Adaptive 3D unstructured mesh refinement

I Gruzintsev[1], *M* Kornilina[2,*], *M* Yakobovskiy[2]

[1]Moscow Institute of Physics and Technology, Russia
[2]Keldysh Institute of Applied Mathematics of Russian Academy of Sciences, Russia

Abstract. Algorithms for generating three-dimensional detailed computational meshes are considered. The algorithms are based on adaptive refinement of the original coarse meshes describing a 3D object. The purpose of adaptation is to form an accurate description of the volume and surface of a three-dimensional object for supercomputer modeling. Refinement of the boundary description is performed by projecting the cut elements of the coarse mesh onto the corresponding elements of the object's surface.

1 Introduction

In the numerical HPC modeling of physical processes, discrete computational meshes are widely used for description of the geometry of particular problem (or general domain). High requirements are imposed on discrete meshes in terms of the level and quality of discretization, in order to ensure the required computing accuracy.

Detailed meshes consist of a large number of elements, which leads to higher computation cost. Still large meshes can be processed very efficiently on modern supercomputers. Most HPC systems aggregate multiple processors, processor cores, graphics accelerators, and other operating devices. Effective well-coordinated operation of such complex computing systems can be performed only if the workloads consist of a large amount of mutually independent computing operations, which can be simultaneously processed on any operating device. From this point of view, massive computational meshes allow to effectively adapt numerical algorithms to the supercomputer architecture, and thereby reduce the computational time by involving a lot of processors.

Computational meshes can be divided into two large groups: structured and unstructured. The difference between them lies in the way of definition of their elements. In a structured mesh all elements and their interconnection are totally defined by their computational coordinates. That's why the elements can be easily recomputed and they are usually not stored in the computer memory. For unstructured meshes, this functional relationship does not exist. Use of unstructured meshes usually requires more complex algorithms and a large amount of RAM per mesh element. Due to the architecture of modern computer nodes, especially processors and RAM, the use of unstructured meshes increases execution time of elementary operations. The latter is caused by the more chaotic accesses to RAM addresses, and therefore, to the poor use processors' cache. However, the unstructured meshes have a

*Corresponding author: mary@imamod.ru

number of advantages, which have gained an increased interest to them, especially recently. We can name two main advantages of unstructured meshes over structured ones:

1) more accurate description of boundaries of geometric objects, without effect of gradation for curved boundaries;

2) less computational mesh elements due to the use of enlarged mesh cells in not significant areas.

The construction of computational grids containing 10^9 or more elements is a rather difficult challenge. Further, we consider a way to improve the quality of 3D domain discretization, which is based on the adaptive refinement of relatively coarse three-dimensional tetrahedral meshes formed using available third-party mesh generators.

2 Problem statement

There are several aspects in the considered problem, both algorithmic and technological. We have a great diversity at each step of construction of refined mesh:

1) geometry descripting of 3D object via CAD system;

2) generating of a coarse 3D mesh;

3) cutting of the coarse 3D mesh;

4) adaptation of the subdivided mesh to the boundaries of the 3D object;

5) achieving the desired quality of the adapted mesh.

At the step 1 a three-dimensional object may be described using geometric transformations and set-theoretic operations over basic 2D and 3D primitive shapes (such as polyhedral, sphere, objects defined by spline surfaces, and others). Let's consider an example of approximation of a boundary fragment of a two-dimensional object (Fig. 1). In Fig. 1a a triangle of a coarse mesh is highlighted in bold, and the result of its single dividing is represented by thin cuts. The cut is made through additional vertices on the edges.

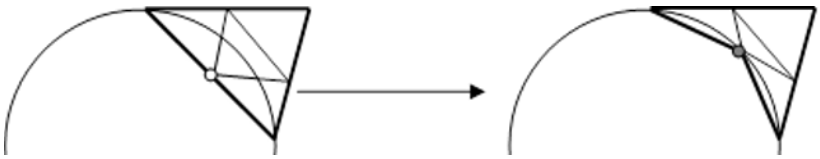

Fig. 1. Refining of the computational mesh.

