Using a tetrahedral mesh for simulation of internal flows by the DSMC method

doi: 10.1063/1.5117412

Using a tetrahedral mesh for simulation of internal flows by the DSMC method

Cite as: AIP Conference Proceedings **2125**, 030030 (2019); https://doi.org/10.1063/1.5117412
Published Online: 26 July 2019

Pavel Vashchenkov, Alexandr Kashkovsky, Yevgeniy Bondar, Shigeru Yonemura, and Yoshiaki Kawagoe

Using a Tetrahedral Mesh for Simulation of Internal Flows by the DSMC Method

Pavel Vashchenkov[1,a)], Alexandr Kashkovsky[1], Yevgeniy Bondar[1], Shigeru Yonemura[2] and Yoshiaki Kawagoe[3]

[1]*Khristianovich Institute of Theoretical and Applied Mechanics SB RAS, Institutskaya 4/1, Novosibirsk, 630090, Russia*
[2]*Department of Nanomechanics, Graduate School of Engineering, Tohoku University, Japan*
[3]*Institute of Fluid Science, Tohoku University, Japan*

[a)]Corresponding author: vashen@itam.nsc.ru

Abstract. The paper describes the specific features of using a tetrahedral mesh for simulating rarefied gas flows by the DSMC method by an example of the SMILE++ software system. Algorithms used for modeling molecule transfer and for computation parallelization are presented; an example of modeling a flow in a microchannel is described.

INTRODUCTION

The most popular method used for simulating rarefied gas flows is the Direct Simulation Monte Carlo (DSMC) method [1]. It is a numerical statistical method of solving the Boltzmann equation.

In DSMC simulations, the gas motion is modeled statistically on the basis of the motion of model particles, each representing a large number of real gas molecules. The simulation process is divided into two stages:

- intermolecular collisions;
- free-molecular transfer.

Collisions between the particles located in one spatial cell are calculated at the stage of intermolecular collisions. After these collisions, the particles remain at the same positions, but their velocities change. Collision cells are used to group particles that can collide with each other. The cell shape is not important for modeling collisions, but the speed of determining in which cell the particle is located plays a key role. At the stage of free-molecular transfer, the particles are shifted by a distance $\Delta\vec{r} = \vec{V}\Delta t$.

There are various codes for DSMC computations. Many of them use rectangular meshes for simulations (SMILE [2, 3], SMILE-GPU[4, 5] , DSMCFoam+[6], SPARTA[7]); some other codes use unstructured meshes (Monaco [8]). The use of the Cartesian mesh in DSMC codes is caused by the simplicity of its application. The particles move immediately from the initial to the final point in accordance with the time step and particle velocity. In this case, it is not important how many cells are crossed by the particle in the process. The cell number can be easily calculated from the particle coordinates; therefore, there is no need to store it.

If the Cartesian mesh is used, the body surface is presented by a triangulated unstructured mesh. Its faces are not connected to the spatial mesh. When the particle is transferred, it is necessary to determine whether the particle collides with a triangular face on the body surface and, if so, with which particular face it collides. In searching for the point of intersection of the particle trajectory with the surface mesh, it may also happen that the particle falls onto the rib separating the faces. Another option for the particle is to stop in a small vicinity of the face of the surface mesh at the end of some time step. Because of the machine accuracy of number presentation, the particle may penetrate through the surface in such situations. The probability of such through flight is rather low; in the SMILE++ code for external flows, this probability is estimated to be lower than 10^{-6}. Therefore, the error due to the existence of such

High-Energy Processes in Condensed Matter (HEPCM 2019)
AIP Conf. Proc. 2125, 030030-1–030030-6; https://doi.org/10.1063/1.5117412
Published by AIP Publishing. 978-0-7354-1865-3/$30.00

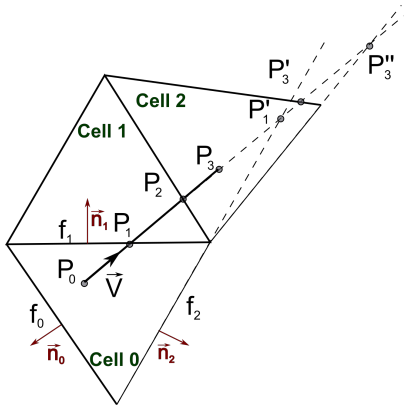

FIGURE 1. Schematic of free-molecular transfer of particles.

particles is significantly smaller than the statistical error of the DSMC method. The use of such a grid shows a good applicability in problems of external aerodynamics both for modeling of external flows and for jet flows [9, 10].

