Molecular dynamics-based unstructured grid generation method for aerodynamic applications

A.L. Zheleznyakova, S.T. Surzhikov *

Institute for Problems in Mechanics Russian Academy of Sciences, Vernadskogo 101/1, 119526 Moscow, Russia

HIGHLIGHTS

• Triangular mesh generation based on the molecular dynamics method is developed.
• Mesh adaptation technology for molecular dynamics simulation is suggested.
• Suggested algorithm works identically for 1D, 2D and 3D problems.
• Computational grids have been tested for hypersonic flow past unmanned airplane X-43.

ABSTRACT

A new approach to triangular mesh generation based on the molecular dynamics method is proposed. Mesh nodes are considered as interacting particles. After the node placement by molecular dynamics simulation, well-shaped triangles or tetrahedra can be created after connecting the nodes by Delaunay triangulation or tetrahedrization. Some examples are considered in order to illustrate the method's ability to generate a mesh for an aircraft with a complicated boundary. Mesh adaptation technology for molecular dynamics simulation is presented.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

The concept of a mesh as a discretization of space has been associated with computational methods since the first attempts to obtain numerical solutions of partial differential equations. Establishing a suitable mesh is a rather tedious exercise and a minor part of the computational effort involved in solving partial differential equations by either a finite difference or finite element method.

Although a mesh is a prerequisite for numerical computation in many areas of science and engineering, it was computational fluid dynamics, and computational aerodynamics in particular, that were the key drivers in stimulating the development of reliable and efficient mesh generators.

The accuracy of the flow simulations based on the Reynolds averaged Navier Stokes (RANS) equations over the complete aircraft is influenced by many factors: discretization formulas for the flow equations which introduce only minimal amounts of artificial dissipation, a turbulence model that captures the boundary layer behavior, a flow solver that will achieve a high degree of convergence, and, finally, a mesh that has good quality and is sufficiently fine to permit a good approximation of the flow solution throughout the region of interest.

Unstructured grids are widely used in computational aerodynamics in spite of the fact that predictions based on the RANS equations using a structured (i.e. hexahedral) mesh are generally higher, and closer to the experimental values, than comparable predictions on an unstructured mesh of tetrahedra.

Unstructured grids have significant advantages:

• The advantage of unstructured grid methods is that they have the ability to be automated to a large degree and, therefore, require little user time or effort. The user need not worry about laying out block structure or connections. Additionally, unstructured grid methods are well suited to inexperienced users because they require little user input and will generate a valid mesh under most circumstances. Unstructured methods also enable the solution of very large and detailed problems in a relatively short period of time. Grid generation times are usually measured in minutes or hours.

* Corresponding author. Tel.: +7 495 4338298.
E-mail address: surg@ipmnet.ru (S.T. Surzhikov).

0010-4655/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cpc.2013.07.013

• Unstructured grids have the advantage of generality in that they can be made to conform to nearly any desired geometry. Grids have a great flexibility in handling complex geometries.
• These grids tend to be easier to generate than structured or composite multiblock grids.
• The multiblock structured mesh generation technique requires definition and discretization of the block interfaces. A good graphics user interface (GUI) can ease the burden of creating the block structure, but it can still be a largely manual task and time consuming process. Therefore multiblock grid generation times are usually measured in days if not weeks. More significantly, the difficulty of automating this process inhibits the use of composite block methods for solving time evolving problems, or other situations such as automatic design, where the boundary shape is changing.

This generality, however, comes with a price:
• The major drawback of unstructured grids is the lack of user control when laying out the mesh. Typically any user involvement is limited to the boundaries of the mesh with the meshing automatically filling the interior.
• Triangle and tetrahedral elements have the problem that they do not stretch or twist well; therefore, the grid is limited to being largely isotropic, i.e. all the elements have roughly the same size and shape (great difference in neighboring element sizes can increase numerical approximation errors). This is a major problem: when trying to refine the grid in a local area, often the entire grid must be made much finer in order to get the point densities required locally.
• Another drawback of the methods is their reliance on good CAD data. Most meshing failures are due to some (possibly microscopic) error in the CAD model.
• Unstructured grids require more information to be stored and recovered than structured grids (e.g., the neighbor connectivity list). Unstructured flow solvers typically require more memory and have longer execution times than structured grid solvers on a similar mesh.
• Post-processing the solution on an unstructured mesh requires powerful tools for interpolating the results onto planes and surfaces of rotation for easier viewing.
• Unstructured grid methods generally have poorer numerical accuracy. For example, it is difficult to construct approximations that maintain an accurate propagation of one-dimensional flow disturbances because tetrahedral grid elements have no parallel faces.

There are several essentially different approaches to generating triangular or tetrahedral meshes. The most common of these are the moving front technique [1,2] and Delaunay based methods [3–12].

The moving front technique starts from a prescribed boundary definition (set of edges in 2D, set of triangular faces in 3D) which remains intact throughout the mesh generation process. The boundary triangulation is regarded as a front on which a new layer of elements is built. The original front triangles become interior faces of the mesh and a new set of front faces is created, a process that continues until the entire domain has been filled.