It is evident, that in general a growth of mesh cells does not necessarily result in a better accuracy of the description of object boundary (Fig. 1a). Only two vertices are still located on the object boundary, although an additional vertex has been added to the edge which approximates this curve. The problem can be solved by shifting the added vertex to the boundary of the object (Fig. 1b), that will provide a better accuracy of approximation. However, to do this, we should at least have the knowledge about curve which should be approximated by this the edge. A significant problem is that usually when the coarse mesh is generated all excessive data describing the source object (e.g. spline parameters) and its cross-reference with the mesh elements. This is a technical problem, which can be solved in many ways and it is beyond the scope of this article. Here we present an approach that partially eliminates the need to store such data.

At step 2 the following two consistent discrete meshes should be constructed:

- coarse 3D mesh (Fig. 2);
- a fine surface mesh (Fig. 3).

Both meshes are formed by the same third-party software package. In this work the Salome integrated CAD-CAE system [2] is used for generation of the coarse tetrahedral

mesh. This system can be used to construct a surface triangulation of the object boundary as well.

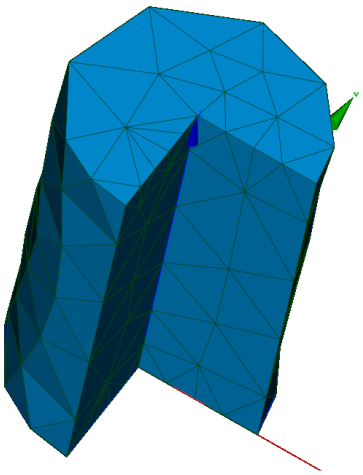

Fig. 2. Coarse 3D mesh describing a cylinder fragment.

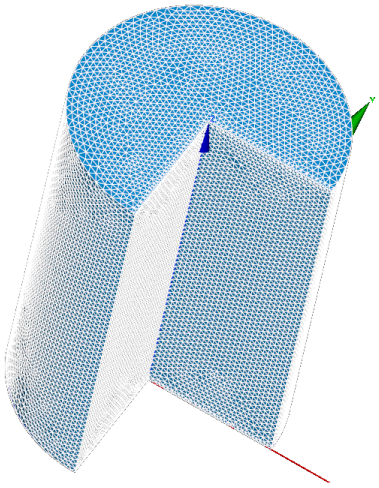

Fig. 3. Fine boundary mesh describing a cylinder fragment.

Further refinement should be performed by adapting the boundary of the generated mesh to the boundary of the 3D object described by a fine surface grid. Experiments demonstrate that the good results can only be achieved if a refined mesh has at least some hundred times more surface nodes than the number of triangles on the surface of a coarse 3D mesh. The memory required for storing a surface mesh is much less than the memory required for storing a volume mesh with the same number of nodes, since in the former case far less data is required about the interconnection of mesh elements. So according to estimations a fine surface mesh on the average by two orders of magnitude more detailed than the surface fragment of a coarse volume mesh can be generated quite successfully.

The reason for using Salome generator is due to the technological sequence underling the package for solving hydrodynamic problems MARPLE3D developed by our team. The data structure of the MARPLE3D code is quite compatible with Salome, i.e. it permits processing of geometric information created by this platform. There is a data Converter from the Salome system to an intermediate format. This format is, in its turn, converted to the internal format of the MARPLE3D package [1]. The intermediate format that is obtained after the SALOME-to-MARPLE3D converter is itself a very natural and clear format of setting the computational mesh, thus the codes that implement refinement algorithms are mainly focused on this format.