This problem becomes important in modeling flows in complex-shaped channels. Passing through the channel, the particles experience thousands of collisions with the surface, and vanishing of model particles due to the reasons discussed above will lead to unphysical density decrease inside the channel.

Microchannel flows are usually characterized by complicated geometry in which it is rather difficult to construct a structured mesh. Thin internal partitions can divide the mesh cell into several parts with no gas interaction between them in reality. However, the DSMC method implies that all particles located in one cell can collide with each other. As a result, the flow pattern can be significantly distorted.

The use of an unstructured mesh allows one to avoid these problems. A particle in an unstructured mesh always belongs to some cell. Even if this particle leaves the cell with the machine accuracy, this fact does not affect its further motion, and the particle never crosses the solid wall and becomes lost. According to the definition of the unstructured mesh, it is always body-fixed, which automatically resolves the problem of thin walls. For simulating flows inside channels, it is necessary to use unstructured meshes.

MODELING OF PARTICLE TRANSFER

Implementation of algorithm for simulations on an unstructured tetrahedral mesh in the SMILE++ software code [11, 12] is considered in the present paper.

The unstructured mesh in SMILE++ consists of tetrahedra. The faces that form the tetrahedral cell are described by the equation of the plane $\vec{n}\cdot\vec{r}+D=0$, where $\vec{n}$ is the normal to the plane, $\vec{r}$ is the radius-vector of a point located on the plane, and D is a free term. The equation components for each tetrahedron face are stored. Moreover, the numbers of cells separated by each face are also stored. If the face is located at the computational domain boundary, the type of the boundary condition is stored instead of the number of one of the separated cells.

Particle transfer is performed in the following way. Let the particle be located at a point $\vec{P}_0$, which belongs to cell 0. The particle velocity is $\vec{V}$. We have to determine in which cell the particle will be located after its motion during the time step Δt. The transfer simulation is schematically shown in Figure 1.

1. The particle velocities toward the faces $V_{n,i}=\vec{V}\cdot\vec{n}_i$ ($\vec{n}_i$ is the external normal to the face) are calculated. Obviously, the particle trajectory can cross only such a face for which $V_{n,i}>0$. Further steps are performed only for faces of this kind.
2. The time of particle trajectory intersection with the plane of the i-th face is calculated: $t_{c,i}=-\frac{\vec{P}_0\cdot\vec{n}_i+D_i}{V_{n,i}}$. Obviously, the particle will leave the cell through the face for which this time has the minimum value ($t_{c,min}$).
3. The point on the face through which the particle leaves the cell is determined: $\vec{P}_1=\vec{P}_0+\vec{V}t_{c,min}$. Depending on the face type, the following results of its intersection are possible:
   - The face is an input or output boundary. The particle is eliminated from the calculation, and the next particle should be considered.

- The face is a usual face between the cells. In this case, the particle is moved to the neighboring cell.
- The face is the body surface. In this case, the particle is reflected from the surface. The particle velocity changes.
- The face is the plane of symmetry. The particle velocity vector is replaced by its mirror reflection.

4. Such a procedure of particle transfer is continued until the total time of reaching the cell faces exceeds the total time step. When this condition is satisfied, the particle transfer procedure is finalized.

The main idea of this algorithm is fairly close to that presented in [13], except for two aspects: (1) instead of the panel center, the proposed algorithm contains the free term D from the canonical equation of the plane (this fact saves the computer memory); (2) The time step is used in our algorithm as a parameter for tracing intersection of the cell boundary by the particle instead of the distance to the final point. This fact ensures a gain of one operation of vector summation for each intersection of the cell boundary.

Implementation of this algorithm is simple and fast. It can be used on meshes with convex cells of any kind.

PARALLELIZATION OF COMPUTATIONS

The use of multi-core computers is a common practice in modern computations. Parallelization in the DSMC method is usually performed by means of decomposition of the computational domain, which is divided into subdomains with the number of subdomains equal to the number of computational cores. Each core operates with one subdomain allocated to it.

At the particle transfer stage, the particles can move from the subdomain allocated to one core to the subdomain allocated to another core. Therefore, after the particle transfer stage is finalized, the particles that moved to the subdomain controlled by another core are transferred between the processors in accordance with the MPI protocol. The computational domain is divided with the use of the METIS library with weight coefficients. At the first instant of computational domain division, the cell volume is taken as a weight factor. Based on this information, a decomposition map is generated, where the belonging of each cell to a certain core is stored.

