ATLANTIS PRESS

Advances in Engineering Research (AER), volume 157

International Conference "Actual Issues of Mechanical Engineering" (AIME 2018)

Development of Preprocessor of SIGMA Software for Computer Simulation in Aerospace Engineering

Yury I. Dimitrienko and Andrey A. Zakharov
Department of computational mathematics and mathematical physics,
Bauman Moscow State Technical University
Moscow, Russia
azaharov@bmstu.ru

***Abstract*— The paper covers the area of block-structured adaptive grid generation for finite difference, finite element and finite volume methods. It deals with a problem of development of information technologies and algorithms for generating such grids in complex geometries. Its key idea is to bring together the different existing techniques for computer-aided geometric design of domains, grid generation, grid concentration and grid subdivision. The methods used for a construction of domain geometries, working with block-structured quasi-continuous grids, a subdivision of block-structured adaptive grids into vertex-centered control volumes, generation of ghost nodes and cells, calculation of distances to boundaries are discussed. It is described of the software interfaces of the geometry editor and the mesh generator that were developed in the SIGMA preprocessor. The results of the generation of three-dimensional adaptive grids for the domain of external flow near a hemisphere, flow around a surface of the high-speed aircraft of Falcon HTV-2 type and for the domain of the gap between a wheel and a body of an aircraft are analyzed. The presented methods and algorithms for grid generation have the following advantages: 1. The grids are structured and mesh cells are hexahedra in the entire complex curvilinear computational domain. 2. The grids can be applied for finite difference methods. 3. The grid lines are adjusting with the flow in the whole domain.**

Keywords*— *block-structured adaptive grid, finite difference grid, computer-aided geometric design, complex domain, mesh subdivision, preprocessing

I. INTRODUCTION

Development of unified computer-aided software packages for generating adaptive grids currently represents a large-scale problem of applied mathematics, computational physics, mechanics, biology and medicine [1–3]. Existing open-source and commercial software packages mainly use mesh generation methods that are universally applicable for complex geometry and focus on the generation of unstructured finite element or finite volume grids [4–6]. Structuring and adaptation of a mesh, as well as the hexahedral shape of its cells, usually can be held only near some boundaries of the domain. At the same time, for some problems of mathematical physics (e.g. high-speed aerodynamics problems), it is important to hold the structuring and hexahedral shape of the mesh cells in the entire domain [7–9], the applicability of the mesh for finite difference methods [8–10], geometric and dynamic adaptation of the mesh in the entire domain [11]. Therefore, quadrilateral and hexahedral mesh generation has become a topic of intense research [12, 13].

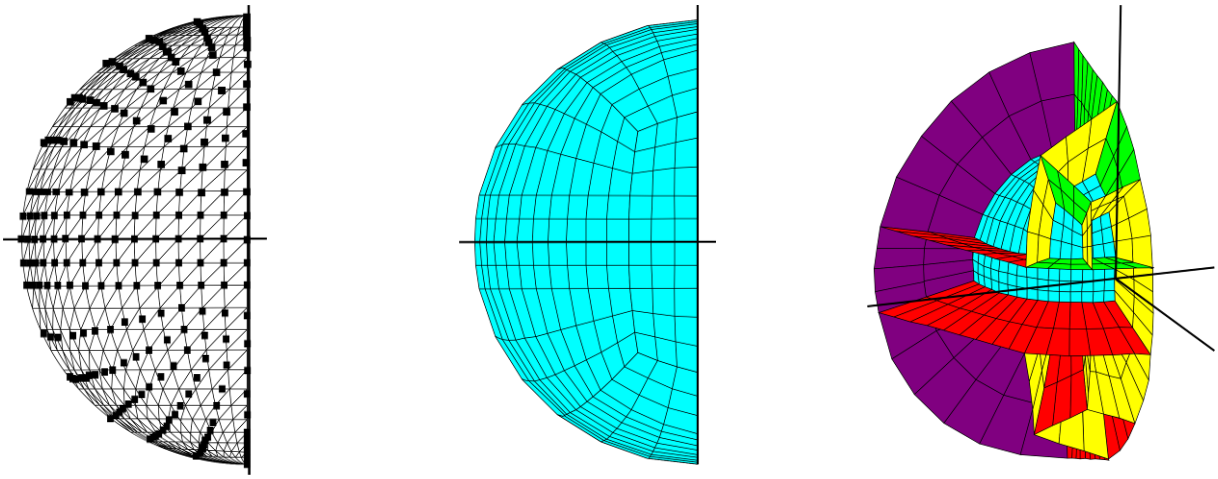
Software package SIGMA has been successfully applied for numerical solutions of a wide range of problems in aerospace engineering for over ten years (e.g. see [14, 15]). It is developed by Computational Mathematics and Mathematical Physics Department of Bauman Moscow State Technical University. It includes a full set of modules that are required for the numerical simulation. SIGMA preprocessor includes a three-dimensional geometric simulation module, which allows generating a wide range of geometric shapes; a module, which allows setting properties and initial conditions and adaptive mesh generator. Let us consider the methods and algorithms that underlie the preprocessor module.