The majority of Delaunay based methods exploit an incremental algorithm that starts with an initial triangulation of just a few points [7,8]. The complete triangulation is generated by introducing points and locally reconstructing the triangulation after each point insertion.

In contrast to the approach of incrementally inserting nodes one by one into an existing mesh, some mesh generation methods place all the nodes first before connecting them into a mesh. Nodes are first distributed into optimal locations to cover the whole domain, and then connected into mesh. In the bubble meshing method [13], Shimada and Gossard place nodes at centers of a set of packed spheres or bubbles. An inter-bubble force function is defined between adjacent bubbles, a force balancing configuration of the nodes is found by performing dynamic simulation. After node placement, the method connects the nodes into well-shaped mesh elements by constrained Delaunay triangulation or tetrahedrization. In [14] Persson and Strang treat a simplex mesh as a truss structure and mesh points as nodes of the truss. A linear force-displacement function is defined for each pair of nodes connected by a bar (or edge). Nodes are gradually moved to optimal locations by iteratively solving for force equilibrium. At each iteration, the node set is re-triangulated using the Delaunay triangulation algorithm in order to decide the edges. In the end, a high quality triangular mesh is obtained.

In study [15] surface or volume geometries to be meshed are treated as atomic systems, and mesh nodes are considered as interacting particles. By minimizing system potential energy with Monte Carlo simulation, particles are placed into a near optimal configuration. Well-shaped triangles or tetrahedra can then be created after connecting the nodes by constrained Delaunay triangulation or tetrahedrization.

The isosurface stuffing algorithm is presented in [16] that fills an iso-surface with a uniformly sized tetrahedral mesh whose dihedral angles are bounded between 10.7° and 164.8°. The algorithm is fast, numerically robust, and easy to implement because it generates tetrahedra from a small set of pre-computed stencils. A variant of the algorithm creates a mesh with internal grading: on the boundary, where high resolution is generally desired, the elements are fine and uniformly sized, but in the interior they may be coarser and vary in size. The isosurface stuffing method is a powerful tool for dynamic fluid simulation, large-deformation mechanics, and applications that require interactive remeshing or use objects defined by smooth implicit surfaces. The algorithm is perfect for robust remeshing in physically-based animation at interactive rates.

In paper [17] Nochetto and Walker presented a hybrid variational front tracking-level set method for generating 2D unstructured triangular meshes that undergo large deformations and topological changes in an automatic way. The method is based on the idea of combining the level set and front-tracking methods. The level set method advects a scalar field function whose zero level set represents the interface and has the advantage of being completely Eulerian and can automatically handle topological changes. The front-tracking method uses an explicit representation of the interface, such as an interface mesh or marker particles to “track” the interface. This hybrid approach combines the accuracy of the explicit mesh methods with the ease of topological transformation to generate meshes of arbitrary domains. In addition, authors introduce a shape optimization approach for ensuring mesh conformity. The method is advantageous for Arbitrary Lagrangian–Eulerian (ALE) type methods and directly allows for using a variational formulation of the physics being modeled and simulated, including the ability to account for important geometric information in the model (such as for surface tension driven flow). The meshing procedure is not required at every time-step and the level set update is only needed during a topological change. Hence, the method does not significantly affect the computational cost.

Our method is not so fast due to the fact that molecular dynamics simulations are inherently time-consuming and computationally expensive and is not applicable for problems requiring interactive remeshing. The molecular dynamics method was developed for mesh generation mainly for aerodynamic problems which do not need frequent remeshing. However, our method is very convenient for relatively rapid local remeshing which is used to solve aerodynamic shape optimization problems. Moreover, in
the 2D case, the algorithm produces triangles that are almost equilateral (the ratio between the radius of the largest circumscribed circle times two and the smallest circumscribed circle is close to 1). This is a desirable property when solving PDEs with the finite volume or finite element methods. Upper bounds on the errors depend only on the smallest angle in the mesh, and if all angles are close to 60, good numerical results are achieved.

In three dimensions there is a problem to generate well-shaped Delaunay tetrahedralization of the well-spaced point set. Although mesh created using our method consists of almost equilateral tetrahedral elements, some sliver shaped tetrahedra may occur in the mesh (a sliver is a tetrahedron whose vertices are almost coplanar) [18, 19].

The molecular dynamics method enables us to completely control the size of interior elements and allows us to specify the desired cell size depending on the distance to the wall for adaptive mesh. This gives ability to guarantee the quality of not only the boundary but also the interior elements. This is an important property because great difference in neighboring element sizes and shapes can increase numerical approximation errors especially when trying to refine the grid in a local area.

In our preceding paper [20] we briefly reviewed the key ideas of the molecular dynamics method. The present paper proposes a significantly improved molecular dynamics algorithm and gives a more detailed description of main concepts. In addition, this work expands the applicability of the molecular dynamics technique.

