

A Cartesian cut cell method for rarefied flow simulations around moving obstacles



G. Dechristé^{a,b,*}, L. Mieussens^{a,b,c,d}

^a Univ. Bordeaux, IMB, UMR 5251, F-33400 Talence, France

^b CNRS, IMB, UMR 5251, F-33400 Talence, France

^c Bordeaux INP, IMB, UMR 5251, F-33400 Talence, France

^d INRIA, F-33400 Talence, France

ARTICLE INFO

Article history:

Received 12 March 2015

Received in revised form 6 March 2016

Accepted 11 March 2016

Available online 16 March 2016

Keywords:

Kinetic equations

Deterministic method

Immersed boundaries

Cut cell method

Rarefied gas dynamics

ABSTRACT

For accurate simulations of rarefied gas flows around moving obstacles, we propose a cut cell method on Cartesian grids: it allows exact conservation and accurate treatment of boundary conditions. Our approach is designed to treat Cartesian cells and various kinds of cut cells by the same algorithm, with no need to identify the specific shape of each cut cell. This makes the implementation quite simple, and allows a direct extension to 3D problems. Such simulations are also made possible by using an adaptive mesh refinement technique and a hybrid parallel implementation. This is illustrated by several test cases, including a 3D unsteady simulation of the Crookes radiometer.

© 2016 Elsevier Inc. All rights reserved.

1. Introduction

In gas dynamic problems, the rarefied regime appears when the mean free path of the molecules of the gas is of the same order of magnitude as a characteristic macroscopic length. The flow has to be modeled by the Boltzmann equation of the kinetic theory of gases. Most of numerical simulations for rarefied flows are made with the stochastic DSMC method [8], especially for aerodynamical flows in re-entry problems. In the past few years, several deterministic solvers have been proposed, that are based on discretizations of the Boltzmann equation or simplified models, like BGK, ES-BGK, or Shakhov models [30]. They are efficient for accurate simulations, multi-scale problems, or transitional flows, for instance.

A recent issue is the account of solid boundary motion in rarefied flow simulations. This is necessary to simulate flows around moving parts of micro-electromechanical systems (MEMS) [19,25], as well as flows inside vacuum pumps. A fascinating illustration of rarefied flows with moving boundaries is the Crookes radiometer, subject of many debates from the late 19th to early 20th century [26]. Recent deterministic simulations help to understand the origin of the radiometric forces [41,46,47,10,48]. The numerical simulation of the Crookes radiometer is difficult because the motion of the vanes is induced by gas/solid interaction (like thermal creep), which means that an accurate prediction of the flow in the vicinity of the boundary is needed in order to predict the correct velocity of the vanes.

There are several numerical methods for moving boundary problems designed for computational fluid dynamics: some of them have recently been extended to deterministic discretizations of kinetic models, and can be divided in two main categories.

* Corresponding author.

E-mail addresses: Guillaume.Dechriste@math.u-bordeaux1.fr (G. Dechristé), Luc.Mieussens@math.u-bordeaux1.fr (L. Mieussens).

First, with body fitted methods, the mesh is adapted at each time step so that the boundary of the computational domain always fit with the physical boundary: moving mesh [49] and ALE methods [20,21] fall into this category. Despite their extensive use in computational fluid dynamics, very few similar works have been reported in kinetic theory, except by Chen et al. [11]. Methods of the second category are based on Cartesian grid computations and are usually referred to as immersed boundary methods [31]. The mesh does not change during computations, and hence does not fit with the physical boundary. Special treatment is applied on mesh cells that are located close to the boundary in order to take its motion into account. Various extensions of these methods to kinetic theory have been proposed by several authors in [3,34,16,6]. Two recent variants are the inverse Lax–Wendroff immersed boundary method proposed by Filbet and Yang [18] and the Cartesian grid-based unified gas kinetic scheme of Chen and Xu [9]: the boundary motion is not taken into account in these two works, but these methods could in principle be extended to this kind of problem. We also mention the Lagrangian method: while it falls into the first category in CFD, it does not in kinetic theory. Indeed, whatever the motion of the mesh, the distribution function has to be interpolated at the foot of the characteristic for each microscopic velocity. The accuracy of these methods has been shown in [40,53] for one dimensional problems. Finally, we mention that moving boundary flows can also be treated with DSMC solvers: see, for instance, [33,38,44,45,58].

In this paper, we try to mix the advantages of body fitted and Cartesian methods: we present a cut cell method for computing rarefied gas flows around moving obstacles. The method consists in extracting a body fitted mesh from an initial Cartesian mesh. This approach has first been proposed to study inviscid flows [13,5,59]. It has later been extended to viscous flow simulations [54,57] and also adapted to moving wall problems [55,32,56]. A more extensive review of its applications can be found in [24]. However, this is the first extension to moving boundary problems in kinetic theory (complex 3D stationary DSMC simulations have already been investigated in [28,60]). This approach is well suited to deterministic approximations of the Boltzmann equation and is easy to implement because of the Cartesian structure of the mesh. Moreover, this is, up to our knowledge, the only immersed boundary method to be conservative. The versatility and robustness of the technique is illustrated by various 2D flows, and by the simulation of the unsteady rotation of the vanes of a 3D Crookes radiometer. This article is an extended version of our work announced in [16]. Here, the Boltzmann collision operator is replaced by BGK like models, that are approximated by a discrete velocity method. However, this is not a restriction: other collision operators could be used, and any velocity approximation (like the spectral method) could be used.

Generally, the problem of cut cell methods is that it is difficult to take into account the various shapes of cells that are cut by the solid boundary: for instance, in 2D, a cut cell can be a triangle, a quadrangle, or a pentagon, and this is worse in 3D. Here, we propose a simple representation of these cells by using the notion of virtual cells that are polygons (or polyhedrals) with possibly degenerated edges (or faces). This makes the treatment of any cut cell completely generic: in the implementation, the different kinds of cut cells and the non-cut cells are treated by the same algorithm. This makes the extension of the method to 3D problems very easy. However, to make large scale 3D simulations possible, we also use an adaptive mesh refinement (AMR) technique and a special parallel implementation.

The outline of our paper is as follows. In section 2, we give the governing equations of rarefied gas flows and introduce some notations. Our cut cell method is presented in section 3 for 2D problems. It is validated on three different numerical examples in section 4. Then, in section 5, our algorithm is extended to 3D simulations, and a 3D unsteady simulation of the Crookes radiometer is presented. Finally, some conclusions and perspectives are discussed in section 6. Technical details like computations of geometric parameters of the cells are presented in the Appendix.

2. Rarefied gas dynamics

2.1. Boltzmann equation

In rarefied regimes, a monoatomic gas is described by the Boltzmann equation:

$$\frac{\partial F}{\partial t} + \vec{v} \cdot \nabla F = Q(F). \quad (1)$$

The distribution function $F(t, \vec{x}, \vec{v})$ is the mass density of molecules at time t that are located at the space coordinate $\vec{x} \in \mathbb{R}^3$ and that have a velocity $\vec{v} \in \mathbb{R}^3$. For our approach, it is more relevant to look at the integral form of (1) in a time dependent volume $V(t)$. The Reynolds transport theorem leads to:

$$\frac{\partial}{\partial t} \int_V F dV + \int_{\partial V} (\vec{v} - \vec{w}) \cdot \vec{n} F dS = \int_V Q(F) dV, \quad (2)$$

where $\partial V(t)$ is the surface of the volume $V(t)$. Let \vec{x} be a point of this surface: it is moving at a velocity $\vec{w}(t, \vec{x})$ and the vector $\vec{n}(t, \vec{x})$ is the outward normal vector to the surface at this point.

The density ρ , momentum $\rho \vec{u}$, total energy E and stress tensor $\overline{\Sigma}$, are computed by the first moments of the distribution function with respect to the velocity:

$$\begin{aligned} \begin{bmatrix} \rho \\ \rho \vec{u} \\ E \end{bmatrix} &= \int_{\mathbb{R}^3} \begin{bmatrix} 1 \\ \|\vec{v}\| \\ \frac{1}{2} \|\vec{v}\|^2 \end{bmatrix} F(t, \vec{x}, \vec{v}) d\mathbf{v}_x d\mathbf{v}_y d\mathbf{v}_z, \\ \vec{\Sigma} &= \int_{\mathbb{R}^3} (\vec{v} - \vec{u}) \otimes (\vec{v} - \vec{u}) F(t, \vec{x}, \vec{v}) d\mathbf{v}_x d\mathbf{v}_y d\mathbf{v}_z, \end{aligned} \quad (3)$$

where the norm is defined by $\|\vec{v}\|^2 = v_x^2 + v_y^2 + v_z^2$. The temperature T of the gas is related to the energy by the relation $E = \frac{1}{2} \rho \|\vec{u}\|^2 + \frac{3}{2} \rho RT$, where R is the gas constant defined as the ratio between the Boltzmann constant and the molecular mass of the gas. Moreover, the pressure is computed with the standard equation of state for ideal gases: $P = \rho RT$.

When the gas is in thermodynamic equilibrium, the molecules are uniformly distributed around the macroscopic velocity and the distribution function is a Gaussian function called Maxwellian:

$$\mathcal{M}[\rho, \vec{u}, T](\vec{v}) = \frac{\rho}{(2\pi RT)^{3/2}} \exp\left(-\frac{\|\vec{v} - \vec{u}\|^2}{2RT}\right). \quad (4)$$

The Boltzmann collision operator $Q(F)$ is computationally expensive, and is often simplified by the BGK [7] relaxation operator

$$Q(F) = \frac{1}{\tau} (\mathcal{M}[\rho, \vec{u}, T] - F).$$

This model leads to a Prandtl number equal to 1, but generalized equilibrium can be used instead of M to get a correct Prandtl number, like in Shakhov [43] or ES-BGK models [22].

In this article, solid wall interactions are taken into account by the standard fully diffuse reflection. This model states that all particles that collide with a boundary are absorbed by the wall and re emitted with a Maxwellian distribution:

$$F(t, \vec{x} \in \Gamma, \vec{v} \in \mathcal{V}_{\text{in}}) = \phi \mathcal{M}[1, \vec{u}_w, T_w], \quad (5)$$

where T_w and \vec{u}_w are the temperature and the velocity of the boundary Γ at position \vec{x} . The coefficient ϕ is computed in order to set the net mass flux across the wall to zero:

$$\phi = - \left(\int_{\vec{v} \in \mathcal{V}_{\text{out}}} (\vec{v} - \vec{u}_w) \cdot \vec{n}_w F d\mathbf{v}_x d\mathbf{v}_y d\mathbf{v}_z \right) / \left(\int_{\vec{v} \in \mathcal{V}_{\text{in}}} (\vec{v} - \vec{u}_w) \cdot \vec{n}_w \mathcal{M}[1, \vec{u}_w, T_w] d\mathbf{v}_x d\mathbf{v}_y d\mathbf{v}_z \right).$$

The set of incoming velocities is defined by $\mathcal{V}_{\text{in}} = \{\vec{v} \text{ such that } (\vec{v} - \vec{u}_w) \cdot \vec{n}_w < 0\}$, where \vec{n}_w is the normal vector to the boundary, pointed to the wall. Similarly, the set of outgoing velocities is $\mathcal{V}_{\text{out}} = \{\vec{v} \text{ such that } (\vec{v} - \vec{u}_w) \cdot \vec{n}_w > 0\}$. Note that the boundary condition is defined only for the relative incoming microscopic velocities.