II. CONSTRUCTION OF DOMAIN GEOMETRIES

For the generation of block-structured adaptive grids, the equations of boundary curves and surfaces need to be written in parametric form. These equations are built on the spline interpolation methods, which are based on a created grid of control points. The control points are located on the boundary surfaces and form the structured surface mesh.

The SIGMA preprocessor module has a graphical interface that allows creating a computational domain visually. The domain is constructed from a set of initial hexahedral blocks (primitives) by their combining and subsequent deformation. In the current version of SIGMA, such primitives can be a cube, cylinder, cylindrical and spherical segments. The deformation is performed by changing of coordinates of control points of the primitives. We can enter these coordinates or read from a file.

It is possible to load the incoming information about geometric shapes of bodies from the solid simulation software. In this case, it is used the STL geometry definition file format. The functions for generation of points in given sections and along lines between the two specified points on a triangulated surface are implemented for the construction of the structured mesh of control points on the imported surfaces (Fig. 1). Then the predetermined thickness curvilinear blocks are generated based on data about the normals in cells. Fig. 1 illustrates the process of importing the geometry from STL file, the created grid of control points, view of the surface cubic splines and the construction of the curvilinear blocks.



(a) STL geometry and grid of control points (b) surface spline (c) curvilinear blocks

Fig. 1. Import of geometry and domain construction

III. ADAPTIVE GRID GENERATION

A. *Introduction of Adaptive Coordinates*

A three-dimensional non-orthogonal coordinate system X^i, i=1,2,3 is introduced to generate an adaptive mesh. In this coordinate system, the boundaries of each curvilinear block are the coordinate surfaces. To go to these coordinates, we use the explicit form of algebraic transformation, which refers to the Lagrangian coordinate transformation of transfinite interpolation methods [1–3, 16]. There are functions that allow finding the Jacobian matrix of the transformation, neighboring nodes, tangent and normal vectors at the boundary points.

Generally, it is impossible to build an acceptable global transformation from a single block without singularities. Therefore, we use the following approach for the grid generation. A domain consists of hexagonal blocks which are continuously joined at the boundaries. The parameterizations and coordinate directions of the boundary surfaces of the blocks should be matched.

B. *Grid Concentration*

A preliminary transformation of an initial uniform mesh into a non-uniform one is introduced to control the distribution of nodes near the boundaries. We use the functions to grid concentration in adaptive coordinates [1], which are based on the numerical solution of equations with a small parameter in the highest derivatives. In addition to using the standard parameters of these functions, there is an option to manually input a domain of concentration (for example, the boundary layer thickness). In this case, the parameters of concentration functions are found by solving the equations in accordance with the inputted values.

C. *Generation of Block Structured Quasi-Continuous Grids*

It is impossible to ensure the continuity of adaptive coordinate lines for some computational domains; therefore, the introducing of additional internal discontinuities is necessary. The mesh nodes on these internal discontinuities are generated twice. The parameters of each of these nodes correspond to the parameters of the adaptive coordinate system in which it was generated. The physical coordinates of these nodes are the same. Each node has a link to its twin. There are different classes of geometry topologies with internal discontinuities in SIGMA preprocessor (Fig. 2). The corner nodes with the extreme values of some adaptive coordinates are separately selected among the twin nodes (Fig. 2 (a) and (b)). Also, we separately select the nodes wherein the adaptive coordinate lines are closed (Fig. 2 (c)).