2. Model of particle interactions

The main idea of our method is that mesh nodes are considered as similarly charged particles with charge \( q_i \) and mass \( m_i \). Charged particles interact with each other. Interacting forces are electrical in nature. Fig. 1 shows a schematic that illustrates the interaction between two particles.

Let us consider a system of \( N \) interacting similarly charged particles. The magnitude of the electrostatic force of repulsion \( \vec{F}_{ij} \) acting between the \( i \)-th and the \( j \)-th particles is directly proportional to the product of the magnitudes of each of the charges and inversely proportional to the \( k \)-th power of the distance between the two particles (Eq. (1)). The direction of the force given by the unit vector \( \vec{R}_{ij} \), parallel with line joining the \( i \)-th particle to the \( j \)-th particle. The vector form for electrostatic force of repulsion acting between any two particles in the system can be represented as follows:

\[
\vec{F}_{ij} = \frac{-q_i q_j}{|\vec{R}_{ij}|^k} \cdot \frac{\vec{R}_{ij}}{|\vec{R}_{ij}|}, \quad k \geq 1, \quad |\vec{R}_{ij}| = l,
\]

where \( l \) is the distance between two charges.

The resultant of forces acting on the \( i \)-th particle is defined as sum of all the internal forces exerted by all the particles of the system on the particle \( i \):

\[
\vec{F}_i = \sum_{j=1}^{N} \vec{F}_{ij}.
\]

Besides the interacting forces the particles experience a medium resistance \( \vec{F}_{r,i} \). Drag forces depend on particle velocity and act in a direction opposite to the velocity vector.

3. The medium resistance

Drag forces play a significant role in a simulation. Various power-law velocity dependences are used to determine the resistance force:

\[
\vec{F}_{r,i} = -K \left( \frac{\vec{v}_i}{|\vec{v}_i|} \right)^{m} \vec{v}_i, \quad m \geq 2.
\]

The temporal behavior of the system of interacting particles in the absence of drag for the planar region is shown in Fig. 2. The particles move freely past one another throughout the solution domain. In the three-dimensional case the process develops similarly. In the absence of the resistance interacting particles are not able to find their stable positions even for an arbitrarily long period of time. The interaction process resembles a chaotic Brownian motion. If you add resistance, the pattern of interaction will change dramatically. Particles cease to move randomly, and come to equilibrium within a relatively short period of time. Each particle occupies a stable equilibrium position (particles perform small oscillations around their equilibrium positions). Furthermore, the distribution of particles in the computational domain becomes highly ordered (Fig. 3).

4. Particle interactions with the boundaries of the computational domain

One of the most important aspects to automatic and semi-automatic mesh generation is the possibility of using CAD geometry. In our method we use standard CAD geometry representations, as initial surface definition (Fig. 4). Domain boundaries are defined using lists of polygons and facet normals. The algorithm works as follows: for every particle at each time-step of the simulation we find the nearest boundary facet and check the condition that the particle has not left the computational domain. For each boundary polygon we choose the direction of the facet normal \((n_{xj}, n_{yj}, n_{zj})\) into the computational domain. If the scalar product of the internal normal vector with a vector from one of the polygon’s vertices (polygon’s center) to movable particle \( \vec{r}_{ij} \) is greater than zero, the particle is inside the domain. If the value of dot product \( \vec{n}_i \) is less than zero, the additional force acts on the particle. The direction of the force is defined by the vector, parallel with internal polygon normal \( \vec{n}_i \). Thus the particles that have left the domain immediately move to their projection points on the closest facets. These particles become the boundary nodes.

The interacting force acting on a movable “external” particle \( i \) with nearest boundary facet \( j \) obeys the equation:

\[
\vec{F}_{i_{wall}} = C_{wall} \frac{q_i}{|\vec{n}_j|^{m}} \cdot \frac{\vec{n}_j}{|\vec{n}_j|}, \quad m \geq 1.
\]

Fig. 4 shows the interaction between the movable interior particle and domain boundary.
The proposed interaction model does not allow any particles to leave the computational domain. The algorithm enables us to produce a surface mesh without meshing the edges first, and volume mesh without meshing the boundary surfaces first. In this way the bounding edges and surfaces are meshed at the same time as the volume. As will be shown below, this technique allows us to get a high quality mesh on the domain boundaries. Though, there is little probability of occurrence of surface defects during process of 3D node distribution in the computational domain. In the case of bounding surfaces, demanding high resolution mesh, the volume mesh can be obtained after previous distribution of boundary nodes by the same MD method (the algorithm works identical for 1D, 2D and 3D problems). These boundary nodes are considered as fixed input boundary particles which do not move during molecular dynamics simulation and correspond to the desired positions of the external grid nodes.
5. The system energy

As stated above, under the action of the drag forces interacting particle system comes to equilibrium within a relatively short period of time. The molecular dynamics method considered in this paper suggests the integration of the differential equations of motion at each time step. The total number of time steps necessary to achieve the equilibrium is not known in advance. Consequently, the efficient criterion for obtaining desired equilibrium particle distribution is necessary. In the two-dimensional case an equilibrium state can be judged by the degree of ordering of nodes (Fig. 3). For planar domain, the time when the equilibrium is obtained can be determined on the basis of a visual assessment.

