E(K, V) = E_{\text{dist}}(K, V) + E_{\text{rep}}(K) + E_{\text{string}}(K, V).$$

The distance energy $E_{\text{dist}}(K, V)$ is equal to the sum of squared distances from a given set of points to the mesh. It measures the closeness of fit. The representation energy $E_{\text{rep}}(K)$ is proportional to the number of vertices m of K. It penalizes meshes with a large number of vertices. The spring energy $E_{\text{string}}(K, V)$ places on each edge a spring of rest length 0 and spring constant k. It guarantees existence of a minimum. More detailed information can be found in [46].

This optimization method is used in computer graphics applications. In our specific case we are optimizing mesh for further FEM calculations. So in present work we are not changing the

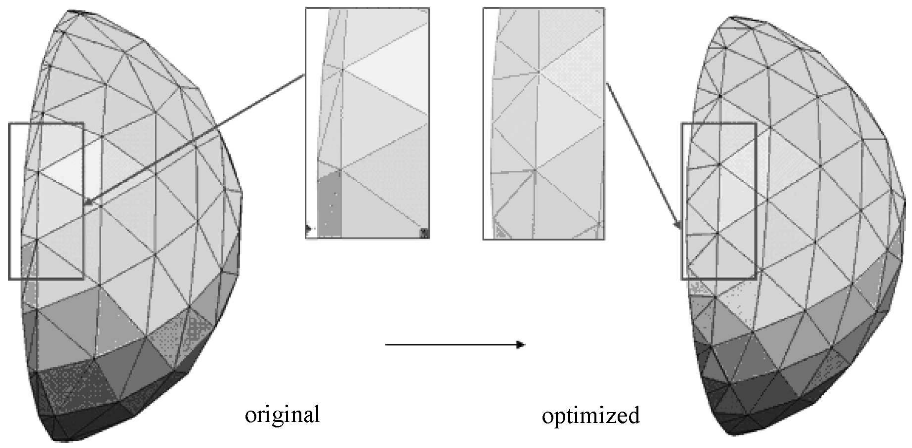


Fig. 9. Mesh optimization.

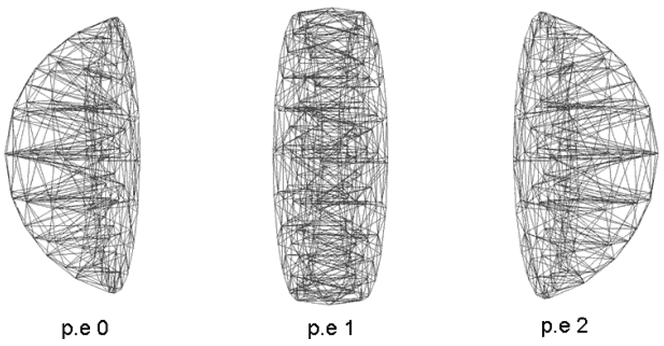


Fig. 10. Independent volume mesh generation within each subdomain.

topology of the mesh and the number of its nodes, but are just moving the mesh nodes. As it can be seen in fig. 9 this optimization technique improves the quality of the mesh.

2.4. Parallel volume meshing

After a compatible and closed surface mesh for each subdomain is constructed, it is possible to generate the volume mesh inside independently. The TetMesh-GHS3D meshing software component [45] has been used for volume meshing. It creates tetrahedral Delaunay volume meshes from closed triangular surface meshes (fig. 10).

3. Parallelization

The MPI library has been used for the implementation of the parallel grid generator. A Single Program — Multiple Data computational model was employed. It means that each processor runs the same executable program so the message passing must be masked by IF statements, for example.

If we consider one processing element for the parallel grid construction, then the sequence of its actions can be described by scheme shown in fig. 11.