2.2. Reduced model

For plane flows, the computational complexity of the Boltzmann equation can be decreased by the use of a standard reduced distribution technique [12]. This classical method has been extensively used for numerical computations of BGK and Shakhov models. First note that in plane flows, the third component of the macroscopic velocity u_z is equal to zero, as well as Σ_{xz} , and Σ_{yz} . From now on, we define the two dimensional variables

$$\mathbf{x} = (x, y), \quad \mathbf{v} = (v_x, v_y), \quad \mathbf{u} = (u_x, u_y), \quad \Sigma = \begin{pmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{pmatrix}.$$

Let f and g be the reduced distribution functions defined by

$$f(\mathbf{v}) = \int_{\mathbb{R}} F(\vec{v}) d\mathbf{v}_z, \quad g(\mathbf{v}) = \int_{\mathbb{R}} \frac{1}{2} v_z^2 F(\vec{v}) d\mathbf{v}_z.$$

The macroscopic quantities can be computed from f and g . Indeed, the set of equations (3) readily becomes

$$\begin{aligned} \begin{bmatrix} \rho \\ \rho \mathbf{u} \\ E \end{bmatrix} &= \int_{\mathbb{R}^2} \begin{bmatrix} 1 \\ \mathbf{v} \\ \frac{1}{2} \|\mathbf{v}\|^2 \end{bmatrix} f(\mathbf{v}) d\mathbf{v}_x d\mathbf{v}_y + \int_{\mathbb{R}^2} \begin{bmatrix} 0 \\ \mathbf{0} \\ 1 \end{bmatrix} g(\mathbf{v}) d\mathbf{v}_x d\mathbf{v}_y, \\ \Sigma &= \int_{\mathbb{R}^2} (\mathbf{v} - \mathbf{u}) \otimes (\mathbf{v} - \mathbf{u}) f(\mathbf{v}) d\mathbf{v}_x d\mathbf{v}_y, \end{aligned} \quad (6)$$

where $\|\mathbf{v}\|^2 = v_x^2 + v_y^2$. Multiplying by $(1, \frac{1}{2}v_z^2)$ the Boltzmann equation (2) and then integrating the result with respect to v_z gives the following set of equations:

$$\begin{aligned} \frac{\partial}{\partial t} \int_S f \, dS + \int_{\partial S} (\mathbf{v} - \mathbf{w}) \cdot \mathbf{n} f \, dl &= \int_S \frac{1}{\tau} (F - f) \, dS, \\ \frac{\partial}{\partial t} \int_S g \, dS + \int_{\partial S} (\mathbf{v} - \mathbf{w}) \cdot \mathbf{n} g \, dl &= \int_S \frac{1}{\tau} (G - g) \, dS. \end{aligned} \quad (7)$$

In this case, $\partial S(t)$ is the contour of the surface $S(t)$. Each point $\mathbf{x} \in \partial S(t)$ is moving at a velocity $\mathbf{w}(t, \vec{x})$ and $\mathbf{n}(t, \vec{x})$ is the outward normal vector to the contour at this point. The reduced equilibrium functions are defined by

$$F = M[\rho, \mathbf{u}, T] \quad \text{and} \quad G = \frac{RT}{2} M[\rho, \mathbf{u}, T], \quad (8)$$

where $M[\rho, \mathbf{u}, T]$ is the reduced Maxwellian given by

$$M[\rho, \mathbf{u}, T](\mathbf{v}) = \frac{\rho}{2\pi RT} \exp\left(-\frac{\|\mathbf{v} - \mathbf{u}\|^2}{2RT}\right).$$

To close this section, the boundary conditions (5) are written with the reduced distribution functions as:

$$f(t, \mathbf{x} \in \Gamma, \mathbf{v} \in \mathcal{V}_{\text{in}}) = \phi M[1, \mathbf{u}_w, T_w], \quad \text{and} \quad g(t, \mathbf{x} \in \Gamma, \mathbf{v} \in \mathcal{V}_{\text{in}}) = \phi \frac{RT}{2} M[1, \mathbf{u}_w, T_w], \quad (9)$$

where ϕ is computed by

$$\phi = - \left(\int_{\mathbf{v} \in \mathcal{V}_{\text{out}}} (\mathbf{v} - \mathbf{u}_w) \cdot \mathbf{n}_w f \, dv_x dv_y \right) / \left(\int_{\mathbf{v} \in \mathcal{V}_{\text{in}}} (\mathbf{v} - \mathbf{u}_w) \cdot \mathbf{n}_w M[1, \mathbf{u}_w, T_w] \, dv_x dv_y \right).$$

In this formula, \mathbf{u}_w and T_w are the velocity and temperature of the point \mathbf{x} that belongs to the boundary Γ , and \mathbf{n}_w is the normal vector to the boundary pointed to the wall. The relative incoming and outgoing velocities at this point are therefore defined by $\mathcal{V}_{\text{in}} = \{\mathbf{v} \text{ such that } (\mathbf{v} - \mathbf{u}_w) \cdot \mathbf{n}_w < 0\}$ and $\mathcal{V}_{\text{out}} = \{\mathbf{v} \text{ such that } (\mathbf{v} - \mathbf{u}_w) \cdot \mathbf{n}_w > 0\}$.

3. The cut-cell method for two dimensional problems

In this section, we present a numerical method to simulate plane flows with moving boundaries. The governing equations are detailed in section 2.2. The discretization of each variable (velocity, space, and time) is presented in separate sections.

3.1. Discrete velocity approximation

The velocity space is discretized by a Cartesian grid. Let $\mathbf{v}_{\min} \in \mathbb{R}^2$ and $\mathbf{v}_{\max} \in \mathbb{R}^2$ be the lower-left and upper-right corners of this discrete velocity grid. The number of discrete velocities is N^2 , the velocity step is denoted by $(\Delta v_x, \Delta v_y) = (\mathbf{v}_{\max} - \mathbf{v}_{\min})/N$, and the p th velocity is $\mathbf{v}_p = \mathbf{v}_{\min} + (p_1 \Delta v_x, p_2 \Delta v_y)$, such that $p = p_2 N + p_1$ for all $(p_1, p_2) \in [0, N-1]^2$. The approximation of the distribution function is defined by $f_p(t, \mathbf{x}) = f(t, \mathbf{x}, \mathbf{v}_p)$. The set of equations (7) is discretized with respect to \mathbf{v} by the following set of $2N^2$ equations:

$$\begin{aligned} \frac{\partial}{\partial t} \int_S f_p \, dS + \int_{\partial S} (\mathbf{v}_p - \mathbf{w}) \cdot \mathbf{n} f_p \, dl &= \int_S \frac{1}{\tau} (F_p - f_p) \, dS, \\ \frac{\partial}{\partial t} \int_S g_p \, dS + \int_{\partial S} (\mathbf{v}_p - \mathbf{w}) \cdot \mathbf{n} g_p \, dl &= \int_S \frac{1}{\tau} (G_p - g_p) \, dS. \end{aligned} \quad (10)$$

The macroscopic quantities are computed with (6), where the integrals over \mathbb{R}^2 are approximated by a sum over the N^2 discrete velocity points. They are therefore given by

$$\begin{bmatrix} \rho \\ \rho \mathbf{u} \\ E \end{bmatrix} = \sum_{p=0}^{N^2-1} \begin{bmatrix} 1 \\ \mathbf{v}_p \\ \frac{1}{2} \|\mathbf{v}_p\|^2 \end{bmatrix} f_p \Delta v_x \Delta v_y + \sum_{p=0}^{N^2-1} \begin{bmatrix} 0 \\ \mathbf{0} \\ 1 \end{bmatrix} g_p \Delta v_x \Delta v_y, \quad (11)$$

$$\Sigma = \sum_{p=0}^{N^2-1} (\mathbf{v}_p - \mathbf{u}) \otimes (\mathbf{v}_p - \mathbf{u}) f_p \Delta v_x \Delta v_y. \quad (12)$$

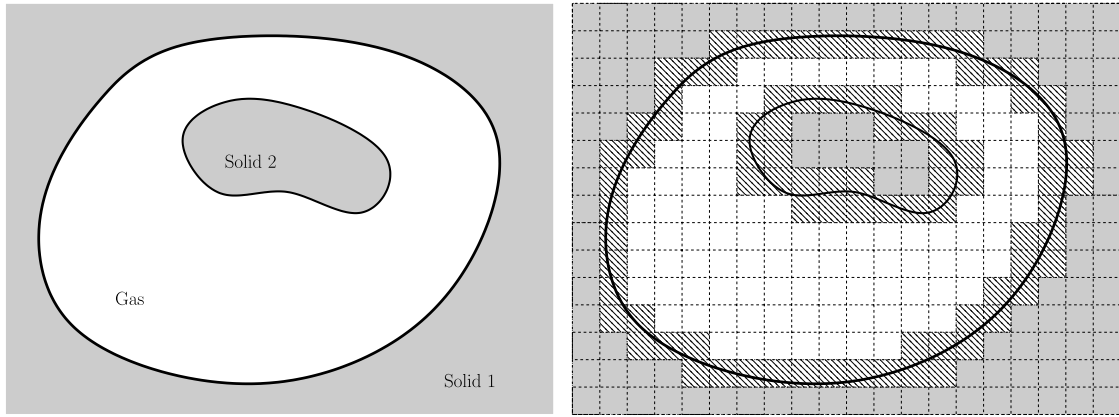


Fig. 1. Cells of the computational domain are classified in three categories: gas cells are represented in white, solid cells are shaded and cut cells are hatched.

Finally, the equilibrium functions F_p and G_p are computed with (8). However, the triplet (ρ, \mathbf{u}, T) used in these formulas is obtained with a Newton algorithm that preserves the discrete conservation of Boltzmann equation, rather than with direct computation (11) of the macroscopic quantities. Note that instead of using the algorithm of [29] which is based on entropic variables, we use the algorithm of Titarev [51].