The situation in 3D is much more complicated and it is almost impossible to check a mesh with several million points around a complete aircraft and decide whether the distribution of nodes is acceptable.

Let us consider a collection of particles interacting with each other. In a stable configuration of particles, the system should have minimal potential energy.

At the initial time system has the maximum potential energy corresponding to random distribution of particles. During the simulation, the system energy declines when increasing the number of time steps. The positions of the particles determine the potential energy of the system.

Considering the pair potential only, the system potential energy $U$ can be represented as

$$U = \sum_i \sum_{j<i} U_{ij} \left( \frac{1}{|\vec{r}_{ij}|} \right), \quad |\vec{r}_{ij}| = l,$$

(5)

where $l$ is the distance between a pair of particles $i$ and $j$, and $U_{ij} \left( \frac{1}{|\vec{r}_{ij}|} \right)$ is the pair potential.

A function describing the interaction energy can be obtained from the expression (1) for the force of repulsion between two charged particles.

$$F_{ij} = -C \frac{q_i q_j}{|\vec{r}_{ij}|^2} \cdot \frac{\vec{r}_{ij}}{|\vec{r}_{ij}|},$$

(6)

$$U_{ij} = \frac{C}{m - 1} \cdot \frac{q_i q_j}{|\vec{r}_{ij}|^{m-1}} + A.$$

(7)

The interaction potential between the particles is always positive, because only the force of repulsion acts between them. Function describing the interaction energy between individual particles has no minimum and smoothly decreases with increasing distance, asymptotically approaching zero. In spite of this, the total interaction potential energy of the system of particles $U$ has a minimum.

Let us look at a concrete example how system state changes over time. A system consists of $N = 300$ particles moving in a cubic computational domain. Fig. 5 shows the change in potential energy of the system over time. The abscissa indicates the number of time steps. The time step size is 0.01. At the initial time a set of nodes is randomly distributed in the region to be meshed. Time $t = 0$ corresponds to the maximum value of the potential energy. The system energy declines rapidly, and within fifteen hundred of time steps reaches a near equilibrium. Since then we can assume that the optimal distribution of particles is reached.

Let us also follow the evolution of the kinetic energy of a system of particles. The total kinetic energy of a system at any instant in time is the sum of the kinetic energies of the particles it contains:

$$E = \sum_i E_i = \sum_i \frac{m_i v_i^2}{2}.$$

(8)

Fig. 6 shows the graph of the kinetic energy of a system of 300 identical interacting particles. The abscissa indicates the number of time steps. The initial velocity of the particles is equal to zero. In the next moment particles begin to move under the influence of the forces acting on the system. The potential energy of interaction between particles is transformed into kinetic energy of motion. The kinetic energy of a system reaches its maximum value at $t = 0$. The total kinetic energy also decreases with time, asymptotically approaching a small but nonzero value, because of the forces of resistance acting in the system. This value corresponds to the kinetic energy of particle oscillations around their equilibrium positions. As noted earlier, in the absence of the resistance, interacting particles are not able to find their stable positions. Fig. 6 shows, also, the velocity–time graph of an individual particle. The graph indicates the presence of minor oscillations of a particle in equilibrium.

It should be noted that the system of interacting particles has a high sensitivity to the model parameters variation. A careful design
for a combination of physical parameters is needed in order to avoid instabilities and slow convergence.

A 90 times reduction of the coefficient $C$ in the expression (1) for the interaction force leads to significantly reduce the rate of convergence. Figs. 7–9 show the time dependence of the potential and kinetic energy as well as the time dependence of the rate of single particle for a system of 1000 particles for different values of coefficient $C$.

As seen from the figures the potential energy experiences significant fluctuations and the rate of convergence considerably decreases. The kinetic energy and particle velocity also oscillate considerably. Once the system reaches equilibrium, the particle oscillations remain significant.

Increasing the density of particles results in an increase of velocities and energies and has no effect on the stability and speed of convergence. Behavior of temporal dependences of these values does not change with increasing number of particles in the computational domain.

Careful selection of parameters (choice of interaction forces and resistance force dependences, constants and exponents in these expressions) was carried out during the method development in order to achieve stability and fast convergence and to obtain a high quality mesh. The values of the parameters were determined through numerous numerical experiments. It is not necessary to choose these parameters at each simulation (although such possibilities exist). The user needs to specify: the geometry of computational domain, which can be represented by CAD data; the number of movable particles; the desired distance between points (if the user needs to obtain a uniform mesh of equally sized elements); the degree of nodes condensation to a given bounding surface.

Fig. 6. The graph of the kinetic energy of a system of 300 particles and the velocity–time graph of an individual particle.