At the stage of particle transfer between the cells, it may happen that the particles travels through subdomain that belong to several cores during one time step. If a Cartesian mesh is used, there is no need to trace the intermediate positions of the particle during the time step because the number of the final cell is easily calculated. If a tetrahedral mesh is used, it is rather difficult to determine the cell number from the particle coordinates. Therefore, it is preferable to trace the transitions between the neighboring cells consecutively. In this case, each core should "know" the information about the cells not only from its own subdomain, but also about all cells where its particles can arrive. The simplest variant to ensure such a situation is to store the entire mesh on each core. During particle transfer, the number of the core controlling the cells where the particles moved is determined, and, if necessary, these particles are transferred to this core. In this case, however, the memory is not rationally used. If the computation is performed on a multi-core computer and the mesh occupies approximately 10% of the memory, it means that the entire memory will be used for mesh storage if ten cores are involved.

Therefore, parallelization is performed in SMILE++ with the following procedure:

- All parameters of the mesh necessary for simulations are determined and written to a binary dump file with arbitrary access.
- Each core reads the decomposition map.
- In accordance with the map, each core loads information about all cells belonging to it from the dump file.
- For convenience of collecting surface characteristics, all boundary faces are duplicated on all cores.
- At the stage of particle transfer, each core moves the particles between its cells. If some particle moves away to a cell belonging to another core, information about this cell is read out from the dump file, and further motion of the particle is calculated. Thus, an additional layer of near-boundary cells appears on each core. This near-boundary layer of cells is used only for determining to which core the particle should be transferred at the end of the time step.
- After the particle transfer stage, all cores exchange by particles that passed to subdomains belonging to other cores.

Uploading of additional cells from the dump file for calculating particle transfer is performed only once. Thus, the near-boundary layer is generated at the initial steps of simulations, and then there is no need to address the disk for adding new cells. As the particles in the DSMC method should not move by more than one cell on the average during one time step, each core receives data on the cells from their own computational subdomain and from a rather narrow layer of cells that belong to other cores.

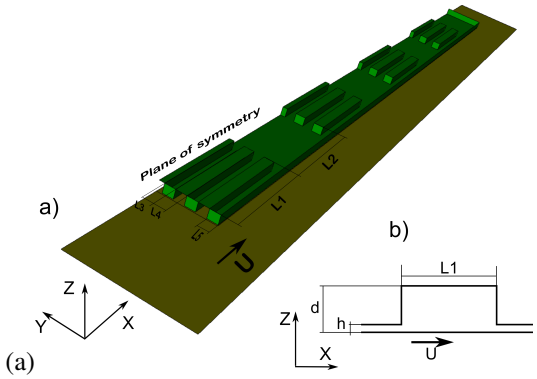

FIGURE 2. Geometric model of the microchannel.

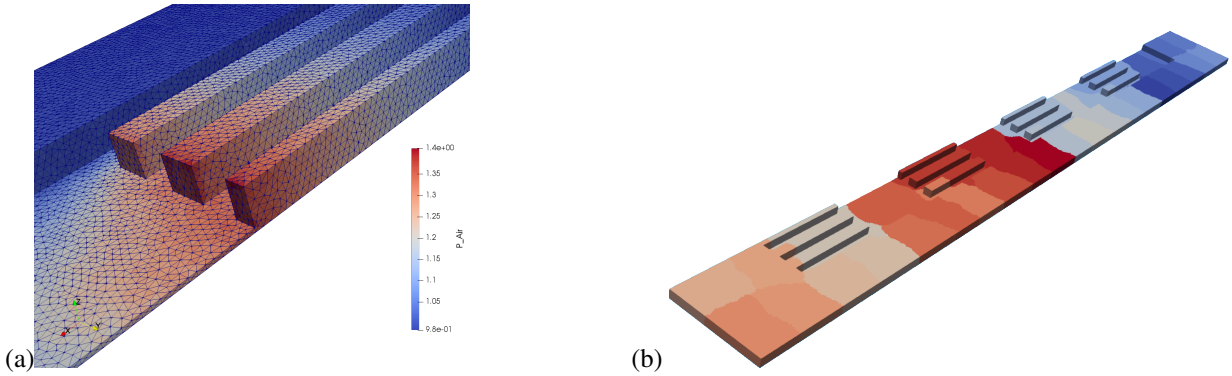

FIGURE 3. Grid and domain decomposition.