One of the main requirements for the refinement algorithms is that the constructed tetrahedral mesh should reproduce all the main peculiarities of the domain boundary, including sharp edges and conical vertices. From Salome generator one can get vertices that lie in the intersection of physical subdomains which describe the boundary of the computational domain. These vertices mostly define narrow elements with sharp edges, since they belong to the intersections of surfaces. But you will never say that there aren't any other sharp edges on the surface. Therefore, the algorithms are required for adapting of neighboring cells to such edges without any preliminary indicating them while formation of computational domain.

This problem is somewhat discussed i.e. in [9, 10]. However, in [10], the initial surface mesh is adapted to the domain defined as an isosurface of an implicitly defined function, while in [9], the general issues of constructing computational meshes are considered, and adaptation to the surface is considered along with improving of tetrahedrons quality.

Note, that in this paper, the algorithms of step 5 improving the quality of the 3D grid are not considered. The corresponding activities can be performed, for example, using the ANI3D package [13].

The results visualized using the ParaViewViewer package.

3 Refinement algorithms

3.1 Non-adaptive subdivision of coarse mesh

At step 3 subdivision of coarse 3D mesh can be performed by any algorithm enabling to subdivide tetrahedra. In this work, the algorithm [12] was used. According to this algorithm, all tetrahedra of the initial grid were divided into 4 tetrahedra and one octahedron (Fig. 4). New vertexes were added to the midpoints of the edges. Then each octahedron was divided into 4 tetrahedra (Fig. 5). The new edge is placed so that the quality of the resulting tetrahedra would be the best. Quality criterion was the fallowing: the ratio of the longest and shortest edges lengths should be minimal [7]. A more detailed description of this algorithm can be found in [3].

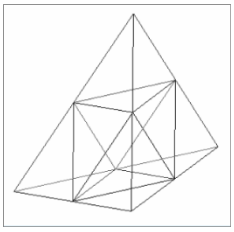

Fig. 4. Cutting of tetrahedron.

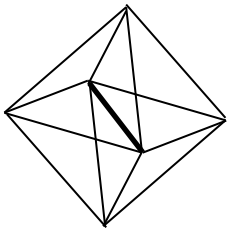

Fig. 5. Cutting of octahedron.

3.2 Adaptive subdivision of coarse mesh

Two algorithms for adaptive subdividing of computational grids have been developed. They differ in the way how the sharp edges of the 3D object surface can be determined. The first algorithm does not use any a priori information whether the elements of a fine surface grid belong to the boundaries of the initial surfaces or not. The second algorithm uses available information, for example, due to storing a fine surface grid in GMF format, which incorporates the data about the junctions of surfaces (all edges are listed with the constituent vertexes). Thus, sharp edges can be identified at the stage of the initial mesh generation.

3.3 Formalization of the problem statement

Let us consider the general problem of constructing tetrahedral grids in implicit domains and formulate the necessary requirements to the constructed three-dimensional computational grid [9]. A three-dimensional domain Ω with a boundary $\partial\Omega$ is approximated by a polyhedron Ω_h with boundary $\partial\Omega_h$ composed of flat triangles joint together by the whole edges. A normal tetrahedral partition T is constructed inside the polyhedron Ω_h, at that the set of boundary triangles forms a subset of the partition faces. When the size of the edges of the boundary grid approach zero the surface $\partial\Omega_h$ should converges pointwise to $\partial\Omega$. In addition, it is necessary that the piecewise constant field ν_h of unit normals to $\partial\Omega_h$ should converges pointwise to a piecewise continuous field of unit normals ν of the surface $\partial\Omega$. The parameter h plays the role of the cell size, i.e. it is assumed that the maximum length of the grid edge T

does not exceed Ch, where C is a constant. There is no independent parameter h in this problem statement, and the boundary of the domain $\partial\Omega$ is represented by a set of triangles of the fine mesh $\partial\Omega_m$. Repeated subdividing of a tetrahedral mesh is performed by sequential executing the step of dividing of the initial coarse mesh, or the mesh obtained in the previous iteration.