3.2. Space discretization

3.2.1. Cartesian grid and cut cells

Let $\Omega = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ denote the space computational domain. It is discretized by a Cartesian grid of $(N_x + 1) \times (N_y + 1)$ points. Their coordinates are computed for all $(i, j) \in [0, N_x] \times [0, N_y]$ by $\mathbf{x}_{i+\frac{1}{2}, j+\frac{1}{2}} = \mathbf{x}_{\min} + (i\Delta x, j\Delta y)$, where $(\Delta x, \Delta y) = ([x_{\max} - x_{\min}]/N_x, [y_{\max} - y_{\min}]/N_y)$ and $\mathbf{x}_{\min} = (x_{\min}, y_{\min})$. The computational mesh is therefore made up by $N_x \times N_y$ rectangular cells: each cell is denoted by $\Omega_{i,j}$ and its center is $\mathbf{x}_{i,j}$.

Since the computational domain is rectangular, physical boundaries do not necessarily fit with the mesh boundary. In order to simulate arbitrary shaped objects, solid and gaseous domains $\Omega_s(t)$ and $\Omega_g(t)$ are introduced. They correspond to the solid and gaseous parts of the computational domain and hence $\Omega = \Omega_s(t) \cup \Omega_g(t)$. We point out that while Ω_s and Ω_g are time dependent, Ω is not. At time $t > 0$, a rectangular cell $\Omega_{i,j}$ can be in one of these three different states only:

- $\Omega_{i,j}$ is a *gas cell* if it is completely contained in the gaseous domain: $\Omega_{i,j} \cap \Omega_s(t) = \emptyset$.
- $\Omega_{i,j}$ is a *solid cell* if it is completely contained in the solid domain: $\Omega_{i,j} \cap \Omega_g(t) = \emptyset$.
- $\Omega_{i,j}$ is a *cut cell* if it is partially contained in the gaseous domain and partially contained in the solid domain: $\Omega_{i,j} \cap \Omega_s(t) \neq \emptyset$ and $\Omega_{i,j} \cap \Omega_g(t) \neq \emptyset$.

These three states of cells are shown in Fig. 1.

3.2.2. Virtual cells

To each cell $\Omega_{i,j}$ is now associated a virtual cell $\overline{\Omega}_{i,j}(t)$, which is the section of $\Omega_{i,j}$ contained in the gaseous domain: this reads $\overline{\Omega}_{i,j}(t) = \Omega_{i,j} \cap \Omega_g(t)$. Whatever the state of the cell (that is to say gas, solid or cut), it is defined with five virtual edges, whose lengths can be zero. Four of them, that are denoted by $L_{i\pm\frac{1}{2},j}(t)$ and $L_{i,j\pm\frac{1}{2}}(t)$, fit with the lines of the Cartesian mesh. The last one, denoted by $L_{i,j}(t)$, is a linear approximation of the solid boundary. If the virtual cell has less than five real edges, then at least one length is zero. Finally, we denote by $\overline{s}_{i,j}$ the area of the virtual cell, and $|L|$ denote the length of any edge L .

At a given time $t^n = n\Delta t$, all these parameters are denoted as follows:

$$\overline{\Omega}_{i,j}^n = \overline{\Omega}_{i,j}(t^n), \quad L_{i\pm\frac{1}{2},j}^n = L_{i\pm\frac{1}{2},j}(t^n), \quad L_{i,j\pm\frac{1}{2}}^n = L_{i,j\pm\frac{1}{2}}(t^n), \quad L_{i,j}^n = L_{i,j}(t^n), \quad \overline{s}_{i,j}^n = \overline{s}_{i,j}(t^n).$$

Note that a difficult problem in the cut cell method is that cut cells can take many different shapes (mainly in 3D), which can make the code very complex. A key element of our approach is that all these different shapes are treated generically by using this notion of virtual cell with its 5 virtual edges. Indeed, all the cells are treated in the same way, whatever their state (gas, solid, cut cell) or shape. The different parameters of the three cell states are summarized below, and we refer to Fig. 2 for three examples of cut cells:

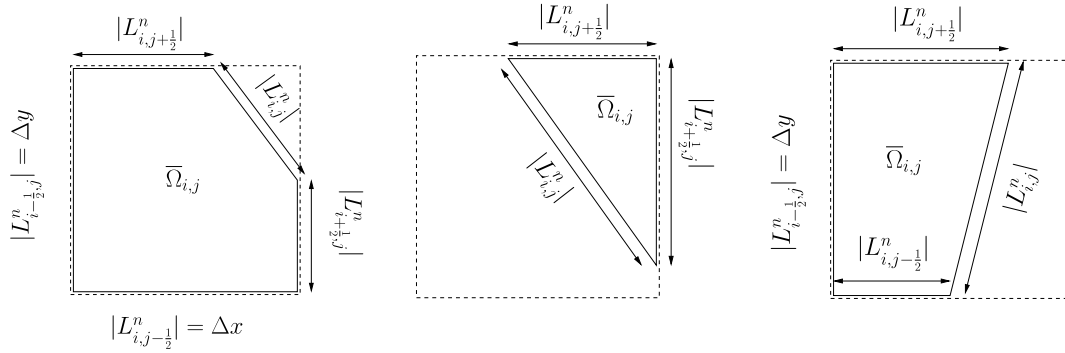


Fig. 2. Three examples of virtual cells: the cell $\Omega_{i,j}$ is drawn with the dashed line, while the corresponding virtual cell $\bar{\Omega}_{i,j}$ is drawn with the solid line. This virtual cell is a polygon with at most five edges. Left: five edges. Middle: three edges, while two virtual edges $L^n_{i,j-1/2}$ and $L^n_{i,j-1}$ have zero length. Right: four edges while the virtual edge $L^n_{i+1/2,j}$ has zero length.

- gas cell: $\bar{\Omega}_{i,j} = \Omega_{i,j}$, $|L^n_{i\pm 1/2,j}| = \Delta y$, $|L^n_{i,j\pm 1/2}| = \Delta x$, $|L^n_{i,j}| = 0$, $\bar{s}^n_{i,j} = \Delta x \Delta y$,
- solid cell: $\bar{\Omega}_{i,j} = \emptyset$, $|L^n_{i\pm 1/2,j}| = 0$, $|L^n_{i,j\pm 1/2}| = 0$, $|L^n_{i,j}| = 0$, $\bar{s}^n_{i,j} = 0$,
- cut cell: $\bar{\Omega}_{i,j}$ is a part of $\Omega_{i,j}$, and the five lengths of the virtual cell $|L^n_{i\pm 1/2,j}|$, $|L^n_{i,j\pm 1/2}|$, and $|L^n_{i,j}|$ can take any value between 0 and Δy , Δx , and $\sqrt{\Delta x^2 + \Delta y^2}$, respectively (see Fig. 2).

All these parameters are computed with a levelset technique, as explained in [Appendix A](#).

3.2.3. Control volumes

The notion of control volume is essential to avoid the use of very small virtual cells that would lead to prohibitively small time steps. The idea is to merge small virtual cells with larger neighboring cells when their areas are smaller than half of the area of a Cartesian cell. This technique is close to the one presented in [\[36,37\]](#) (see comments in [Appendix A.3](#)). There are other ways to proceed, see for instance [\[1,5\]](#).

The control volume is constructed by recursion: we look at a given cut cell $\Omega_{i,j}$ whose center is inside the solid domain. The corresponding virtual cell $\bar{\Omega}_{i,j}$ necessarily has an area smaller than $\frac{1}{2}\Delta x \Delta y$, and it has to be merged with one of its non-solid neighboring cells. This cell is chosen by looking at the largest non-solid edge of $\bar{\Omega}_{i,j}$: the neighboring cell that shares the same edge is chosen for merging (for instance $\bar{\Omega}_{i,j+1}$ in [Fig. 3](#), top). If the corresponding neighboring cell has its center inside the gas domain, then the algorithm is stopped and the resulting control volume contains two virtual cells. It happens sometimes that the neighboring virtual cell is also too small (its center is inside the solid domain too): in this case, the same algorithm is used recursively for this virtual cell. This merging procedure ensures that the area of the control volume is always greater than $\frac{1}{2}\Delta x \Delta y$.

It is convenient to denote by $\sigma^n_{i,j}$ the set of indices (i', j') such that all the virtual cells $\bar{\Omega}_{i',j'}(t)$ are merged together. For example, if $\bar{\Omega}_{i,j}$ and $\bar{\Omega}_{i,j+1}$ merge, then $\sigma^n_{i,j} = \{(i, j), (i, j+1)\}$.

The previous algorithm defines the control volume at time t^n . For $t > t^n$, the virtual cells change (since the solid boundary moves), and the control volume as well. For t between t^n and t^{n+1} the time dependent control volume $\mathcal{C}^n_{i,j}(t)$ is defined as follows:

$$\mathcal{C}^n_{i,j}(t) = \bigcup_{(i',j') \in \sigma^n_{i,j}} \bar{\Omega}_{i',j'}(t). \quad (13)$$

In other words, the set of virtual cells selected at time t^n for merging defines the control volume up to t^{n+1} . We point out that if the shape of the virtual cells (and hence of the control volume) can vary in time, the set $\sigma_{i,j}$ is fixed for $t \in [t^n, t^{n+1}]$.

At time t^{n+1} , we have to take into account that there are new virtual cells, some others have disappeared, and the shape of all of them have changed: therefore, a new control volume, denoted by $\mathcal{C}^{n+1}_{i,j}(t^{n+1})$, has to be constructed (by the previous recursive algorithm). We refer to [Fig. 3](#) for an illustration of this algorithm.

While the previous procedure might look complicated, note that most of the virtual cells do not merge, and hence $\mathcal{C}^n_{i,j}(t) = \bar{\Omega}_{i,j}(t)$ for most of them.