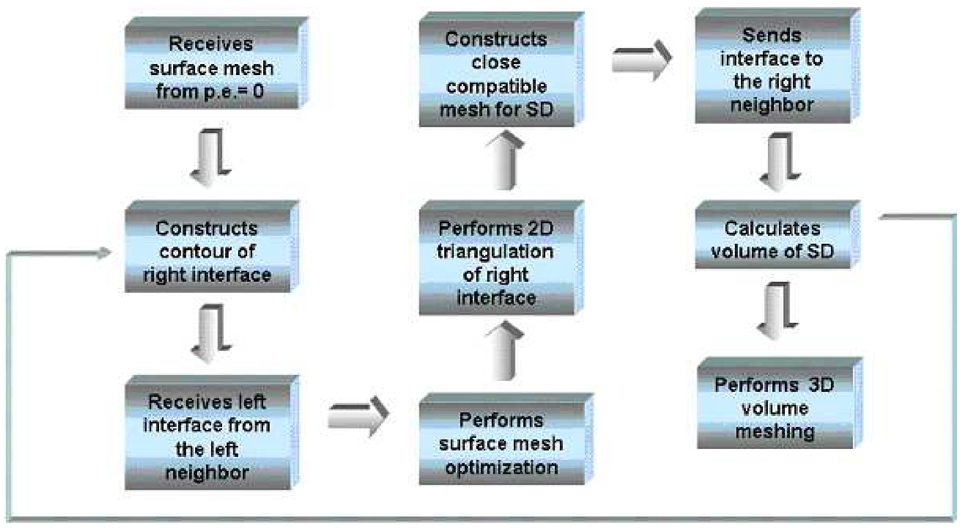

Receives surface mesh from p.e.= 0
Constructs contour of right interface
Receives left interface from the left neighbor
Performs surface mesh optimization
Performs 2D triangulation of right interface
Constructs close compatible mesh for SD
Sends interface to the right neighbor
Calculates volume of SD
Performs 3D volume meshing

Fig. 11. Scheme of the sequence of actions for one processing element (p.e.).

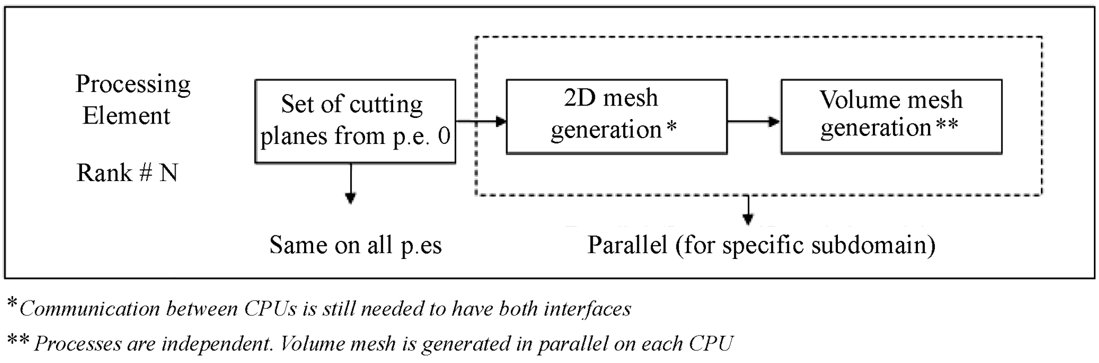

Processing
Element
Rank # N
Set of cutting planes from p.e. 0
2D mesh generation *
Volume mesh generation **
Same on all p.es
Parallel (for specific subdomain)
* Communication between CPUs is still needed to have both interfaces
** Processes are independent. Volume mesh is generated in parallel on each CPU

Fig. 12. Parallel and sequential actions.

It is clear that the construction of the two-dimensional mesh, cutting of volume and other operations with the surface mesh are easy computations with maximum running time $O(N^2)$, where N is a number of mesh nodes. Hence it is not computationally expensive and, could be done sequentially. But from an algorithmic point of view it is easier to do it in parallel. Then one does not have to take care of the data distribution since each processing element already has the data. The goal of all parallelization is to perform simultaneous construction of three-dimensional grid in each subdomain. This is the heaviest computations with running time $O(N^3)$ (fig. 12).

4. Coupling parallel grid generator with DDFEM solver

DDFEM [47] is a parallel 3D linear elasticity solver for steady-state problems, developed at ITWM. The parallelization is based on the domain decomposition approach and PETSc routines are used for the parallel solution of the linear system. At the moment the program reads the whole mesh and then partitions it among the CPUs by using METIS [48]. This sequential step can be a real bottleneck when large meshes are involved. Therefore it is necessary to couple a parallel grid generator with the parallel solver in order to provide the mesh for each CPU and remove the bottleneck. So the "throughout" use of parallel programming will be achieved.

After independent generation of the volume meshes we have several meshes with local node enumeration. It is necessary to have global enumeration of nodes and elements because the solver has to distinguish nodes which are in one subdomain only from those which are common for two neighboring subdomains (fig. 13).