Let's consider one step of adaptive subdividing (hereinafter we will call this a step iteration). As a result of its execution, the elements of the polyhedron Ω_{n1} whose boundary is a set of triangles $\partial\Omega_{n1}$, are replaced by the polyhedron Ω_{n2}, whose boundary is $\partial\Omega_{n2}$. Here n_1, n_2 is the number of elements describing the surface of polyhedra, and $m \gg n_2 > n_1$. Let us briefly describe the corresponding adaptive algorithms for mesh subdividing. In detail they are presented in [12].

3.4 Adaptive algorithm without use of information about the fine mesh vertices located on the domains boundaries (A_1)

Algorithm A_1 consists of the following steps:

1. Non-adaptive cutting of coarse mesh.
2. Adaptation of the boundary of constructed mesh to the boundary of calculation domain (fine mesh). At this step the iterative shift of new vertices located on the edges of the triangles of the course mesh boundary is performed by the algorithms presented in [12].
3. Checking the obtained grid for the presence of degenerate triangles and tetrahedral.
4. In case of degenerate elements the mesh should be corrected. Correction is always possible, since step 1 of this algorithm does not lead to the appearance of degenerate elements. Thus, in the worst case, an unadapted grid will be constructed.

The results of the algorithm A_1 show that the heuristic nature of compliance determining for the triangles of the target boundary and the boundary formed by mesh in many cases makes impossible the proper adaptation of sharp edges even for relatively simple objects (Fig. 6). In many cases, only shift is made towards the junctions of surface fragments, but the vertices do not fall on the junctions themselves.

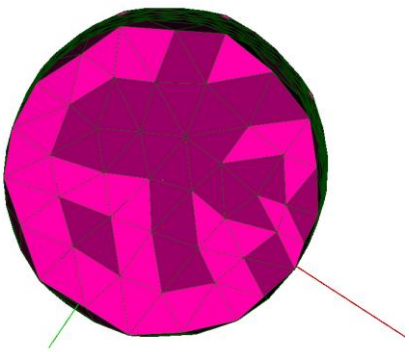

Fig. 6. Result of the algorithm A1.

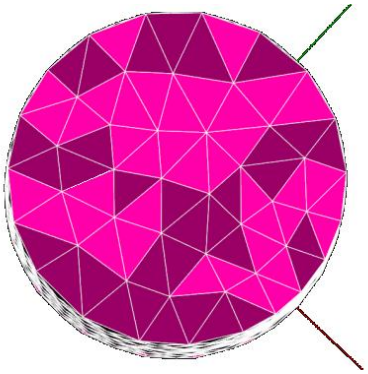

Fig. 7. Result of the algorithm A2.

For a more accurate reproduction of sharp edges and to reduce the number of formed degenerate elements, it is necessary to fully use the information obtained from the initial mesh files. The use of meshes in the GMF format for recording ensures preservation of information about the junctions of surfaces. Thus, the edges of the junctions are selected at the stage of initial mesh constructing using the complete information about the surface of 3D object.

3.5 Adaptive algorithm using information about the fine mesh vertices located on the domains boundaries (A_2)

1. Non-adaptive cutting of coarse mesh.
2. For all vertices that are not marked as belonging to surface junctions, the nearest vertex with a fine surface mesh around it should be searched for.
3. All vertices of the coarse 3D mesh that were initially marked as belonging to the junctions of surfaces are considered. For such vertices, the closest one is found among the vertices of the fine mesh, marked at the initial stage, as residing at the junctions of the surfaces of the fine mesh. And current vertex is shifted to the found one.
4. The resulting mesh is checking for the presence of degenerate triangles and tetrahedra.
5. In case of degenerate elements, the mesh is corrected.

Results of the algorithm A_2 show that use of a priori information provides a higher quality of object boundaries approximation as compared to algorithm A_1 (Fig. 7-9).