Fig. 7. The time dependence of the potential energy for a system of 1000 particles for different values of coefficient $C$: (1) $C_1 = C/90$; (2) $C_2 = C/10$; (3) $C_3 = C$. 

Fig. 8. The time dependence of the kinetic energy for a system of 1000 particles for different values of coefficient $C$: (1) $C_1 = C/90$; (2) $C_2 = C/10$; (3) $C_3 = C$.

Fig. 9. The time dependence of the rate of single particle for a system of 1000 particles for different values of coefficient $C$: (1) $C_1 = C/90$; (2) $C_2 = C/10$; (3) $C_3 = C$.

depending on the distance to the surface (if the user needs to produce a non-uniform mesh). For complex domain and adaptive mesh, it can be difficult to determine how many nodes need to be distributed in the system. If there are slightly more nodes than required, the element size in the final mesh will be a little smaller than a user specified value. If there are slightly less nodes than required, the element size in the final mesh will be a little larger. In both cases, the quality of the mesh does not suffer.

6. Integrating the motion equations

Let us assume that all the forces acting on a particle are known (Figs. 1 and 4). As discussed in the previous section, these forces come from interactions with other mobile particles, from interactions with boundary, from resistance force acting on particle. We can thus compute the total force acting on particle $i$ at time $t$:

$$ F_{\Sigma i}(t) = F_i(t) + F_{fr,i}(t) + F_{wall,i}(t), \quad i = 1, \ldots, N, \quad (9) $$

where

$$ F_i = \sum_{j=1}^{N} F_{ij}, \quad (10) $$

- the resultant of forces exerted on the $i$-th particle by the remaining mobile particles (where $N$ is the total number of particles);
- the resistance force $F_{fr,i}$ is defined by the Eq. (3).

We can then use Newton’s equations of motion for determining the motion characteristics of the $i$-particle (velocity, acceleration and position in space at each time step $t$).

Newton’s law of motion for the system of $N$ material points can be written as follows:

$$ m_i \frac{d\vec{v}_i}{dt} = \vec{F}_i + \vec{F}_{fr,i} + \vec{F}_{wall,i}, \quad i = 1, \ldots, N. \quad (11) $$

Differential equations for the velocity close the original system of $N$ equations. As a result, we get

$$ \begin{cases} 
    m_i \frac{d\vec{v}_i}{dt} = \vec{F}_i + \vec{F}_{fr,i} + \vec{F}_{wall,i}, & i = 1, \ldots, N, \\
    \vec{v}_i = \frac{d\vec{r}_i}{dt}, & i = 1, \ldots, N. \quad (12) 
\end{cases} $$

In the projection into axes:

$$ \begin{cases} 
    m_i \frac{d\vec{v}_{ai}}{dt} = \vec{F}_{ai} + \vec{F}_{fr,ai} + \vec{F}_{wall,ai}, & i = 1, \ldots, N, \\
    \vec{v}_{ai} = \frac{d\vec{r}_{ai}}{dt}, & \alpha = x, y, z. \quad (13) 
\end{cases} $$

Fig. 10. Meshes with different number of nodes for a circle produced by our method and the GAMBIT.

The well-known Verlet algorithm is applied to numerical integration of equation of motion [21,22]. Integration of these equations goes as follows:

1. Define the initial number \( N \) of simulated particles, assignment of their mass \( m_i \) and charge \( q_i \). Generate random values for coordinates of the mobile particles \( x_i, y_i, z_i \) within the region to be meshed.

2. At time \( t = 0 \) the particles are at rest. The initial velocities of the particles are equal to zero (\( \nu_{x_i} = 0, \nu_{y_i} = 0, \nu_{z_i} = 0 \)).

3. At time \( t \neq 0 \) the resultant of the forces \( F_{\Sigma \alpha} \) yields an acceleration \( a_{\alpha i} \) for particle \( i \):

\[
a_{\alpha i}(t) = \frac{1}{m_i} F_{\Sigma \alpha i}(t), \quad \alpha = x, y, z. \tag{14}\]

4. From the current positions, velocities and accelerations we can then predict new positions and velocities at time \( t + \delta t \):

\[
\nu_{\alpha i}(t + \delta t) = \nu_{\alpha i}(t) + a_{\alpha i}(t) \delta t,
\quad r_{\alpha i}(t + \delta t) = r_{\alpha i}(t) + \nu_{\alpha i}(t + \delta t) \delta t,
\quad \alpha = x, y, z. \tag{15}\]

5. Further the algorithm uses positions and velocities at time \( t + \delta t \) to calculate forces \( F_{\alpha i}(t + \delta t) \) and accelerations \( a_{\alpha i}(t + \delta t) \) at time \( t + \delta t \).

The molecular dynamics method provides a numerical solution of classical (Newtonian) equations of motion. Molecular dynamics of large systems requires significant computational resources including CPU time and memory.

The algorithm needs to store the coordinates and velocities of all particles at each time step during a simulation and update data from one time step to another. The forces and accelerations for a given time step can be derived from the coordinates and velocities. This technique has memory requirements which grow linearly with the number of particles. The required memory resources also depend on data types. The more nodes, the greater accuracy and resolution of data required.