In fig. 14 an example solution is shown. The mesh was generated by the parallel grid generator and then the elasticity problem was solved by DDFEM. Dirichlet boundary conditions

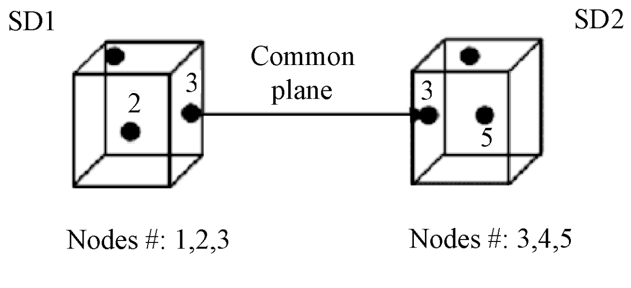


Fig. 13. Global enumeration of the nodes.

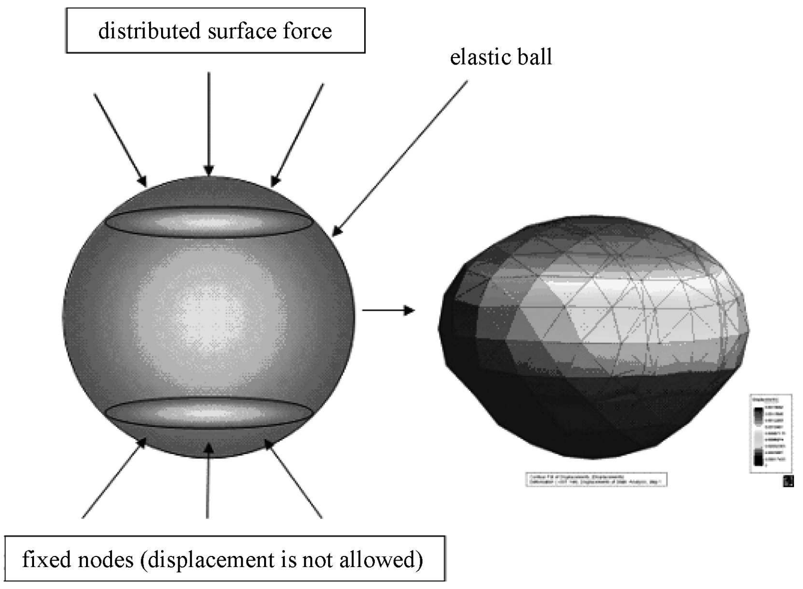

distributed surface force
elastic ball
fixed nodes (displacement is not allowed)

Fig. 14. Elasticity problem. Deformation of elastic ball under applied force.

were applied to the bottom (displacement is not allowed) and face traction boundary conditions at the top (distributed surface force).

5. Summary and conclusion

A parallel generator for unstructured three-dimensional tetrahedral grids has been developed. The algorithm based on domain decomposition approach was described in the paper. This algorithm allowed us to use standard sequential 2D and 3D triangulators in parallel. The programs TRIANGLE [44] and TetMesh-GHS3D [45] 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 are also explored in the paper. The surface mesh optimization technique has been used for improving the mesh quality. The parallelization strategy is based on Single Program — Multiple Data computational model and employs the MPI library for the implementation of the parallel grid generator. Hence the most expensive computations, the construction of volume mesh inside each subdomain is totally independent and performed in parallel. The developed parallel grid generator has been coupled with the DDFEM solver in order to remove the grid generation bottleneck and achieve “throughout” use of parallel programming at all stages of solving the problem. Further work on testing, handling real-life examples with complex geometries, automatic evaluation of the mesh quality, improving of surface mesh optimization technique, creating of an interface with CAD and performing local adaptive mesh refinement is in progress.

6. Acknowledgements

This work is supported by the DAAD, TU Kaiserslautern, Fraunhofer Institute of Industrial Mathematics, and Institute of Theoretical and Applied Mechanics SB RAS. The author would like to thank Heiko Andrae, Dimitar Stoyanov, and Alexey Kudryavtsev for their careful supervision and help. Also the author is grateful to Emanuel Teichmann and Robert Zillich who made contributions to this work and proposed corrections of the paper. The author wishes to thank Academician S.K. Godunov for the interest to this work and many comments during the presentation on his seminar.

References

[1] Delaunay B. Sur la sphère vide // Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk. 1934. Vol. 7. P. 793–800.

[2] Jin H., Tanner R.I. Generation of unstructured tetrahedral meshes by the advancing front technique // Intern. J. Num. Meth. Eng. 1993. Vol. 36. P. 1805–1823.