The area of the control volumes $\mathcal{C}^n_{i,j}(t^n)$ and $\mathcal{C}^n_{i,j}(t^{n+1})$ are computed easily with

$$s^n_{i,j} = \sum_{(i',j') \in \sigma^n_{i,j}} \bar{s}^n_{i',j'} \quad \text{and} \quad s^{n+1,*}_{i,j} = \sum_{(i',j') \in \sigma^n_{i,j}} \bar{s}^{n+1}_{i',j'}$$

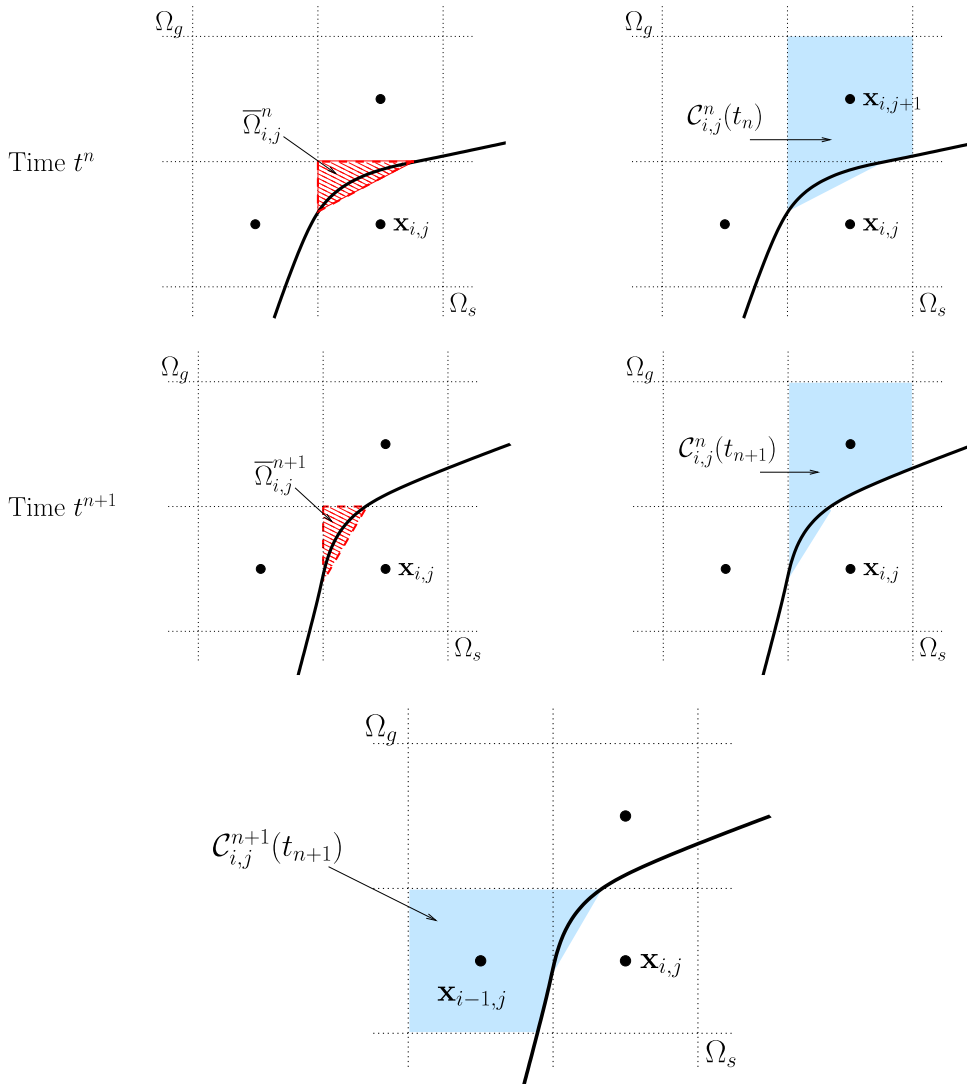


Fig. 3. Top-left: the virtual cell $\overline{\Omega}_{i,j}^n$ (hatched) has to merge with $\overline{\Omega}_{i,j+1}^n$ because $|L_{i,j+\frac{1}{2}}^n| > |L_{i-\frac{1}{2},j}^n|$. Top-right: this results in the control volume $C_{i,j}^n(t^n)$, in blue. Middle-left: at time t^{n+1} , the solid boundary has slightly turned counter clock-wise and the virtual cell $\overline{\Omega}_{i,j}^{n+1}$ is smaller. Middle-right: the control volume at this time now is $C_{i,j}^n(t^{n+1})$, in blue. Bottom: since the larger edge of the virtual cell $\overline{\Omega}_{i,j}^{n+1}$ now is $|L_{i-\frac{1}{2},j}^{n+1}|$, it has to merge with $\overline{\Omega}_{i-1,j}^{n+1}$, and this results in the new control volume $C_{i,j}^{n+1}(t^{n+1})$, which is different from $C_{i,j}^n(t^{n+1})$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Note that there is a kind of redundancy with this approach: indeed, in the previous example, since $\overline{\Omega}_{i,j}^n$ and $\overline{\Omega}_{i,j+1}^n$ belong to the same control volume, then the control volumes $C_{i,j}^n(t)$ and $C_{i,j+1}^n(t)$ are the same. However, this makes the implementation much simpler, while the overhead of the computational time is very small: indeed, the number of merged cut cells is very small as compared to the number of gas cells.

3.3. Numerical scheme

3.3.1. A finite volume scheme

From now on, calculations are detailed with the reduced distribution function f . The same analysis can be done with g . The cut cell method is based on a finite volume scheme. One time iteration (which will be divided into three steps) consists in computing the average value $\overline{f}_{i,j,p}^{n+1}$ of the distribution function over the virtual cell $\overline{\Omega}_{i,j}^{n+1}$ from the average value $\overline{f}_{i,j,p}^n$. Moreover, let $f_{i,j,p}^n$ and $f_{i,j,p}^{n+1,*}$ stand for the average values of the distribution function over the control volumes $C_{i,j}^n(t^n)$ and $C_{i,j}^n(t^{n+1})$.

The first step of the method is the computation of $f_{i,j,p}^n$ through the average values of f over the virtual cells included in $\mathcal{C}_{i,j}^n(t^n)$. Definitions of $\overline{\Omega}_{i,j}^n$ and $\mathcal{C}_{i,j}^n$ readily lead to

$$f_{i,j,p}^n := \frac{1}{s_{i,j}^n} \sum_{(i',j') \in \sigma_{i,j}^n} \overline{s}_{i',j'}^n \overline{f}_{i',j',p}^n. \quad (14)$$

The second step is the time integration of the integral form of the Boltzmann equation (7) between t^n and t^{n+1} : this relation is applied by choosing the surface $S(t)$ as the control volume $\mathcal{C}_{i,j}^n(t)$. This gives

$$\begin{aligned} s_{i,j}^{n+1,*} f_{i,j,p}^{n+1,*} - s_{i,j}^n f_{i,j,p}^n &:= - \int_{t^n}^{t^{n+1}} \int_{\partial \mathcal{C}_{i,j}^n(t)} (\mathbf{v}_p - \mathbf{w}(t)) \cdot \mathbf{n}(t) f_p(t, \mathbf{x}) \, dl \, dt \\ &+ \int_{t^n}^{t^{n+1}} \int_{\mathcal{C}_{i,j}^n(t)} \frac{1}{\tau(t, \mathbf{x})} (F_p(t, \mathbf{x}) - f_p(t, \mathbf{x})) \, dS \, dt, \end{aligned} \quad (15)$$

The transport integral can be computed as follows. First, definition (13) implies that the integral over $\partial \mathcal{C}_{i,j}^n(t)$ is the sum of the integrals over the contours of all the virtual cells $\overline{\Omega}_{i',j'}(t)$ that merge into the control volume $\mathcal{C}_{i,j}^n(t)$. Moreover, the velocity $\mathbf{w} \cdot \mathbf{n}$ is zero for the four edges that fit with the Cartesian mesh lines, while this velocity \mathbf{w} is \mathbf{u}_w for the last edge of the cell, since it fits with the solid boundary. Finally, the transport integral is written as:

$$\int_{\partial \mathcal{C}_{i,j}^n(t)} (\mathbf{v}_p - \mathbf{w}(t)) \cdot \mathbf{n}(t) f_p(t, \mathbf{x}) \, dl = \sum_{(i',j') \in \sigma_{i,j}^n} \left[\int_{L_{i',j'}(t)} (\mathbf{v}_p - \mathbf{u}_w(t)) \cdot \mathbf{n}(t) f_p(t, \mathbf{x}) \, dl + \int_{L(t)} \mathbf{v}_p \cdot \mathbf{n}_w(t) f_p(t, \mathbf{x}) \, dl \right], \quad (16)$$

where $L = L_{i'+\frac{1}{2},j'} \cup L_{i'+\frac{1}{2},j'} \cup L_{i',j'-\frac{1}{2}} \cup L_{i',j'-\frac{1}{2}}$ is the union of the four edges that fit with the Cartesian mesh lines. A backward Euler method is applied in order to get a first order approximation of the Boltzmann equation. This means that the time integral in (15) is approximated by the rectangle rule and we find that equation (15) becomes:

$$\begin{aligned} f_{i,j,p}^{n+1,*} &= \frac{s_{i,j}^n}{s_{i,j}^{n+1,*}} f_{i,j,p}^n - \frac{\Delta t}{s_{i,j}^{n+1,*}} \sum_{(i',j') \in \sigma_{i,j}^n} \left[\left(\mathcal{F}_{i'+\frac{1}{2},j',p}^n - \mathcal{F}_{i'-\frac{1}{2},j',p}^n \right) + \left(\mathcal{F}_{i',j'+\frac{1}{2},p}^n - \mathcal{F}_{i',j'-\frac{1}{2},p}^n \right) + \mathcal{F}_{i',j',p}^n \right] \\ &+ \frac{s_{i,j}^n}{s_{i,j}^{n+1,*}} \frac{\Delta t}{\tau_{i,j}^n} (F_{i,j,p}^n - f_{i,j,p}^n), \end{aligned} \quad (17)$$

where $\mathcal{F}_{i\pm\frac{1}{2},j,p}$, $\mathcal{F}_{i,j\pm\frac{1}{2},p}$ and $\mathcal{F}_{i,j,p}$ are the numerical fluxes across the five edges of the virtual cell that are computed with a standard upwind scheme:

$$\begin{aligned} \mathcal{F}_{i+\frac{1}{2},j,p}^n &:= |L_{i+\frac{1}{2},j}^n| (\min(v_{p1}, 0) f_{i+1,j,p}^n + \max(v_{p1}, 0) f_{i,j,p}^n), \\ \mathcal{F}_{i,j+\frac{1}{2},p}^n &:= |L_{i,j+\frac{1}{2}}^n| [\min(v_{p2}, 0) f_{i,j+1,p}^n + \max(v_{p2}, 0) f_{i,j,p}^n], \\ \mathcal{F}_{i,j,p}^n &:= |L_{i,j}^n| [\min([\mathbf{v}_p - \mathbf{u}_w(t^n, \mathbf{r}_{i,j}^n)] \cdot \mathbf{n}_w(t^n, \mathbf{r}_{i,j}^n), 0) f_w(t^n, \mathbf{r}_{i,j}^n, \mathbf{v}_p) \\ &+ \max([\mathbf{v}_p - \mathbf{u}_w(t^n, \mathbf{r}_{i,j}^n)] \cdot \mathbf{n}_w(t^n, \mathbf{r}_{i,j}^n), 0) f_{i,j,p}^n], \end{aligned} \quad (18)$$