Let the coordinate lines X^1 and X^2 of different adaptive coordinate systems (green and red on the Fig. 2) be perpendicular to an internal discontinuity. Let a derivative of a function at an ordinary node j be approximated by the central differences:

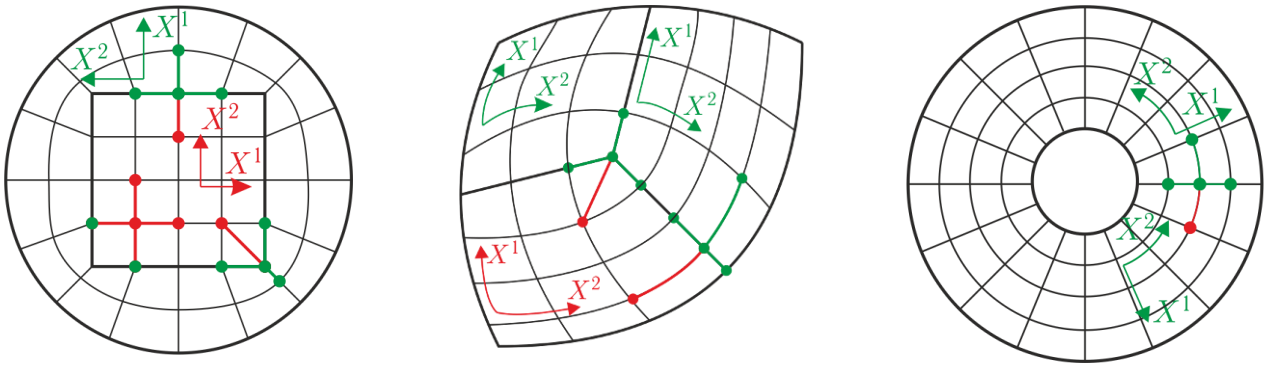
$$\frac{\partial f}{\partial X^1} \approx \frac{f_{j_R} - f_{j_L}}{X^1_{j_R} - X^1_{j_L}}. \tag{1}$$

Then the derivative at a node j on the internal discontinuity can be approximated for example as follows:

$$\frac{\partial f}{\partial X^1} \approx \frac{f_{j_R} - f_j}{X^1_{j_R} - X^1_j} + \frac{f_j - f_{j_D}}{X^2_{j_T} - X^2_{j_D}}. \tag{2}$$

Here j_R and j_L are the two neighbors of the node j in the direction X^1, j_D is the neighbor of the twin j_T of the node j in the direction X^2.

Results of a solution at the node j on the internal discontinuity correspond to only one of the two twin nodes. The second node is a subsidiary and its parameters are used only for the calculations of the difference approximations and searching for the appropriate neighbors.



(a) O-grid block (b) triangular block (a quarter of an O-grid block) (c) closed domain

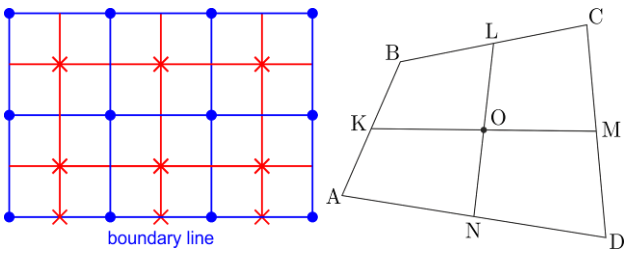
Fig. 2. Different classes of geometry topologies with internal discontinuities. It shows the cross sections along the adaptive coordinate X^3

D. *Generation of Tetrahedral Elements*

The block-structured hexahedral mesh can be subdivided into a consistent set of tetrahedra for the using of tetrahedral finite element or finite volume solvers. There are exactly two possible configurations that lead to subdivisions into five tetrahedra and 46 into six tetrahedra. Therefore, the tetrahedralization leads to a larger number of elements (from five to six times). The subdividing algorithms are similar to the algorithms described in [17–19]. The adaptive grids retain adaptation grid lines to borders of a geometry and allow one to obtain solutions of better quality than with grids generated by ordinary finite element mesh generators.

E. *Generation of Vertex-Centered Control Volumes*

The elements centered on the mesh nodes (vertex-centered volumes [20, 21]) are used by some finite volume methods to calculate the fluxes through the red facets between the corresponding values in the blue mesh nodes (Fig. 3 (a)). The construction of these elements is based on the already generated adaptive mesh by finding the midpoints of elements. Despite the fact that the application of the control volumes centered on the mesh nodes requires roughly 6 times as much storage as the case where the control volumes coincide with the initial mesh cells, but produces more accurate results in the boundary layer because of the finer mesh size in the near-wall domain. Half the control volume is used near the wall [22].