MODELING OF MICROCHANNEL FLOWS

An important issue of creating microdevices is the motion of rough surfaces with respect to each other (e.g., flows in microbearing, near read/write heads of HDD's, etc.). Owing to an increase in pressure arising on the rough surfaces, these devices are stabilized at a needed distance from each other. Such a pressure distribution was studied in a model formulation, where a gas flow between two plates was considered; one of these plates had a structured three-dimensional surface, while the other (flat) plate moved in the longitudinal direction with a velocity of 10 m/s. Though microdevices operate at atmospheric pressure, the distances between the surfaces are comparable with the mean free path of gas molecules, and the flow can be simulated by the DSMC method. Figure 2 shows the geometric model used in the study. The roughness was defined by rectangular cavities. The computations were performed for the distances between the plates h varied from 0.1 μm to 0.01 μm, $d = 1$ μm, $L_1 = 10$ μm,and $L_2 = 10 - 40$ μm. Thus, the ratio of the channel length to its minimum height varied from 800 to 20000. An air flow at a pressure of 1 atm and a temperature of 300 K was considered. The mean free path under these conditions was $\lambda \approx 0.065$ μm. Thus, the Knudsen number based on the minimum height of the channel varied from 0.65 to 6.5.

Figure 3(a) shows an example of a tetrahedral computational mesh. One can see the external boundary of the computational domain. The convex rectangular elements are cavities on the motionless upper surface. Depending on the geometric parameters of the channel, the number of cells in the computational domain varied from 1.5 to 3.5 million. Figure 3(b) shows the computational domain decomposition for 28 processors with the use of the METIS library.

Figure 4(a) shows the pressure field normalized to the free-stream pressure on the channel surface. The field corresponds to the case with the channel height of 0.01 μm. As is seen from the figure, the maximum pressure (≈ 1.4) is reached at the end of the first row of cavities. It increases monotonically inside the cavity and then becomes almost constant in the narrow part of the channel. When the next cavity is approached, the pressure drastically decreases approximately to 0.85. The pressure in the next rows of cavities reaches ≈ 1.2.

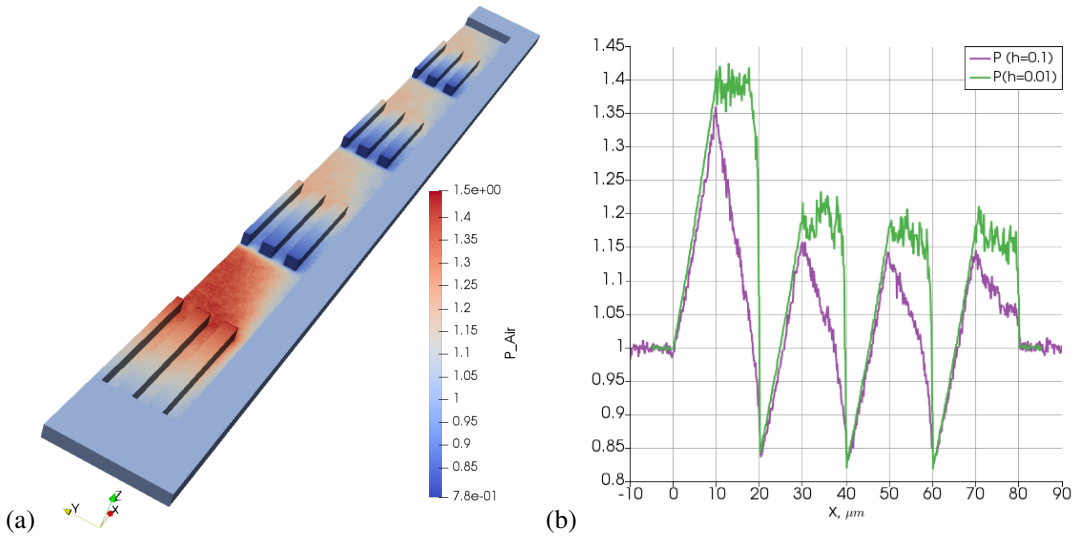

FIGURE 4. Pressure distribution in the channel.

Figure 4(b) shows the pressure in the plane of symmetry. The distributions for the cases with $h = 0.01$ μm and $h = 0.1$ mum are plotted by the green and magenta colors, respectively. An increase in the channel height leads to a somewhat different pressure distribution in the channel. The maximum pressure reaches ≈ 1.35 in the first cavity and ≈ 1.15 in the next cavities. In the narrow part of the channel, in contrast to the previous case, there is no constant-pressure region. It decreases almost linearly along the entire narrow part of the channel. The minimum values of pressure are identical for both cases.