where $\mathbf{r}_{i,j}^n$ is the center of $L_{i,j}^n$. It is recalled that v_{p1} and v_{p2} are the coordinates of the p th microscopic velocity, i.e. $\mathbf{v}_p = (v_{p1}, v_{p2})$. Moreover, $\mathbf{n}_w(t^n, \mathbf{r}_{i,j}^n)$ is the outward normal to the edge $L_{i,j}^n$, that fit with the physical boundary. The computation of the velocity $\mathbf{u}_w(t^n, \mathbf{r}_{i,j}^n)$ of the boundary is detailed in section 3.3.2. Note that when the Knudsen number is very small, the previous scheme (17), which is explicit, is too expensive: the CFL condition induces a time step which is of the order of the relaxation time. It is now standard in kinetic theory to use instead an implicit/explicit scheme (see for instance [35] for the BGK equation and [17] for other methods). The idea is to use an implicit scheme for the stiff collision part, while the transport part is still approximated by an explicit scheme. This kind of scheme is easily extended to the cut cell method. Finally the discrete boundary condition is similar to its continuous form (9), that is to say:

$$f_w(t^n, \mathbf{r}_{i,j}^n \in \Gamma, \mathbf{v}_p \in \mathcal{V}_{in}) = \phi M[1, \mathbf{u}_w(t^n, \mathbf{r}_{i,j}^n), T_w], \quad (19)$$

where ϕ is given by

$$\phi = - \frac{\sum_{\mathbf{v}_p \in \mathcal{V}_{\text{out}}} (\mathbf{v}_p - \mathbf{u}_w(t^n, \mathbf{r}_{i,j}^n)) \cdot \mathbf{n}_w(t^n, \mathbf{r}_{i,j}^n) f_{i,j,p}^n \Delta v_x \Delta v_y}{\sum_{\mathbf{v}_p \in \mathcal{V}_{\text{in}}} (\mathbf{v}_p - \mathbf{u}_w(t^n, \mathbf{r}_{i,j}^n)) \cdot \mathbf{n}_w(t^n, \mathbf{r}_{i,j}^n) M[1, \mathbf{u}_w(t^n, \mathbf{r}_{i,j}^n), T_w] \Delta v_x \Delta v_y}.$$

Note that the boundary condition is defined only for the velocities $\mathbf{v}_p \in \mathcal{V}_{\text{in}} = \{\mathbf{v} | (\mathbf{v} - \mathbf{u}_w) \cdot \mathbf{n}_w < 0\}$, which is compatible with the definition of the numerical boundary flux $\mathcal{F}_{i,j,p}^n$ (see (18)). The third step of the method is the computation of $\bar{f}_{i,j,p}^{n+1}$, the average value of f in the new virtual cell $\bar{\Omega}_{i,j}^{n+1}$. This is done by distributing the value $f_{i,j,p}^{n+1,*}$ given by (17) to the cells $\bar{\Omega}_{i,j}^{n+1}$ merged into the control volume $\mathcal{C}_{i,j}^n(t^{n+1})$:

$$\bar{f}_{i,j,p}^{n+1} := f_{i,j,p}^{n+1,*}. \quad (20)$$

A first summary of the cut cell method is given below:

1. The virtual cells $\bar{\Omega}_{i,j}(t)$ merge into some control volumes $\mathcal{C}_{i,j}(t)$ and the values $f_{i,j,p}^n$ are computed with (14).
2. The numerical scheme (17) is applied in order to compute the values $f_{i,j,p}^{n+1,*}$.
3. The values $\bar{f}_{i,j,p}^{n+1}$ are updated with formula (20).

The three steps of the method are illustrated in Fig. 4 for various situations. Note that because of the merging procedure, there is no issue of appearing/disappearing gas cells: in other words, a small virtual cell necessarily merges with a larger cell before it disappears, and conversely, the average value of f in a new appearing virtual cell is naturally defined through steps 2 and 3. This ensures that the method is conservative, see section 3.4.

It remains to explain how the motion of the solid body is taken into account: this is done in section 3.3.2. The complete scheme is summarized in section 3.3.3.

3.3.2. Motion of the solid body

The motion of the solid body is taken into account in the scheme by the variation of the area of the control volume (from $s_{i,j}^n$ to $s_{i,j}^{n+1,*}$) and by the velocity $\mathbf{u}_w(t^n, \mathbf{r}_{i,j}^n)$ of the solid boundary, see (17) and (18). These quantities are computed as follows.

Let $\mathbf{c}(t)$ and $\theta(t)$ be the coordinates of the center of mass and the inclination of the solid body. Its translational and rotational velocities are then denoted by $\dot{\mathbf{c}}(t)$ and $\dot{\theta}(t)$. The motion of the solid body, with mass m and moment of inertia J , is modeled by the Newton's laws of motion that are discretized as follows:

$$\mathbf{c}^{n+1} = \mathbf{c}^n + \Delta t \dot{\mathbf{c}}^n \quad \text{and} \quad \theta^{n+1} = \theta^n + \Delta t \dot{\theta}^n, \quad (21)$$

$$\dot{\mathbf{c}}^{n+1} = \dot{\mathbf{c}}^n + \Delta t \mathbf{F}^n / m \quad \text{and} \quad \dot{\theta}^{n+1} = \dot{\theta}^n + \Delta t T^n / J, \quad (22)$$

where \mathbf{F} and T are the force and torque exerted by the gas on the solid body. They can be computed by using the stress tensor Σ_w at the boundary with the formula

$$\mathbf{F} = \int_{\partial\Omega_g} \Sigma_w \mathbf{n}_w dl \quad \text{and} \quad T = \int_{\partial\Omega_g} (\mathbf{x} - \mathbf{c}) \times (\Sigma_w \mathbf{n}_w) dl.$$

These relations can be approximated by any quadrature formula, and we find it convenient to use a summation over all the cells of the computational domain to avoid too many tests. This yields:

$$\mathbf{F}^n = \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \Sigma_w(t^n, \mathbf{r}_{i,j}^n) \mathbf{n}_w(t^n, \mathbf{r}_{i,j}^n) |L_{i,j}^n|, \quad (23)$$

$$T^n = \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \left((\mathbf{r}_{i,j}^n - \mathbf{c}^n) \times (\Sigma_w(t^n, \mathbf{r}_{i,j}^n) \mathbf{n}_w(t^n, \mathbf{r}_{i,j}^n)) \right) |L_{i,j}^n|. \quad (24)$$

Since $|L_{i,j}^n|$ is non-zero only for solid edges of cut cells, these formula are consistent approximations of the previous definition.

Moreover, while the stress tensor is defined by (12), the boundary condition has to be taken into account to define the distribution of incoming velocities, and we set

$$\begin{aligned} \Sigma_w(t^n, \mathbf{r}_{i,j}^n) &= \sum_{\mathbf{v}_p \in \mathcal{V}_{\text{in}}} (\mathbf{v}_p - \mathbf{u}_w(t^n, \mathbf{r}_{i,j}^n)) \otimes (\mathbf{v}_p - \mathbf{u}_w(t^n, \mathbf{r}_{i,j}^n)) f_w(t^n, \mathbf{r}_{i,j}^n, \mathbf{v}_p) \Delta v_x \Delta v_y \\ &+ \sum_{\mathbf{v}_p \in \mathcal{V}_{\text{out}}} (\mathbf{v}_p - \mathbf{u}_w(t^n, \mathbf{r}_{i,j}^n)) \otimes (\mathbf{v}_p - \mathbf{u}_w(t^n, \mathbf{r}_{i,j}^n)) f_{i,j,p}^n \Delta v_x \Delta v_y. \end{aligned} \quad (25)$$

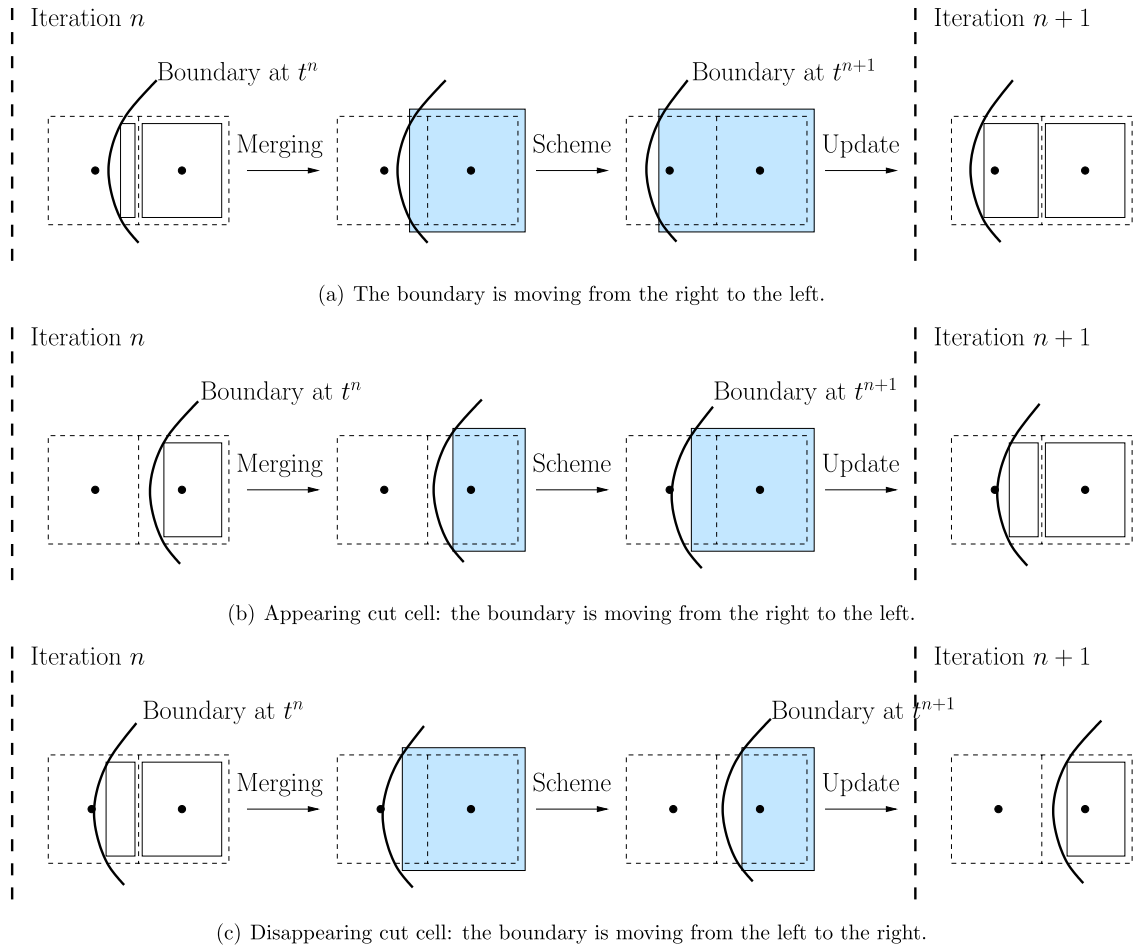


Fig. 4. Illustration of the three steps of the cut cell method. The cells $\Omega_{i,j}$ are drawn with the dashed line and the virtual cells $\bar{\Omega}_{i,j}(t)$ with the solid line. The control volume $C_{i,j}(t)$ is shaded.