[3] Liseikin V.D. Grid Generation Methods. Berlin; Heidelberg: Springer-Verlag, 1999.

[4] Cignoni P., Montani C., Scopigno R. DeWall: A fast divide & conquer Delaunay triangulation algorithm in ed // Computer-Aided Design., Elsevier Sci. 1998. Vol. 30, N 5. P. 333–341.

[5] Guibas L.J., Knuth D.E., Sharir M. Randomized incremental construction of Delaunay and Voronoy diagrams // Lect. Note Comp. Science. Springer-Verlag, 1990. P. 414–431.

[6] Baum J.D., Luo H., Löhner R. Numerical simulation of a blast inside a Boeing 747 // AIAA-93-3091. 1993.

[7] Baum J.D., Luo H., Löhner R. Numerical simulation of a blast in the World Trade Center // AIAA-95-0085. 1995.

[8] Jou W. Comments on the Feasibility of LES for Commercial Airplane Wings // AIAA-98-2801. 1998.

[9] Yoshimura S., Nitta H., Yagava G., Akiba H. Parallel automatic mesh generation method of ten-million nodes problem using fuzzy knowledge processing and computational geometry // 4th World CongComp. Mech. Buenos Aires, 1998.

[10] Mavriplis D.J., Pirzadeh S. Large-Scale Parallel Unstructured Mesh Computations for 3-D High Lift Analysis // ICASE Rep. 99-9. 1999.

[11] Darve E., Löhner R. Advanced Structured-Unstructured Solver for Electromagnetic Scattering from Multimaterial Objects // AIAA-97-0863. 1997.

[12] Morgan K., Brooks P.J., Hassan O., Weatherill N.P. Parallel processing for the simulations of problems involving scattering of electromagnetic waves // Proc. Symp. Advances in Comp. Mech. 1997.

[13] Löhner R., Cebral J.R. Parallel advancing front grid generation // 8th Intern. Meshing Roundtable South Lake Tahoe. CA U.S.A., 1999. P. 67–74.

[14] Löhner R. Three-dimensional fluid-structure interaction using a finite element solver and adaptive remeshing // Comp. Sys. in Eng. 1990. Vol. 1, N 2–4. P. 257–272.

[15] Mestreau E., Löhner R., Aita S. TGV Tunnel-Entry Simulations Using a Finite Element Code with Automatic Remeshing // AIAA-96-0798. 1996.

[16] Mestreau E., Löhner R. Airbag Simulations Using Fluid/Structure Coupling // AIAA-96-0798. 1996.

[17] Baum J.D., Luo H., Löhner R. et al. A Coupled Fluid/Structure Modeling of Shock Interaction with a Truck // AIAA-96-0795. 1996.

[18] Kamoulakos A., Chen V., Mestreau E., Löhner R. Finite element modeling of fluid/structure interaction in explosively loaded aircraft fuselage panels using PAMSHOCK/PAMFLOW coupling // Conf. on Spacecraft Structures, Materials and Mechanical Testing. Noordwijk, The Netherlands, 1996.

[19] Löhner R., Yang C., Cebral J. et al. Fluid-Structure-Thermal Interaction Using a Loose Coupling Algorithm and Adaptive Unstructured Grids // AIAA-98-2419. 1998.

[20] Hassan O., Bayne L.B., Morgan K., Weatherill N.P. An adaptive unstructured mesh method for transient flows involving moving boundaries // Comp. Fluid Dynamics. 1998. Vol. 98. P. 662–674.

[21] Baum J.D., Luo H., Löhner R. The Numerical Simulation of Strongly Unsteady Flows With Hundreds of Moving Bodies // AIAA-98-0788. 1998.

[22] Baum J.D., Luo H., Mestreau E. et al. A coupled CFD/CSD Methodology for Modeling Weapon Detonation and Fragmentation // AIAA-99-0794. 1999.

[23] Löhner R., Camberos J., Merriam M. Parallel unstructured grid generation // Comp. Meth. Appl. Mech. Eng. 1995. Vol. 95. P. 343–357.

[24] De Cougny H.L., Shephard M.S., Ozturan C. Parallel Three-dimensional mesh generation // Comp. Systems in Eng. 1994. Vol. 5. P. 311–323.

[25] Shostko A., Löhner R. Three-dimensional parallel unstructured grid generation // Intern. J. Num. Meth. Eng. 1995. Vol. 38. P. 905–925.