The processing time depends on the size of the system (the number of particles) and the cutoff distance \( R_{\text{max}} \) for the long range interacting forces (such as is the case for Coulomb interactions between particles \( F_{ij} \sim 1/|\vec{r}_{ij}|^2 \)). If all particles interact pairwise the direct of this general \( N \)-particle problem will require the calculation of \( N(N-1) \) forces, i.e., the computational cost will grow as the square of the number of particles \( O(N^2) \).

If we use short-range forces \( F_{ij} \sim 1/|\vec{r}_{ij}|^k, \quad k > 3 \), or if the force cutoff distances are relatively small, the computational cost only grows linearly with the number of particles \( O(N) \). Thus, if the range of the potential is much shorter than the size of the computational domain, each particle only interacts with a limited number of neighbors.

A molecular dynamics algorithm described in this paper involves the following most computationally expensive steps:

- Calculate the particle–particle interactions (for each time step).
- Check if particles belong to the computational domain (at each time step).
- Modeling of particle–wall interactions (for each time step).
- Calculate of motion parameters and the trajectories (for each time step).
- Connect the nodes into a mesh by Delaunay triangulation or tetrahedrization (after the final iteration).

Computational cost of molecular dynamics simulation is measured by time it takes to run a molecular dynamics simulation, divided by the number of particles and the total number of time steps.
steps. The computation times (in ms/particle/step) of the different stages of simulation for fifteen hundred of time steps and different numbers of particles are given in Table 1. The molecular dynamics calculations were performed on an Intel Core 2 Duo 1.86 GHz CPU computer.

The Table 1 shows that most of the computational effort in the molecular dynamics algorithm is spent calculating particle–particle interaction forces. Molecular dynamics simulations are inherently time-consuming and computationally expensive due to a detailed force evaluation at each iteration step and the small iteration time step. Although the use of cutoff radius, verlet neighbor list method, periodic boundary conditions and short range-forces can greatly reduce the computational cost of the simulation, it is only possible to decrease the $O(N^2)$ algorithm to $O(N)$ [23].

It should be noted that the number of iterations required for convergence does not grow with the number of particles. Since the mesh nodes are connected using Delaunay triangulation (tetrahedrization) after the node placement, the time complexity of the method does not depend on the computational cost of the Delaunay triangulation.

### Table 1

<table>
<thead>
<tr>
<th>Computational Costs of the Different Stages of the Algorithm</th>
<th>1000</th>
<th>2000</th>
<th>4000</th>
<th>8000</th>
<th>16 000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Particle–particle interactions</td>
<td>8.175E−02</td>
<td>1.509E−01</td>
<td>2.891E−01</td>
<td>5.535E−01</td>
<td>1.082E−00</td>
</tr>
<tr>
<td>Verifies that the particle belong to the domain and calculating particle–wall interactions</td>
<td>1.563E−02</td>
<td>1.563E−02</td>
<td>1.953E−02</td>
<td>3.320E−02</td>
<td>4.395E−02</td>
</tr>
<tr>
<td>Motion parameters and trajectories</td>
<td>1.562E−02</td>
<td>1.563E−02</td>
<td>1.758E−02</td>
<td>2.344E−02</td>
<td>2.7340E−02</td>
</tr>
</tbody>
</table>
7. The maximum interaction length

It should be noted that the expression (10) for the resultant interaction forces requires summation over all particles. If the number of particles \( N \) is sufficiently large the account of the pair interactions between \( i \)-th particle and all the remaining particles demands too much computing resources and time. Therefore, the maximum interaction length—\( R_{\text{max}} \) is entered to reduce the time of computer calculations. Each particle interacts only with its immediate neighbors located no farther than the maximum interaction length.

In our calculations, we varied the maximum interaction length \( R_{\text{max}} \) from the average distance between particles to the value at which the interaction forces can be neglected due to their smallness (the range of interaction force).

Interestingly, the interaction length significantly affects the grid topology. The smaller the maximum interaction length, the less ordered grid nodes.

8. Grid topology

Most commercial unstructured grid generators use the so-called iterative algorithms. This approach implies incremental node insertion. The important advantage of iterative algorithms is the higher degree of automation that is achieved in the meshing process. Iterative methods are sufficiently flexible. They therefore can be used for triangulation of arbitrary domains. This flexibility, however, comes at a price. Firstly, grids constructed by the iterative methods are usually very heterogeneous. This is due to the fact that the grid topology is formed in the mesh generation process. Secondly, this method is not a reliable enough, because there is a risk of forming unfilled regions (voids or holes in the domain) [24,25].

In contrast to the approach of incrementally inserting nodes one by one into an existing mesh, some mesh generation methods place all the nodes first before connecting them into a mesh. Nodes are
Fig. 14. Three-dimensional adaptive grid around spherical descent vehicle created by the molecular dynamics method: (a) the computational domain; (b) the uniform placement of 25000 nodes inside the computational domain; (c) the node distribution on the bounding surfaces; (d) the mesh on the inner boundary; (e) the cross section of the three-dimensional grid.