The boundary condition $f_w(t^n, \mathbf{r}_{i,j}^n, \mathbf{v}_p)$ is defined by (19). The new velocity of the wall is finally computed with

$$\mathbf{u}_w(t^{n+1}, \mathbf{r}_{i,j}^{n+1}) = \dot{\mathbf{c}}^{n+1} + (\mathbf{r}_{i,j}^{n+1} - \mathbf{c}^{n+1})^\perp \dot{\theta}^{n+1}, \quad (26)$$

where a^\perp is the vector obtained after a rotation of 90 degrees of any vector a in the counter-clockwise sense.

3.3.3. Summary of the numerical scheme

For the convenience of the reader, the different steps of the complete numerical scheme are summarized below.

We assume that, at time t^n , all the following quantities are known: the average value of the distribution function $\bar{f}_{i,j,p}^n$ in each virtual cell $\bar{\Omega}_{i,j}$, the parameters of position (\mathbf{c}^n, θ^n) and velocity $(\dot{\mathbf{c}}^n, \dot{\theta}^n)$ of the solid body, and hence the wall velocity $\mathbf{u}_w(t^n, \mathbf{r}_{i,j}^n)$. One time iteration of the numerical scheme is decomposed into the following 7 steps:

1. The position of the solid body at t^{n+1} is computed with (21).
2. The virtual cells are arranged into control volumes $C_{i,j}^n(t_n)$ following the rule given in section 3.2.3. The distribution function is averaged over the control volumes $C_{i,j}^n(t^n)$.
3. The virtual cells and control volumes are moved according to the new position computed at step 1. The areas $s_{i,j}^n$ and $s_{i,j}^{n+1,*}$ are computed by using a level set method (see Appendix A for more details).
4. The value of the distribution at each solid boundaries is computed through boundary condition (19).
5. The stress tensor Σ_w at each solid boundaries is computed with (25), which gives force \mathbf{F}^n and torque T^n with (23) and (24). Then the translational and rotational velocities are computed at time t^{n+1} by the discrete Newton laws (22), and finally the new wall velocity is computed with (26).

6. Scheme (17) is used to pass from $f_{i,j,p}^n$ (the average value of f at time t^n in the control volume $\mathcal{C}_{i,j}^n(t^n)$) to $f_{i,j,p}^{n+1,*}$ (the average value of f at time t^{n+1} in the control volume $\mathcal{C}_{i,j}^n(t^{n+1})$).
7. The values $\bar{f}_{i,j,p}^{n+1}$ of f at time t^{n+1} in each virtual cell that are merged into the control volume $\mathcal{C}_{i,j}^n(t^{n+1})$ are updated with (20).

3.4. Properties of the scheme

3.4.1. Positivity

Standard arguments show that the explicit scheme (17) preserves the positivity of the solution if Δt satisfies the following CFL condition:

$$\Delta t \left(\max_{i,j} \left[\frac{1}{\tau_{i,j}^n} \right] + \max_{i,j,p} \left[\frac{\phi_{i,j,p}^n}{s_{i,j}^n} \right] \right) \leq 1, \quad (27)$$

where

$$\begin{aligned} \phi_{i,j,p} = & \sum_{(i',j') \in \sigma_{i,j}^n} \left(|L_{i'+\frac{1}{2},j'}| v_{p1}^+ - |L_{i'-\frac{1}{2},j'}| v_{p1}^- + |L_{i',j'+\frac{1}{2}}| v_{p2}^+ - |L_{i',j'-\frac{1}{2}}| v_{p2}^- \right. \\ & \left. + |L_{i',j'}| ((\mathbf{v}_p - \mathbf{u}_w(t^n, \mathbf{r}_{i,j}^n)) \cdot \mathbf{n}_w(t^n, \mathbf{r}_{i,j}^n))^+ \right) \end{aligned} \quad (28)$$

and $v^+ = \max(v, 0)$, $v^- = \min(v, 0)$. For a correct description of the flow, it is necessary that the discrete velocity grid contains the solid velocity \mathbf{u}_w at any time (since the diffuse boundary condition produces particles with \mathbf{u}_w as mean velocity). Under this assumption, it can be proved that $\phi_{i,j,p} \leq C(v_{x,\max}/\Delta x + v_{y,\max}/\Delta y)$, where $C = 1/2$. This gives a simple CFL condition, and in practice, it is relaxed by taking $C = 0.9$ without any stability problems

3.4.2. Conservation

Since our scheme is a finite volume method in which a conservative reflexion boundary condition is applied to compute the numerical fluxes at each solid edge, it is naturally conservative, as it is proved below.

Let M^n be the total mass of gas in the gas domain Ω_g at time t^n . It is convenient to write the total mass at time t^{n+1} as

$$M^{n+1} = \sum_{i,j=1}^{N_x, N_y} \delta_{i,j}^n s_{i,j}^{n+1,*} \sum_{p=0}^{N^2-1} f_{i,j,p}^{n+1,*} \Delta v_x \Delta v_y, \quad (29)$$

where $\delta_{i,j}^n = 1$ if $\mathbf{x}_{i,j} \in \Omega_g$ and 0 else: this function allows to take into account merged cells of a same control volume only once. Indeed, there is only one couple of indices (i', j') in $\sigma_{i,j}^n$ for which $\mathbf{x}_{i',j'}$ is inside the gas domain.

Then $f_{i,j,p}^{n+1,*}$ is replaced by its value given by (17) and we get

$$M^{n+1} = M^n + \sum_{i,j=1}^{N_x, N_y} \sum_{p=0}^{N^2-1} \Delta t \mathcal{F}_{i,j,k}^n \Delta v_x \Delta v_y.$$

Indeed, opposite fluxes across same Cartesian edges cancel out, the velocity sum of the collision operator is zero, and there remains only the numerical fluxes $\mathcal{F}_{i,j,k}^n$ across the solid edge of cut cells. By using the boundary condition, the velocity sum of such fluxes gives

$$\begin{aligned} \sum_{p=0}^{N^2-1} \mathcal{F}_{i,j,p}^n \Delta v_x \Delta v_y &= \sum_{\mathbf{v}_p \in \mathcal{V}_{\text{in}}} (\mathbf{v}_p - \mathbf{u}_w(t^n, \mathbf{r}_{i,j}^n)) \cdot \mathbf{n}_w(t^n, \mathbf{r}_{i,j}^n) f_w(t^n, \mathbf{r}_{i,j}^n, \mathbf{v}_p) |L_{i,j}^n| \Delta v_x \Delta v_y \\ &\quad + \sum_{\mathbf{v}_p \in \mathcal{V}_{\text{out}}} (\mathbf{v}_p - \mathbf{u}_w(t^n, \mathbf{r}_{i,j}^n)) \cdot \mathbf{n}_w(t^n, \mathbf{r}_{i,j}^n) f_{i,j,p}^n |L_{i,j}^n| \Delta v_x \Delta v_y \\ &= 0. \end{aligned}$$

This shows that $M^{n+1} = M^n$ and concludes the proof.

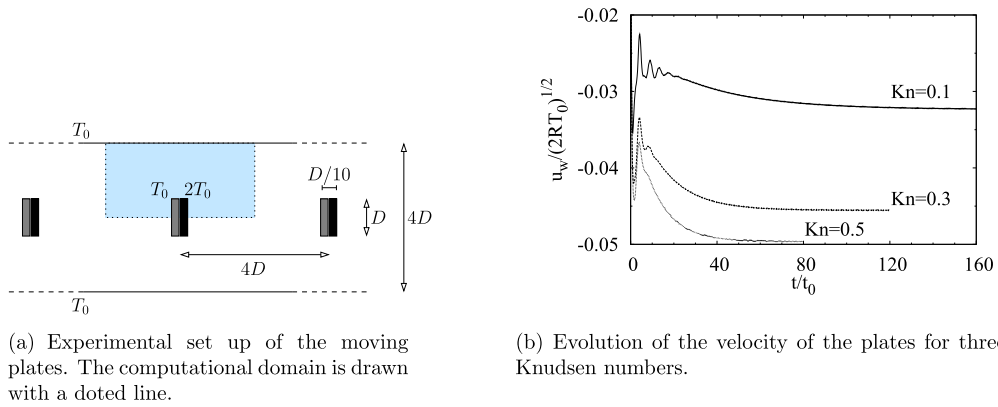


Fig. 5. Translational plates under the radiometric effect.

Finally, note that it is standard from the positivity and conservation properties to conclude that the scheme is L^1 stable.

4. Numerical results

4.1. Translational motion under radiometric effect

An infinite set of thin plates of height D is located in an infinite channel of width $4D$. The distance between two plate centers is $4D$. This experiment has been numerically investigated in [47,48], where the plates are infinitely thin (their thickness is zero). The temperature of the right side of a plate is twice as the temperature of its left side. This induces a force that can be interpreted as the difference between the radiometric force and gas friction, and make the plates move. We point out that all the plates move at the same velocity. After a while, the radiometric force is balanced with gas friction and the total force decreases to zero. At this time, the plates have reached their stationary velocity. In [47,48] the ES-BGK model is used and simulations are made in a moving reference frame in which the channel velocity is positive while the plates are motionless. The total force applied on the plates is zero for a specific channel velocity which corresponds to the stationary velocity of the plates. In our work, the BGK model is used to describe the behavior of the gas and plates are $D/10$ thick (see Fig. 5(a)). The transient motion of the plates (i.e. plates velocity before the stationary state is reached) can be simulated with the cut cell method in a fixed frame of reference. In that case, the motion depends on the mass of a plate. If we denote by ρ_0 the initial density of the gas, this mass is set to $m = \rho_0 D^2/2$. The computational domain is a rectangle of size $4D \times 2D$ that describes the upper part of the channel (Fig. 5(a)). The left and right boundary conditions are periodic so as to simulate the infinite channel. The bottom of the computational domain fits with the center of the channel: specular-reflection boundary condition is applied to take into account the symmetry. At last, the top of the domain corresponds to the wall of the channel, the standard diffuse boundary condition is used. During a whole simulation, there is always one and only one plate in the computational domain: when a plate goes out, another one comes in. Note that solid and cut cells only appear near this plate. To take its temperature into account, the boundary of the plate is modeled with the diffuse-reflection condition. Finally, the relaxation time and Knudsen number are defined by the relations:

$$\tau = \frac{\mu}{\rho RT_0} \quad \text{and} \quad \text{Kn} = \left[\frac{2}{\sqrt{\pi}} \frac{\sqrt{2RT_0}}{\rho_0 RT_0 / \mu} \right] / D,$$

where μ is the viscosity of the gas and T_0 the initial temperature of the gas.