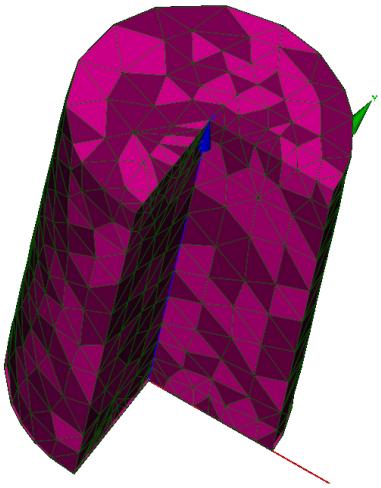

Fig. 8. Refinement, iteration step 1.

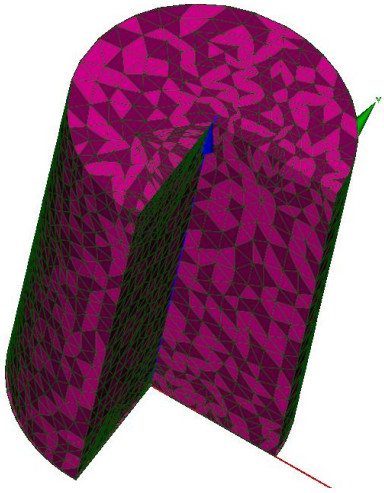

Fig. 9. Refinement, iteration step 2.

4 Further development of work

Despite the good results of the algorithm A_2, the uncontrolled occurrence of irremediable degenerate elements for the objects with complex surface shape remains an open question. The main reason of such behavior is that in the framework of the suggested approach, the triangles of the initial mesh Ω_{n1} and the constructed mesh Ω_{n2} are not adjusted to the triangles of the final fine surface mesh $\partial\Omega_m$. Therefore, errors occur in determining the proximity of elements. An element selected as the nearest one does not correspond to the required topology, which results in the appearance of degenerate elements. Preliminary estimates show that modification of the method used for coarse mesh generation can diminish the number of such errors.

Let $\partial\Omega_m^0 = \partial\Omega_m$. Let us describe the algorithm A_3 for generating the triangulation $\partial\Omega_m^{i+1}$ by uniformly coarsening of triangulation $\partial\Omega_m^i$. A similar method was used for coarsing surfaces in [14]:

- we find a set V_1 of pairwise non-adjacent vertices belonging to the junctions of the mesh $\partial\Omega_m^i$;

- we find a set V_2 of vertices of mesh $\partial\Omega_m^{\mathrm{i}}$, pairwise non-adjacent to each other and non-adjacent to the vertices of the set V_1;

- we exclude vertices $V_1 \cup V_2$ and both incident triangles and incident edges. When each vertex is excluded we do the following: a) the set of incident triangles and edges is recorded; b) the resulting gap is filled with a series of new triangles (Fig. 10); c) for each new triangle the excluded vertex P is recorded.

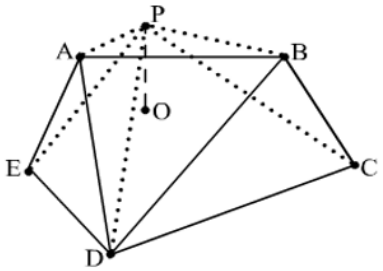

Fig. 10. The example of removing the vertex P and 5 triangles incident to it followed by filling the gap with three triangles.

The following steps for the high-quality mesh generation are suggested:

1) geometry descripting of 3D object via CAD system;
2) generating of detailed 2D surface mesh and its recording with information about the vertices and edges belonging to the boundaries of surface fragments (junctions);
3) generating of a sequence of nested meshes using k-fold application of the algorithm A_3:
   $\partial\Omega_m^1$, $\partial\Omega_m^2$,… $\partial\Omega_m^{\mathrm{k}}$
4) generating a coarse 3D mesh corresponding to $\partial\Omega_m^{\mathrm{k}}$ mesh;
5) single subdividing of the formed 3D mesh with restoration of the positions of vertices and edges which previously belonged to the junctions of the mesh $\partial\Omega_m^{\mathrm{k}-1}$;
6) one-time adaptation of subdevided 3D mesh to $\partial\Omega_m^{\mathrm{k}-1}$ mesh;
7) let $k = k - 1$;
8) if $k > 0$ go to step 5;
9) achieving the desired quality of the adapted mesh (is not discussed here) .