Fig. 15. Two-dimensional adaptive grid with refinement in the large gradient region: (a) the arbitrary specified surface, determining the location of the large gradient region; (b) the non-uniform distribution of 8000 nodes inside the domain with the arbitrary specified large gradient region; (c) the triangular Delaunay mesh corresponding to a obtained distribution of nodes.

first distributed into optimal locations to cover the whole domain, and then connected into mesh. The latter group includes a method proposed in this paper.

In Fig. 10 we compare meshes with a different number of nodes \((N = 230, N = 450, N = 610)\) for a circle produced by our method and the GAMBIT automatic mesh generator [26] using the iterative
advancing front algorithm [26–30]. We can see from the figure that our mesh is more homogeneous than the mesh generated by Gambit. The molecular dynamics method generates higher quality mesh than the advancing front method.

Our mesh has better quality as a result of the optimal location of nodes.

Three-dimensional grids created using this technology also have a high homogeneity. Figs. 11 and 12 show the results of application of molecular dynamics method for the generation of uniform unstructured grids in three-dimensional domains of simple and complex shapes.

Fig. 11 shows the uniform placement of nodes inside the cubic and spherical computational domains by molecular dynamics method, and the corresponding tetrahedral meshes. Fig. 12 shows two more examples of application of molecular dynamics technique considered in this paper for the generation of uniform unstructured grids in three-dimensional multiply connected domains. In the first case, the mesh contains 5000 nodes. In the second case mesh has 10000 nodes.

In all the examples discussed in this section, the preliminary allocation of boundary nodes is not required. The node distribution...
Fig. 18. Numerical simulation of external hypersonic flow about the descent vehicle Fire II using the unstructured grid generated by the molecular dynamics method.

Fig. 19. A model of the unmanned airplane X-43.

on the bounding surfaces was obtained simultaneously with distribution in the computational domain.

9. Boundary adaptation of mesh created using molecular dynamics method

The following technique is used to adapt the grids generated by the molecular dynamics method. The theory developed for the interaction of a particle with a bounding surface may be applied to refinement of the grid nodes to a given boundary. However, in this case we use a more complicated force model between the movable interior particles and boundary facets. We add an additional force of attraction acting on the movable interior particle in the normal direction to the nearest facet. If in the process of molecular dynamics simulation some particles cross the domain boundaries, the force changes sign. (See Fig. 13.)

Thus, we can express the force of interaction between movable particle and boundary as follows:

\[
F_{wall} = \begin{cases} 
-A \frac{q_i}{|\vec{n_j}|} \cdot \frac{\vec{n_j}}{|\vec{n_j}|} & \text{at } (\vec{n_j} \cdot \vec{r}_{ij}) > 0, \ m \geq 1, \\
C_{wall} \frac{q_i}{|\vec{n_j}|} \cdot \frac{\vec{n_j}}{|\vec{n_j}|} & \text{at } (\vec{n_j} \cdot \vec{r}_{ij}) < 0, \ m \geq 1.
\end{cases}
\]

(16)

The force of attraction weakens with distance from the boundaries. Besides the boundary is screened by the particle located close to the boundaries. Consequently, the distance between the particles gradually increases to value corresponding to the uniform particle distribution in the rest of the domain. The magnitude of the force may vary depending on the distance to the wall \(A = A(|\vec{r}_{ij}|)\).

Fig. 14 shows the results of application of molecular dynamics method for the generation of unstructured adaptive computational grid for the 3D external flow problem of numerical simulation of the flow about the spherical descent vehicle. It is clearly seen in the figures that the grid is greatly condensed toward the vehicle surface. The preliminary allocation of boundary nodes is not required. The node distribution on the bounding surfaces was obtained simultaneously with distribution in the computational domain. Figure shows the non-uniform placement of 25,000 nodes inside the computational domain and the tetrahedral mesh corresponding to an obtained distribution of nodes. The mesh consists of 149,848 tetrahedral elements.

The adaptive grid method, considered above, not only enables us to condense the grid to a given bounding surface but also allows us to produce a high grid-point density in large gradient regions (such as the shock waves) inside the computational domain (Fig. 15).

10. Construction of Delaunay triangulation of the point set

The Delaunay triangulation for a set of points in the plane is a triangulation such that no point in the set is inside the circum-circle of any triangle in triangulation. For a set of points in the 3D
Fig. 20. The non-uniform placement of 150,000 nodes inside the 3D computational domain.

space, a Delaunay triangulation (tetrahedrization) is a triangulation such that no point in the set is inside the circum-sphere of any tetrahedron in triangulation.

The mesh generator based on Delaunay triangulation was created for triangulation of the 2D and 3D point sets [31,32].

In addition, the tetrahedral mesh generation for the 3D point set is performed using a built-in MATLAB 7.1 function based on Delaunay triangulation.