Our simulations have been done for a wide range of Knudsen numbers from 10^{-3} to 1. Converged results are obtained for a velocity grid that contains from 20^2 to 40^2 points (this depends on the Knudsen number) and for a spatial mesh made up of 400×200 cells, which means that a plate encloses 10 cells. For coarser grids, the number of cells enclosed in the plate is too small to capture the shape of its edges with enough accuracy. As expected, the magnitude of the velocity of the plates increases until they reach their final velocity, as illustrated for three different Knudsen numbers on Fig. 5(b).

The variation of the stationary velocity of the plates is plotted as function of Kn on Fig. 6. For small Knudsen numbers, the final velocity seems to be proportional to $\sqrt{\text{Kn}}$ while this velocity tends to a constant for high Knudsen number. These simulations show a good agreement between our results and the results obtained in [47,48].

4.2. The Crookes radiometer

The Crookes radiometer was invented by Crookes in 1874 [14]: it is a glass globe containing four vanes immersed in a low pressure gas. Each vane has one black side and one shiny side, and when the globe is exposed to light, the vanes rotate. This was first understood as a rarefied gas dynamics effect by Reynolds [39], but there are still discussions on the order of

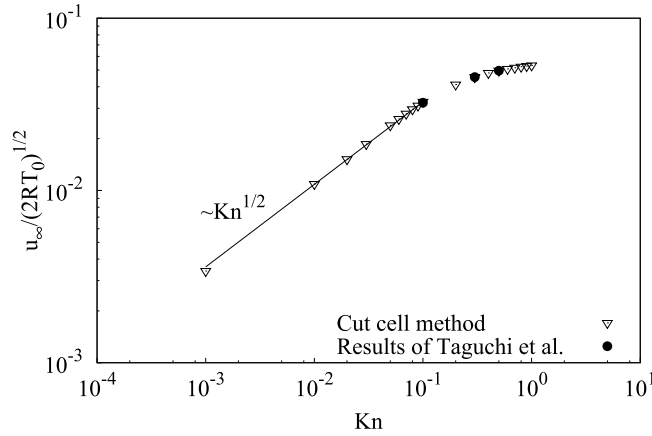
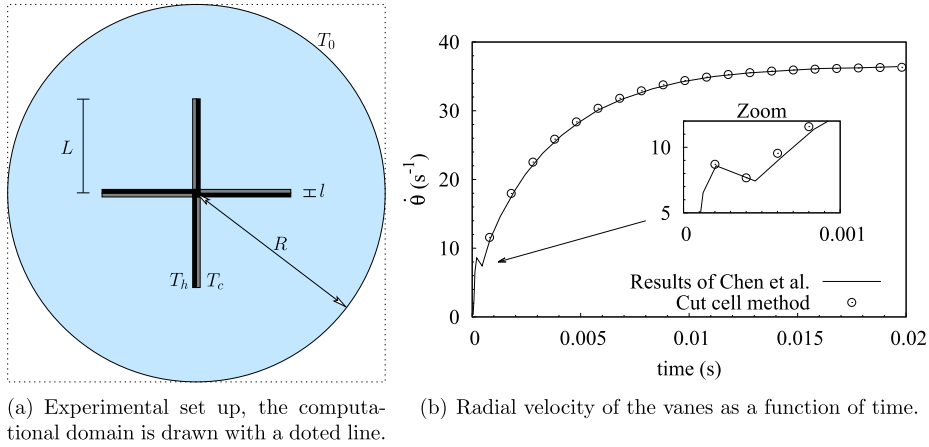


Fig. 6. Translational plates under the radiometric effect: stationary velocity of the plates as a function of the Knudsen number, comparison with Taguchi et al. [47,48].



(a) Experimental set up, the computational domain is drawn with a dotted line.

(b) Radial velocity of the vanes as a function of time.

Fig. 7. The 2D Crookes radiometer.

magnitudes of the forces involved in this device. We refer to the recent review of Ketsdever et al. [26] for historical details. Recently, numerical simulations improved the understanding of the radiometric effect, like in [46,47,42].

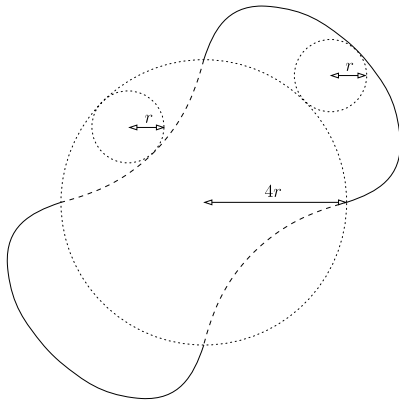
The dynamical acceleration process of the vanes has been recently studied in [10]: it uses the unified gas-kinetic scheme combined with a moving mesh approach [11] to simulate the motion of the vanes. In this case, the moving mesh approach is very convenient because the initial mesh just rotates without distortions. In this section, we show that the same results as [10] can be obtained with our cut cell method.

Since this test is for illustrating our 2D method, the device simulated here is a 2D radiometer composed of endless vanes immersed in an unbounded cylinder of radius $R = 20$ cm. The length of a vane is $L = 0.1$ cm, its thickness is $l = 0.01$ cm (Fig. 7(a)) and its rotational moment of inertia is $J = 4.9 \times 10^{-9}$ kg m². The temperature T_h of the black side of the vane is supposed to be higher than the temperature T_c of its shiny side. These temperatures are set to $T_h = 400$ K and $T_c = 350$ K while the temperature T_0 of the globe is equal to $T_0 = 300$ K. Note that all the boundary conditions are computed with diffuse reflection conditions. In order to compare our results to [10], we take the same Shakhov relaxation model with $Pr = 2/3$. Moreover, the relaxation time is computed by the equation

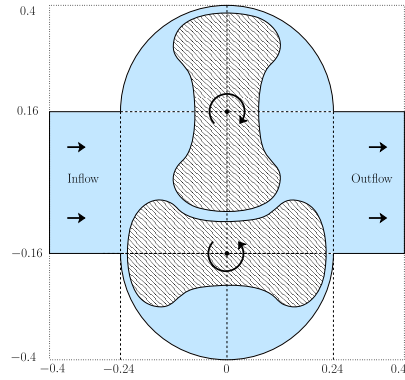
$$\tau = \frac{\mu}{P} \cdot \left(\frac{T}{T_0} \right)^\omega, \quad (30)$$

where μ is determined by the hard sphere model for argon which yields $\mu = 1.678 \times 10^{-5}$ N s m⁻² and $\omega = 0.68$. Finally, we set the initial density $\rho_0 = 8.582 \cdot 10^{-6}$ kg m⁻³ and get a Knudsen number based on the length of a vane equal to 0.1.

Converged results are obtained with a 30² points velocity grid and 400² cells spatial mesh. This large number of spatial cells is required to describe the shape of the vanes with enough accuracy. We plot in Fig. 7(b) the radial velocity of the vanes as a function of time and compare our results to those obtained in [10]. The results are in very good agreement.



(a) Geometry of a lobe. The plain line is an epicycloidal and the dashed line is an hypocycloidal.



(b) Pump geometry. All the units are given in meter. The computational domain is the square $[-0.4, 0.4]^2$.

Fig. 8. Roots blower.

4.3. Roots blower

Various kinds of vacuum pumps are used in industrial processes. A common one is the Roots blower. It is made of several lobes that rotate simultaneously. As a result, the gas is trapped by the lobes at one side and then carried to the other side of the pump. The simplest shape of Roots blower is a two-lobed rotor. In this case the profile of a lobe is defined by sections of epicycloid and hypocycloid (Fig. 8(a)). In parametric coordinates, this profile is given for all $\theta \in [-\pi, -\frac{\pi}{2}] \cup [0, \frac{\pi}{2}]$ by the epicycloidal equation

$$\begin{cases} x(\theta) = 5r \cos(\theta) - r \sin(5\theta), \\ y(\theta) = 5r \sin(\theta) - r \sin(5\theta), \end{cases}$$

and for all $\theta \in [-\frac{\pi}{2}, 0] \cup [\frac{\pi}{2}, \pi]$ by the hypocycloidal equation

$$\begin{cases} x(\theta) = 3r \cos(\theta) + r \sin(3\theta), \\ y(\theta) = 3r \sin(\theta) - r \sin(3\theta), \end{cases}$$

where r is the radius of the small generating circle that rolls on the large circle of radius $4r$. In our simulation, we took $r = 3.8$ cm for both lobes. The full geometry of the pump is detailed in Fig. 8. Note that lobes are not in contact: the minimum distance between them is $d = 1.6$ cm.

Since this type of pump mostly operates in atmospheric environment, the initial conditions are given by $T_0 = 300$ K, $P_0 = 10^5$ Pa, $\vec{u}_0 = \vec{0}$, and the considered gas is argon. The relaxation time is computed with formula (30) where the viscosity coefficient and index for argon are provided by Bird [8], that is $\mu = 2.117 \times 10^{-27}$ N s m⁻² and $\omega = 0.81$. Because there is no friction between the lobes, a Roots blower can proceed at a rotary speed that range from 1500 rpm to 3000 rpm (~ 150 rad s⁻¹ to 300 rad s⁻¹). For the following simulations, the rotational velocity of the lobes is set to $\dot{\theta} = \pm 200$ rad s⁻¹. Note that since the velocity is imposed here, step 5 of the algorithm is not used (see section 3.3.3). In practice, the temperature of the lobes tends to increase because of the mechanical heating due to their high rotary speed. However, to make it simpler, the wall of the Roots blowers and its lobes are modeled with diffuse boundary conditions with constant temperature T_0 . At the left side of the pump, it is assumed that all the gas surrounding the computational domain is in the same state as the gas located at the inlet. This can be modeled by a Neumann boundary condition. At the outlet (right side of the pump), we assume that the gas is released in the atmosphere, and hence the boundary condition is given by a Maxwellian built with the initial conditions P_0 , T_0 and \vec{u}_0 .

Pressure contours at several times are shown Fig. 9. We observe that the pressure at outlet does not change while the pressure at inlet decreases, as it is shown in Fig. 9(d). We stop the computation at $t = 0.1$ s. At this time, the inlet pressure is 80% of the initial pressure, which means that we get a pressure drop of 20%.

We point out that this simulation is only a qualitative analysis. Kinetic equations are not really relevant here because the Knudsen number is very small: $\text{Kn} \approx 3 \times 10^{-5}$ for a reference length equal to the distance d between the two lobes (it would be larger with a smaller distance). Hence Navier–Stokes equations might be more relevant in this case. However, this simulation shows that the cut cell method works well with complex shaped objects for moderate velocities flows ($Ma \approx 0.15$), while this would be much more difficult with the moving mesh approach, for instance.