(a) vertex-centered control volumes (b) quadrilateral facet

Fig. 3. Generation of vertex-centered control volumes

The coordinates of a middle point O (Fig. 3 (b)) of a quadrilateral facet ABCD are calculated as follows:

$$\mathbf{r}_{\mathrm{O}} = \frac{1}{4}\,\mathbf{r}_{\mathrm{A}} + \mathbf{r}_{\mathrm{B}} + \mathbf{r}_{\mathrm{C}} + \mathbf{r}_{\mathrm{D}}\ . \tag{3}$$

A normal vector to the facet at the point O is calculated by the cross product of line segments joining the midpoints K,M and L,N of the two opposite edges of the facet (Fig. 3 (b)):

$$\mathbf{n}_{\mathrm{O}} = \frac{\mathrm{KM}\times\mathrm{LN}}{\left|\mathrm{KM}\times\mathrm{LN}\right|}. \tag{4}$$

The square of the facet is equal to:

$$S = \left|\mathrm{KM}\times\mathrm{LN}\right|. \tag{5}$$

The volume V of a hexahedral element is calculated by the divergence theorem:

$$\int_V \nabla\cdot\mathbf{a}\,dV = \sum_{k=1}^{6}\int_{S_k}\mathbf{a}\cdot\mathbf{n}\,dS_k. \tag{6}$$

The vector $\mathbf{a}$ is selected to be the position vector $\mathbf{a} = \mathbf{r} = \{x,y,z\}$. We use the midpoint rule for approximating integrals, so:

$$3V = \nabla\cdot\mathbf{a}\,V \approx \int_V \nabla\cdot\mathbf{a}\,dV = \sum_{k=1}^{6}\int_{S_k}\mathbf{a}\cdot\mathbf{n}\,dS_k \approx \sum_{k=1}^{6}\sum_{i=1}^{3} r_{\mathrm{O}ik}\,n_{\mathrm{O}ik}\,S_k. \tag{7}$$

Finally, we obtain:

$$V = \frac{1}{3}\sum_{k=1}^{6}\sum_{i=1}^{3} r_{\mathrm{O}ik}\,n_{\mathrm{O}ik}\,S_k. \tag{8}$$

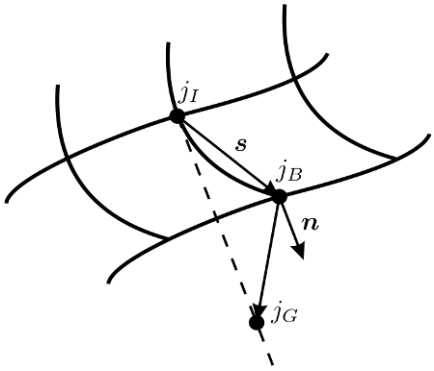


Fig. 4. Determining the location of a ghost node j_G

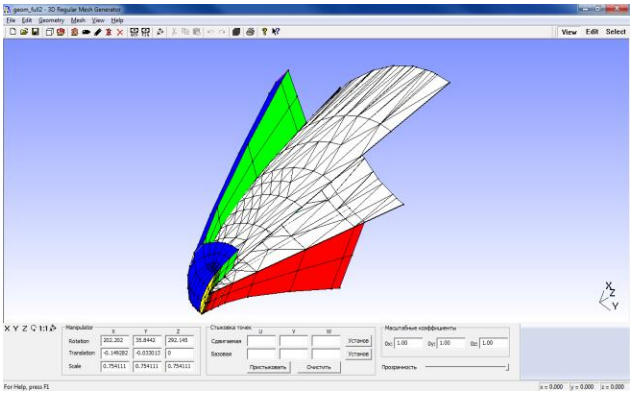


Fig. 5. SIGMA preprocessor interface

F. Generation of Ghost Nodes and Cells

Ghost nodes and cells can be introduced for the numerical implementation of boundary conditions. The initial data for the generation of a ghost node j_G are a boundary node j_B, its inner neighbor j_I, and the boundary element for which these nodes are the vertices. The calculation of the position vector of the ghost node $\mathbf{r}_{j_G}$ generally for a non-orthogonal adaptive grid is based on the following formulas (Fig. 4):

$$\begin{aligned} \mathbf{s} &= \mathbf{r}_{j_B} - \mathbf{r}_{j_I}, \\ c &= \mathbf{s} \cdot \mathbf{n}, \\ \mathbf{r}_{j_G} &= \mathbf{r}_{j_I} + 2c\mathbf{n}, \end{aligned} \tag{9}$$

where $\mathbf{n}$ is the unit outer normal vector at the boundary node j_B.