11. Capability of the method

To illustrate the working of the method we considered several 3D examples. Figs. 16–22 show the results of application of molecular dynamics method for the generation of computational unstructured grids for 3D external flow problems:

- 3D problems of numerical simulation of the external hypersonic flow about the descent vehicle Fire II at its entry into the Earth’s atmosphere;
- 3D problems of numerical simulation of the external hypersonic flow past a model of the unmanned airplane X-43;

The shape of the vehicle Fire II and outer mold line of the geometry is given in Fig. 16 [33].

The inner boundary of the computational domain is determined by the surface of the vehicle. The outer boundary of the domain is specified by the plane and the ellipsoid.

Fig. 17 shows the non-uniform placement of 150,000 nodes inside the three-dimensional computational domain. It is clearly seen in the figure that the grid is greatly condensed toward the vehicle surface.

Fig. 17 shows, also, the tetrahedral Delaunay mesh corresponding to an obtained distribution of nodes and different cross sections of the mesh. The mesh consists of 893,901 tetrahedral elements.

The computational grid has been tested on a problem of the external hypersonic flow about the descent vehicle Fire II.

The flow problem formulation and the incident flow conditions are shown in Fig. 18. The free stream conditions correspond to point in the later part of real path of the descent vehicle entry into the atmosphere. Calculations are performed using methods [34] at zero angle of attack.

The flow solution computed at the zero angle of attack is presented in terms of the Mach number contours in Fig. 18. The data corresponds to the pitch plane. The salient flow features are identified in the \( \alpha = 0^\circ \). These include the bow shock ahead of the body, flow expansion on the shoulder followed by boundary layer separation to form the recirculation bubble on the afterbody. The outer inviscid flow over the separation bubble is turned axially by the recompression shock originating at the neck.

Fig. 18 shows the streamlines on Mach number contours at the zero angle of attack. At the zero angle of attack, the near wake consists of an axisymmetric toroidal vortex. Its two distinct loops in the pitch plane can be seen in Fig. 18.
A model of the unmanned airplane X-43 was created using SolidWorks (Fig. 19). As in the previous example, the inner boundary of the computational domain is determined by the surface of the airplane X-43. The outer boundary of the domain is specified by the plane and the ellipsoid.

Fig. 20 shows the non-uniform placement of 150,000 nodes inside the three-dimensional computational domain. It is clearly seen in the figure that the grid is greatly condensed toward the X-43 surface.

Fig. 21 shows the tetrahedral Delaunay mesh corresponding to an obtained distribution of nodes and different cross sections of the mesh. The mesh consists of 895,103 tetrahedral elements.

The computational grid has been tested on a problem of the external supersonic flow past a model of the unmanned airplane X-43. The flow problem formulation and the incident flow conditions are shown in Fig. 22.

Calculations are performed using methods [35] in the range of Mach number from 3 to 6, at the zero angle of attack. The flow solution computed at the zero angle of attack and \( M = 6 \) is presented in terms of the Mach number contours in Fig. 22.

In all considered cases, the preliminary allocation of boundary nodes is not required. The node distribution on the bounding surfaces was obtained simultaneously with distribution in the computational domain.

12. Conclusions

A new approach to triangular mesh generation based on the molecular dynamics method is proposed. Mesh adaptation technology for molecular dynamics simulation is presented.

There are some advantages for the proposed meshing method. They are listed as follows:

1. The method has sufficient flexibility and reliability. It can be used for triangulation of complex multiply connected domains.
2. Computational grids created using this technology have a high homogeneity and quality as a result of the optimal location of nodes.
3. The algorithm is simple and easy to implement and works identical for 1D, 2D and 3D problems. Unlike other methods, moving from one dimensionality to another will not increase the complexity of the method.
4. In almost all cases, preliminary allocation of boundary nodes is not required. The node distribution on the bounding surfaces was obtained simultaneously with distribution in the computational domain.
5. The method is capable to control nodal density. The molecular dynamics method allows us to get a high quality uniform and non-uniform (adaptive) unstructured grids.
6. A new approach can be used for optimization of an existing mesh.

The method considered in this paper also has some disadvantages:

1. The system of interacting particles has a high sensitivity to the model parameters variation. A careful design for a combination of physical parameters is needed in order to avoid instabilities and slow convergence.
2. The molecular dynamics method suggests the integration of the differential equations of motion at each time step.
Some examples of application of molecular dynamics technique for the generation of uniform and non-uniform unstructured grids about the complex 3D models of hypersonic vehicles are considered in order to illustrate the method's ability.

The results of this paper show that you can get the unstructured adaptive computational grids for various gas dynamics problems using the mesh generator based on the molecular dynamics method.

Acknowledgments

The study was supported by the Russian foundation for basic research (RFBR project No 13-01-0537), and by Russian Academy of Sciences in the framework of Basic research.

The research leading to these results has received funding from the European Community’s Seventh Framework Programme (FP7/2007-2013) under grant agreement no 242311.

References