At step 5, for subdividing of edge both vertices of which lie at the same junction, the previously recorded (and removed) vertex is used. This vertex also lies at the junction of the surface $\partial\Omega_m$. This allows to restore the junctions accurately.

Due to accurate transfer of the boundaries and using at each iteration of surface representation with the most matching scale the proposed strategy significantly lessens the chance of obtaining degenerate elements. The logic of locating of close elements is also significantly simplified owing to the use of information about parent triangles.

5 Conclusion

Adaptive refinement algorithms are represented for generation of computational meshes with multiple mesh-elements required to provide a high accuracy of numerical supercomputer modelling of a wide range of physical processes. The testing results are presented which confirm the efficiency of the considered algorithms. The reasons for the applicability limiting of the considered methods are analyzed and a strategy for further quality improving of the developed adaptive refinement algorithms is proposed.

References

1. D'yachenko S V, Gasilova I V, Dorofeyeva Ye YU 2012 *Preprinty IPM im MVKeldysha* **36**
2. *SALOME – The Open Source Integration Platform for Numerical Simulation* http://wwwsalome-platformorg/
3. Yakobovskiy M V 2006 *Vychislitel'naya sreda dlya modelirovaniya zadach mekhaniki sploshnoy sredy na vysokoproizvoditel'nykh sistemakh: dissertatsiya 051318* (Moskva)
4. Freay P J, George P-L 2000 *Mesh Generation: application to finite elements* (Oxford & Paris: Hermes Science Publishing)
5. Owen S J 1998 *Proceedings of 7th International Meshing Roundtable* (Dearborn, MI) p 239
6. Galanin M P, Shcheglov I A 2006 *Preprinty IPM im MV Keldysha* **9** 32 https://keldyshru/papers/2006/prep09/prep2006_09html
7. Galanin M P, Shcheglov I A 2006 *Preprinty IPM im MV Keldysha* **10** 32 http://keldyshru/papers/2006/prep10/prep2006_10html
8. Danilov A A 2010 *Tekhnologiya postroyeniya nestrukturirovannykh setok i monotonnaya diskretizatsiya uravneniya diffuzii: dissertatsiya kandidata fiziko-matematicheskikh nauk: 051318* (Moskva)
9. Kudryavtseva L N 2014 *Metody samoorganizatsii i optimizatsii dlya postroyeniya trekhmernykh raschetnykh setok: dissertatsiya na soiskaniye stepeni kandidata fiziko-matematicheskikh nauk po spetsial'nosti 010107* (Moskva)
10. Ohtake Y, Belyaev A, Pasko A 2003 *The Visual Computer* **19(2)** 115–126
11. Sukov S A 2015 *Preprinty IPM im MV Keldysha* **23** 22 http://librarykeldyshru/preprintasp?id=2015-23
12. Gruzintsev I O, Yakobovskiy M V 2019 *Parallel'nyye vychislitel'nyye tekhnologii (PaVT'2019) Korotkiye stat'i i opisaniya plakatov XIII Mezhdunarodnoy nauchnoy konferentsii* (Chelyabinsk: Izdatel'skiy tsentr YUUrGU) p 223
13. Lipnikov K, Vasssilevski Y *Advanced Numerical Instruments 3D* http://sourceforgenet/projects/ani3d/
14. Yakobovskiy M V 2004 *Voprosy atomnoy nauki i tekhniki Ser Matematicheskoye modelirovaniye fizicheskikh protsessov* **2** 40