In the case of the generation of ghost nodes and cells for tetrahedral mesh, the parameter c is equal to one of the heights of a tetrahedral boundary element $h = 3V/S$, where V is the volume of the element, and S is the square of the facet that lies on the boundary.

Other parameters of the ghost nodes and cells built on their base are set equal to the corresponding boundary nodes and cells. A step along the adaptive coordinate between ghost and boundary nodes is also assumed to be equal to the appropriate step between the boundary node and its inner neighbor.

G. Calculation of Distances to Boundaries

Some models of physical phenomena require taking into account a distance from a mesh node to a boundary (e.g. some turbulence models use the distance to a wall). The preprocessor uses the following simplest algorithm to calculate the distance from the internal mesh node j_I to the boundary.

1) It is finding the closest boundary node j_B to the node j_I.

2) It is checking that the vector $\mathbf{v} = \mathbf{r}_{j_I} - \mathbf{r}_{j_B}$ entirely lies within the computational domain. For this to be done, the scalar product $d = \mathbf{v} \cdot \mathbf{n}$ is found, where $\mathbf{n}$ is the unit outer normal vector at the node j_B for finite difference grids or the unit outer normal vector averaged over all boundary facets where j_B is one of their vertices for finite volume or finite element grids. If $d > 0$ the vector $\mathbf{v}$ crosses the boundary and it is required to find the next closest node on the boundary.

3) The value of $|d|$ is chosen as the desired distance.

IV. SIGMA Preprocessor Interface

The SIGMA preprocessor that allows generating two-dimensional and three-dimensional adaptive grids has been developed based on the described algorithms. The preprocessor is written in C++ using the STL, Boost, OpenGL and OpenMP libraries. The preprocessor consists of the two parts: the first is a geometry editor, and the second is a mesh generator.

The geometry editor has a graphical interface (Fig. 5) that allows creating computational domains visually, selecting borders and subdomains for the definition of boundary and initial conditions and preparing data for a mesh generation.

When a user creates a primitive, the preprocessor requests its type, bounding box dimensions in adaptive and physical coordinate systems, as well as the number of control points for each coordinate direction. In accordance with the inputted values, the adaptive and physical coordinates of the control points are calculated. Later the user can edit these coordinates, as well as the primitive's dimensions. The preprocessor supports the ability to add and remove primitives, select a domain to zoom in on from the global view, translate and rotate of the domain. An inputted computational domain can be saved or loaded from a file.

Before input of the domain geometry in the preprocessor, it is convenient to prepare in advance the tables of geometrical objects and their attributes. The first table is a set of control points of the domain with the values of their physical coordinates. The second table is a list of the primitives of which the domain will consist, their types and dimensions as well as the number of control points attributable to each primitive. The number of primitives is determined mainly by the number of discontinuities or breaks of the computational domain borders, as well as the various initial data in the domain or boundary conditions on its borders. The third table