[26] De Cougny H.L., Shephard M.S., Ozturan C. Parallel three-dimensional mesh generation on distributed memory MIMD computers // Rensselaer Polytechnic Institute Tech. Rep. 1995. Vol. 7.

[27] Okusanya T., Peraire J. Parallel unstructured mesh generation // 5th Intern. Conf. Num. Grid Generation in CFD and Related Fields. Mississippi, 1996.

[28] Chew L.P., Chrisochoides N., Sukup F. Parallel constrained delaunay meshing // Workshop on Trends in Unstructured Mesh Generation, 1997.

[29] Okusanya T., Peraire P. 3-D Parallel Unstructured Mesh Generation // Joint ASME/ASCE/SES Summer Meeting, 1997.

[30] Said R., Weatherill N.P., Morgan K., Verhoeven N.A. Distributed parallel Delaunay mesh generation // Comp. Meth. Appl. Mech. Eng. 1999. Vol. 177. P. 109–125.

[31] Ivanov E.G. Automatic parallel generation of unstructured three-dimensional grids // Conf. on Stability and Turbulence of Homogeneous and Heterogeneous Fluids Flows. Novosibirsk, 2005. P. 79–82.

[32] Galtier J., George P.L. Prepartitioning as a way to mesh subdomains in parallel // 5th Intern. Meshing Roundtable Sandia National Laboratories, 1998. P. 107–122.

[33] Larwood B.G., Weatherill N.P., Hassan O., Morgan K. Domain decomposition approach for parallel unstructured mesh generation // Intern. J. Num. Meth. Eng. 2003. Vol. 58. P. 177–188.

[34] Chrisochoides N. Demian nave parallel Delaunay mesh generation kernel // Intern. J. Num. Meth. Eng. 2003. Vol. 58. P. 161–176.

[35] Wu P., Houstis E.N. Parallel adaptive mesh generation and decomposition // Eng. With Computers. 1996. Vol. 12. P. 155–167.

[36] Chrisochoides N. Demian nave simultaneous mesh generation and partitioning for Delaunay meshes // 8th Intern. Meshing Roundtable South Lake Tahoe. CA U.S.A., 1999. P. 55–66.

[37] Cignoni P., Montani C., Perego R., Scopigno R. Parallel 3D Delaunay triangulation // Computer Graphics Forum. 1993. Vol. 12, N 3. P. 129–142.

[38] Cignoni P., Laforenza D., Montani C. et al. Evaluation of parallelization strategies for an incremental Delaunay triangulator in E3 // Practice and Experience. 1995. Vol. 7, N 1. P. 61–80.

[39] Chetverushkin B.N., Gasilov V.A., Polyakov S.V. et al. Data structures and mesh processing in parallel CFD project GIMM // To Appear in Proc. ParCo2005, 13–16 Sept., 2005.

[40] Barragan A.J., Reeve J.S. Parallel, three-dimensional finite element mesh generation based on octree data structures // Europ. Congress on Comp. Methods in Appl. Sci. and Eng. Barcelona, 2000.

[41] Chrisochoides N., Chernikov A., Barker K. Parallel programming environment for mesh generation // Proc. of ICNGG Waikiki Beach Honolulu Hawaii, 2002.

[42] Walshaw C., Cross M., Everett M.G. Parallel unstructured mesh partitioning // Domain Decomposition Methods in Sci. and Eng. Bergen, Norway, 1998. P. 647–654.

[43] Meshing Software Survey // http://www.andrew.cmu.edu/user/sowen/softsurv.html

[44] Shewchuk J.R. Triangle: engineering a 2d quality mesh generator and Delaunay triangulators // Proc. First Workshop on Appl. Comp. Geometry. PA USA, 1996. P. 124–133.

[45] TetMesh-GHS3D V3.1 the fast, reliable, high quality tetrahedral mesh generator and optimizer // white paper, available at http://www.simulog.fr/mesh/tetmesh3p1d-wp.pdf

[46] Hoppe H., DeRose T., Duchamp T. et al. Mesh optimization // ACM SIGGRAPH. 1993. P. 19–26.

[47] Andrae H., Stoyanov D. DDFEM: a parallel finite element solver for linear elasticity // Internal ITWM Report 2005.

[48] METIS: Multilevel Partitioning Algorithms // available at http://www-users.cs.umn.edu/ karypis/metis/

Received for publication October 6, 2005