For comparison of the SMILE++ operation efficiency on the tetrahedral and Cartesian meshes, the shortest channel configuration was computed. The correctness of the computation on the Cartesian mesh was ignored in this case. The times needed for the computations were 42.12 hours on the Cartesian mesh with 25 processors and 55.18 hours on the tetrahedral mesh.

CONCLUSIONS

It is not recommended to use Cartesian meshes for modeling flows in complex or long microchannels because of the emergence of unphysical effects. The present paper describes the algorithm of particle transfer simulation in the DSMC method on a tetrahedral mesh and also the method of parallelization of computations based on computational domain decomposition. The test computations show that the simulation of rarefied gas flows on the tetrahedral mesh is slower (approximately by a factor of 1.3) than that on the Cartesian mesh. The proposed algorithm can be also used on a mesh with convex cavities regardless of their type and potentially even on hybrid meshes.

ACKNOWLEDGMENTS

This work was partly supported by the General Collaborative Research Project J19I063 of Institute of Fluid Science, Tohoku University.

The research was partly carried out within the framework of the Program of Fundamental Scientific Research of the state academies of sciences in 2013-2020 (project No. AAAA-A17-117030610138-7) and within the grant of the Russian Foundation for Basic Research (project No. 18-38-20113).

Computational resources were kindly provided by Siberian Supercomputer Center of the Institute of Computational Mathematics and Mathematical Geophysics SB RAS (sscc.ru), Computational Center of Novosibirsk State University (nusc.nsu.ru) and Moscow State University Supercomputing Center (parallel.ru).

REFERENCES

[1] G. A. Bird, Molecular Gas Dynamics and the Direct Simulation of Gas Flows (Clarendon Press, Oxford, 1994).

[2] M. S. Ivanov, G. N. Markelov, S. F. Gimelshein, and L. V. Mishina, J. Spacecraft and Rockets **35**, 16–22 (1998), doi: 10.2514/3.26992.

[3] M. S. Ivanov and S. V. Rogazinsky, Russ. J. Numer. Anal. Math. Model. **3**, p. 453–465 (1988), doi: 10.1515/rnam.1988.3.6.453.

[4] A. Kashkovsky, AIP Conference Proceedings 1628, 192–198 (2014), doi: 10.1063/1.4902592.

[5] A. V. Kashkovsky, A. A. Shershnev, and P. V. Vashchenkov, AIP Conf. Proc. **1893**, p. 030047 (2017), doi: 10.1063/1.5007505.

[6] C. White, M. Borg, T. Scanlon, S. Longshaw, B. John, D. Emerson, and J. Reese, Comput. Phys. Commun. **224**, 22–43 (2018).

[7] M. A. Gallis, J. R. Torczynski, S. J. Plimpton, D. J. Rader, and T. Koehler, AIP Conference Proceedings **1628**, 27–36 (2014), doi: 10.1063/1.4902571.

[8] S. Dietrich and I. D. Boyd, J. Comp. Phys. **126**, 328–342 (1996).

[9] A. V. Kashkovsky, P. V. Vashchenkov, and T. Banyai, Thermophysics and Aeromechanics **21**, 719–728 (2014), doi: 10.1134/S0869864314060067.

[10] A. V. Kashkovsky, P. V. Vashchenkov, A. N. Krylov, and L. V. Mishina, Thermophysics and Aeromechanics **25**, 643–658 (2018), doi: 10.1134/S0869864318050013.

[11] A. V. Kashkovsky, G. N. Markelov, and M. S. Ivanov, "An object-oriented software design for the direct simulation Monte Carlo method," in *Proc. of 35th AIAA Thermophysics Conference* (2001), doi: 10.2514/6.2001-2895.

[12] M. S. Ivanov, A. V. Kashkovsky, P. V. Vashchenkov, and Y. A. Bondar, AIP Conf. Proc. **1333**, 211–218 (2011), doi: 10.1063/1.3562650.

[13] G. B. Macpherson, N. Nordin, and H. G. Weller, Communications in Numerical Methods in Engineering **25**, 263–273 (2009), doi: 10.1002/cnm.1128.

[14] A. V. Kashkovsky, Y. A. Bondar, G. A. Zhukova, M. S. Ivanov, and S. F. Gimelshein, AIP Conf. Proc. **762**, 583–588 (2005), doi: 10.1063/1.1941599.