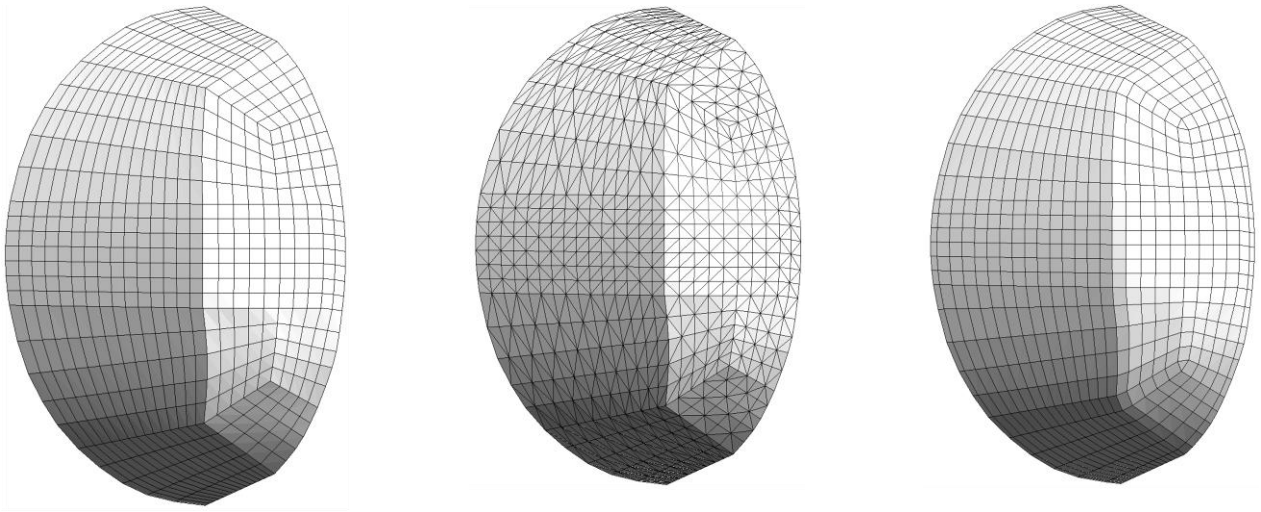
is the bounding box dimensions of domains where the distribution of mesh nodes should differ from the uniform (e.g. this may be a domain where the user wants to concentrate the mesh nodes).

The mesh generator is a cross-platform console application. The domain geometry data and parameters of the generated mesh prepared in the geometry editor are transferred to the mesh generator using the class serialization methods of the Boost library. The data of classes are serialized in the XML format and may be edited by the user in a text or XML editor. The mesh generation parameters are the domain of mesh generation, the total number of partitions and the number of partitions in certain subdomains of the adaptive coordinate system, the mesh type and formats, the mesh data that will be calculated. A two-dimensional mesh can be generated in a given section of the adaptive coordinate system for a constructed three-dimensional domain.

V. Examples of Grid Generation

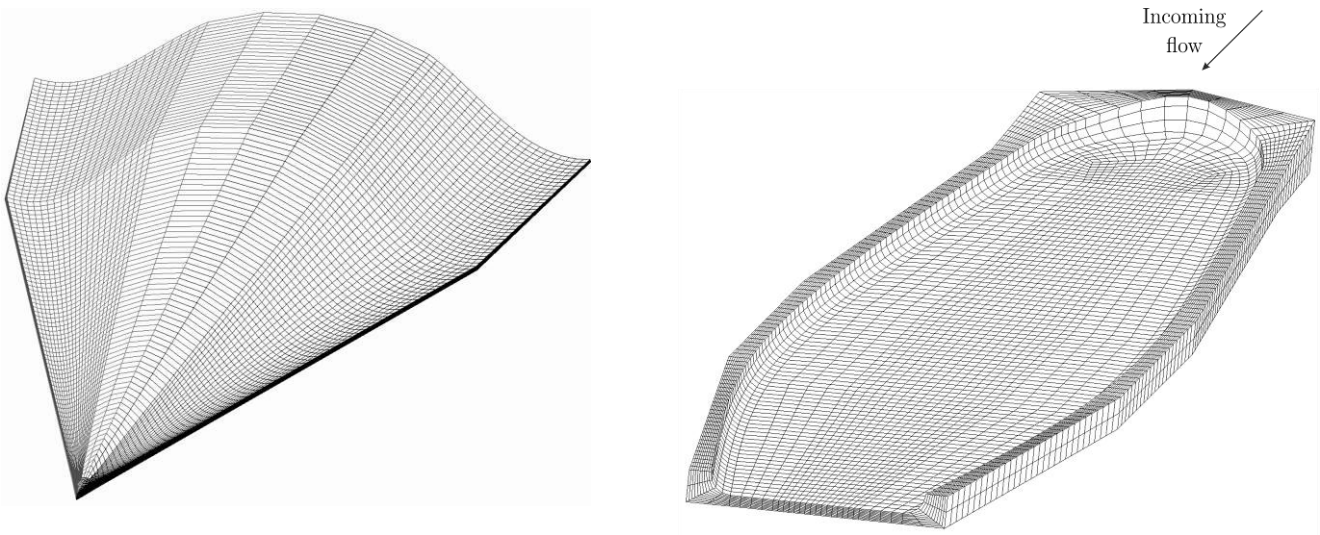
Fig. 6-7 show some results of the generation of three-dimensional adaptive grids. We have generated the sufficiently coarse grids for easy viewing of their structures.

Fig. 6 shows the different types of grids for the domain of external flow near a hemisphere. For the generation the block-structured hexahedral adaptive mesh, the nine curvilinear blocks were constructed by the method described in Sec. II. They are shown in Fig. 1 (c). The central block is located along the equator of the hemisphere. Two triangular O-grid blocks each consisting of the three common blocks are located on the forebody portion of the hemisphere above and below the central block. And the two remaining blocks are located behind the corresponding triangular blocks at the top and bottom of the central block. Fig. 6 (b)-(c) shows the possibility of converting the generated block structured hexahedral adaptive mesh into the finite element or finite volume grids.



(a) block structured mesh (b) tetrahedral mesh (c) vertex-centered control volumes

Fig. 6. Different types of grids for the domain of the external flow near a hemisphere



(a) mesh for the surface of the high-speed aircraft of Falcon HTV-2 [23] (b) mesh for the domain of the gap between a wheel and a body of an aircraft

Fig. 7. Examples of generated block structured adaptive grids

Generally, each hexahedron is divided into five tetrahedra. In exceptional cases, if it is not possible to dock the adjacent elements by means of the division into five tetrahedra, the division into six tetrahedra may apply. An essential feature of the resulting tetrahedral mesh in comparison with those obtained by means of the most grid generators both open-source and commercial is that the grid lines are adjusted with the flow in the whole computational domain (see Fig. 6 (b)). It makes it possible to select more precisely various features of the flow, such as shock waves, contact discontinuities and the depression waves. Fig. 6 (c) shows the red cells of the mesh consisting of vertex-centered control volumes (see subsections III.E). Thus, these red cells are circumscribed around the nodes of the initial mesh shown in Fig. 6 (a). It is seen that for the corner nodes of O-grid blocks (see subsections III.C) a hexahedral element is degenerated into a triangular prism. However, a calculation with such elements is not a significant problem for finite volume methods. It is evident also that the vertex-centered finite volume elements are smaller in size than others near the boundaries.

Fig. 7 shows the block-structured adaptive grids for the real complex curvilinear computational domains: the outer surface of the high-speed aircraft of the Falcon HTV-2 type [23] and for the domain of the gap between a wheel and a body of an aircraft. The last domain has a local expansion near the flow inlet boundary to capture the shock wave. In such gaps, the mesh should be sufficiently fine to reach the necessary accuracy of the solution of temperature and heat flux fields.

An essential feature of all shown grids is that they were generated from not only one curvilinear blocks, but also their combination. It makes it possible to use them with finite-difference methods for complex curvilinear computational domains.

VI. Conclusions

The computer technologies to generate block-structured adaptive grids have been developed. They include the methods of computer-aided geometric design of domains, grid generation techniques, approaches to the grid concentration, methods of working with block structured quasi-continuous grids for the certain types of curved domains, aspects of subdivision block-structured adaptive grids into tetrahedral elements and vertex-centered control volumes, the algorithms for generation of ghost nodes and cells and calculation of distances from internal mesh nodes to domain boundaries. The generated grids can be two-dimensional, two-dimensional axisymmetric (in cylindrical coordinates) or three-dimensional. They are applied for the finite difference, finite element or finite volume methods. The developed computer technologies have been implemented in the SIGMA preprocessor that includes the modules of generation of a wide range of three-dimensional geometric shapes, setting properties, parameters and initial conditions and adaptive mesh generator.

The main advantages of the proposed technologies of adaptive grid generation in comparison with other approaches are the structuring and hexahedral shape of the mesh cells in the complex curvilinear computational domains, the applicability of the mesh for finite difference methods, the adjusting with the flow grid lines in the entire domain.

References

[1] V.D. Liseikin, Grid Generation Methods. Springer-Verlag, Heidelberg, 1999.

[2] R. Shneiders, “Refining quadrilateral and hexahedral element meshes”, Proceedings of 5th International Conference on Numerical Field Simulations, pp. 699-708, 1996.

[3] M. Farrashkhalvat, J.P. Miles, Basic Structured Grid Generation. Butterworth-Heinemann, Oxford, 2003.

[4] C. Geuzaine, J.-F. Remacle, “Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities”, International Journal for Numerical Methods in Engineering, vol. 79, no. 11, pp. 1309-1331, 2009.

[5] H. Si, “TetGen, a Delaunay-based quality tetrahedral mesh generator”, ACM Trans. on Mathematical Software, vol. 41, no. 2, Article no. 11, February 2015.

[6] S.V. Reznik, K.V. Mikhailovskiy, P.V. Prosuntsov, “Modeling the heat and mass transfer in the pores of the thermal protection carbon-carbon frame during the gas-phase deposition of silicon carbide”, MATEC Web Conf., vol 92, Article no. 01075, 2017.

[7] V.V. Kulik, A.N. Parkin, E.S. Navasardyan, “Numerical modeling procedure for micromachined cryogenic cooler elements using ANSYS Fluent software and viscous flow in a small-diameter channel with heat transfer as an example”, Chem. Petrol. Eng., vol. 52, no. 7-8, pp. 531-538, November 2016.

[8] I.K. Marchevskii, V.V. Puzikova, “Numerical simulation of the flow around two circular airfoils positioned across the stream using the LS-STAG method”, J. Mach. Manuf. Reliab., vol. 46, no. 2, pp. 114-119, 2017.

[9] V.V. Shumaev, V.V. Kuzenov, “Development of the numerical model for evaluating the temperature field and thermal stresses in structural elements of aircrafts”, J. Phys.: Conf. Series, vol. 891, no. 1, Article no. 012311, 2017.

[10] V.V. Kuzenov, A.O. Dobrynina, V V Shumaev, “Calculating processes of laminar and turbulent heat transfer around the elements of the aircraft”, J. Phys.: Conf. Series, vol. 980, no. 1, Article no. 012023, 2018.

[11] V.V. Kuzenov, S.V. Ryzhkov, “Radiation-hydrodynamic modeling of the contact boundary of the plasma target placed in an external magnetic field”, Appl. Phys., no. 3, pp. 26-30, 2014.

[12] R. Schneiders, “A grid-based algorithm for the generation of hexahedral element meshes”, Engineering with Computers, vol. 12, pp. 168-177, 1996.

[13] J.F. Thompson, B.K. Soni, N.P. Weatherill, Handbook of grid generation, CRC Press, Boca Raton, 1999.

[14] Yu.I. Dimitrienko, M.N. Koryakov, A.A. Zakharov, “Application of finite difference TVD methods in hypersonic aerodynamics”, LNCS, vol. 9045, pp. 161-168, 2015.

[15] Yu.I. Dimitrienko, M.N. Koryakov, A.A. Zakharov, “Computational simulation of conjugated problem of external aerodynamics and internal heat and mass transfer in high-speed aircraft composite constructions”, International Journal of Mechanical Engineering and Robotics Research, vol. 6, no. 1, pp. 58-64, 2017.

[16] P.R. Eiseman, “Control point grid generation”, Computers Math. Applic, vol. 24, no. 5/6, pp. 57-67, 1992.

[17] G. Albertelli, R.A. Crawfis, “Efficient subdivision of finite-element datasets into consistent tetrahedra”, in IEEE Visualization, November 1997, R. Yagel, H. Hagen, Eds. Phoenix, AZ, pp. 213-220.

[18] J. Dompierre, P. Labbé, M.G. Vallet, R. Camarero, “How to subdivide pyramids, prisms and hexahedra into tetrahedra”, in 8th International Meshing Roundtable. Lake Tahoe, Californie, 10–13 October 1999.

[19] S.S. Bahrainian, “Construction of hexahedral block topology and its decomposition to generate initial tetrahedral grids for aerodynamic application”, JAST, vol. 5, no. 2, pp. 81-90, 2008.

[20] T.J. Barth, “Aspects of unstructured grids and finite-volume solvers for the Euler and Navier-Stokes equations”, in VKI Lecture Series, no. 1994-05. Belgium: Von Karman Institute for Fluid Dynamics, 1994.

[21] A. Jameson, D. Mavripils, “Finite volume solution of the two-dimensional Euler equations on a regular triangular mesh”, AIAA Paper, no. 85-0435, 1985.

[22] K.N. Volkov, “Unstructured-grid finite-volume discretization of the Navier-Stokes equations based on high-resolution difference schemes”, Computational Mathematics and Mathematical Physics, vol. 48, no. 7, pp. 1181-1202, 2008.

[23] http://flagman.top/ru/about-war/lockheeds-marilyn-hypersonic-weapons/