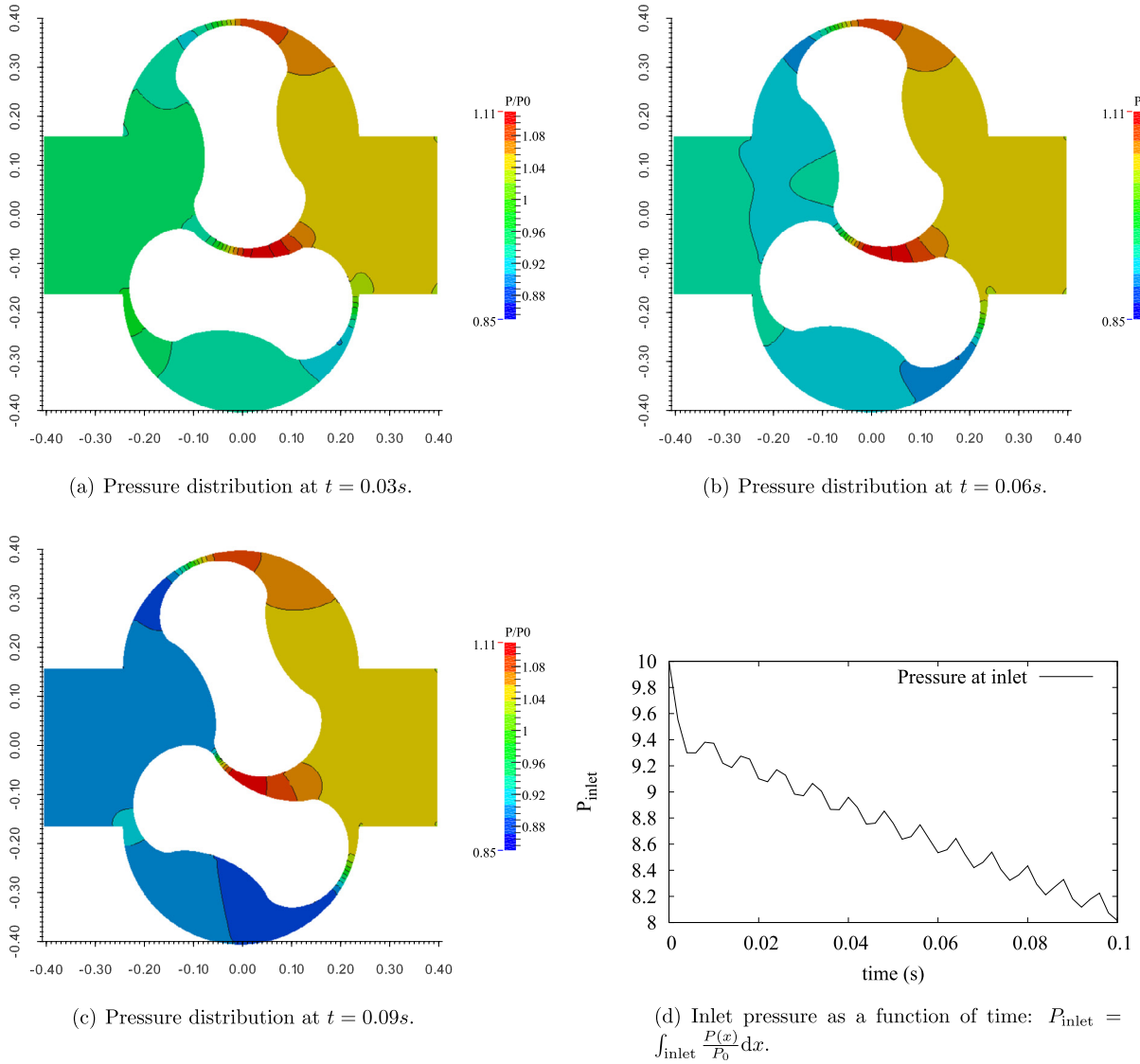


Fig. 9. Pressure in the pump at several times.

5. Three dimensional flow simulations

In this section, the cut cell method presented in section 3 is extended to 3D simulations. The approach is only detailed here for the simulation of the Crookes radiometer, which means that only pure rotation is considered. The angle between the position of a vane at time t and its initial position is denoted by $\theta(t)$, and $\dot{\theta}(t)$ stand for its rotational velocity. The gas governing equations are that of section 2.1 and the velocity discretization is similar to the one explain in section 3.1: there are N^3 velocity points and the p th velocity is denoted by $\vec{v}_p = (v_{p1}, v_{p2}, v_{p3})$. The different steps of the cut cell method – that consists in updating the values θ^n , $\dot{\theta}^n$ and $\bar{F}_{i,j,k,p}^n$ for the next time step – are that of section 3.3.3. They are written below to highlight the differences with respect to the 2D case.

We assume that at time t^n we have the average value of the distribution function $\bar{F}_{i,j,k,p}^n$ in every virtual cell $\bar{\Omega}_{i,j,k}^n$. Like in 2D, a virtual cell is defined as the intersection between the gaseous domain $\Omega_g(t^n)$ and the cuboid cell $\Omega_{i,j,k}$ of the Cartesian mesh, but now it is a polyhedral with seven virtual faces. The first six faces fit with the Cartesian mesh interfaces and are denoted by $S_{i\pm\frac{1}{2},j,k}^n$, $S_{i,j\pm\frac{1}{2},k}^n$, $S_{i,j,k\pm\frac{1}{2}}^n$. The seventh one is a plane approximation of the solid boundary denoted by $S_{i,j,k}$ and \vec{n}_w is its normal vector directed outward. We also assume that at time t^n , the angle θ^n and rotational velocity $\dot{\theta}^n$ of the solid are known. The different steps are the following:

1. The position of the solid body at t^{n+1} is computed with (21).
2. The virtual cells are arranged into control volumes $C_{i,j,k}^n(t_n)$ following the rule given in section 3.2.3 which is naturally extended to 3D. The average value $F_{i,j,k,p}^n$ of the distribution over the control volume $C_{i,j,k}^n(t^n)$ is computed: 2D relation (14) is replaced by

$$F_{i,j,k,p}^n := \frac{1}{V_{i,j,k}^n} \sum_{(i',j',k') \in \sigma_{i,j,k}^n} \bar{V}_{i',j',k'}^n \bar{F}_{i',j',k',p}^n.$$

3. The virtual cells and control volumes are moved according to the new position computed at step 1. The volumes $\bar{V}_{i,j,k}^n$, $V_{i,j,k}^n$, $V_{i,j,k}^{n+1,*}$ of the virtual cells $\bar{\Omega}_{i,j,k}^n$, control volumes $C_{i,j,k}^n(t^n)$ and $C_{i,j,k}^n(t^{n+1})$ at times t^n and t^{n+1} , respectively, are computed.
4. The boundary condition is computed for each cut cell with the discrete form of (5), which yields:

$$F_w(t^n, \bar{r}_{i,j,k}^n \in \Gamma, \bar{v}_p \in \mathcal{V}_{in}) = \phi \mathcal{M}[1, \bar{u}_w, T_w],$$

where ϕ is given by

$$\phi = - \frac{\sum_{\bar{v}_p \in \mathcal{V}_{out}} (\bar{v}_p - \bar{u}_w(t^n, \bar{r}_{i,j,k}^n)) \cdot \bar{n}_w(t^n, \bar{r}_{i,j,k}^n) F_{i,j,k,p}^n \Delta v_x \Delta v_y \Delta v_z}{\sum_{\bar{v}_p \in \mathcal{V}_{in}} (\bar{v}_p - \bar{u}_w(t^n, \bar{r}_{i,j,k}^n)) \cdot \bar{n}_w(t^n, \bar{r}_{i,j,k}^n) \mathcal{M}[1, \bar{u}_w(t^n, \bar{r}_{i,j,k}^n), T_w] \Delta v_x \Delta v_y \Delta v_z}$$

Here, $\bar{r}_{i,j,k}^n$ is the center of $S_{i,j,k}$ and $\mathcal{V}_{in} = \{\bar{v}_p | (\bar{v}_p - \bar{u}_w(t^n, \bar{r}_{i,j,k}^n)) \cdot \bar{n}_w(t^n, \bar{r}_{i,j,k}^n) < 0\}$ and $\mathcal{V}_{out} = \{\bar{v}_p | (\bar{v}_p - \bar{u}_w(t^n, \bar{r}_{i,j,k}^n)) \cdot \bar{n}_w(t^n, \bar{r}_{i,j,k}^n) > 0\}$ are the sets of incoming and outgoing velocities, respectively. We introduce the cylindrical coordinates of $\bar{r}_{i,j,k}^n = (r^n \cos \alpha^n, r^n \sin \alpha^n, z^n)$ in order to write the boundary velocity as $\bar{u}_w(t^n, \bar{r}_{i,j,k}^n) = r^n \dot{\theta}^n \times (\sin \alpha^n, \cos \alpha^n, 0)$.

5. The stress tensor is computed by 3D extension of (25):

$$\begin{aligned} \bar{\Sigma}_w(t^n, \bar{r}_{i,j,k}^n) &= \sum_{\bar{v}_p \in \mathcal{V}_{in}} (\bar{v}_p - \bar{u}_w(t^n, \bar{r}_{i,j,k}^n)) \otimes (\bar{v}_p - \bar{u}_w(t^n, \bar{r}_{i,j,k}^n)) F_w(t^n, \bar{r}_{i,j,k}^n, \bar{v}_p) \Delta v_x \Delta v_y \Delta v_z \\ &+ \sum_{\bar{v}_p \in \mathcal{V}_{out}} (\bar{v}_p - \bar{u}_w(t^n, \bar{r}_{i,j,k}^n)) \otimes (\bar{v}_p - \bar{u}_w(t^n, \bar{r}_{i,j,k}^n)) F_{i,j,k,p}^n \Delta v_x \Delta v_y \Delta v_z. \end{aligned}$$

The rotational velocity is then computed with

$$\dot{\theta}^{n+1} = \dot{\theta}^n + \frac{\Delta t}{J} \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \sum_{k=1}^{N_z} \left[\bar{r}_{i,j,k}^n \times (\bar{\Sigma}_w \cdot \bar{n}_w(t^n, \bar{r}_{i,j,k}^n)) |S_{i,j,k}^n| \right] \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},$$

where the sum is an approximation of the torque acting on the vanes. Then the new wall velocity at time t^{n+1} is computed.

6. Scheme (17) is easily extended to 3D to compute the average value $f_{i,j,p}^{n+1,*}$ of f at time t^{n+1} in the control volume $C_{i,j,k}^n(t^{n+1})$: indeed, we write the integral form of Boltzmann equation (7) with $V(t) = C_{i,j,k}^n(t)$ and simplify the transport term just like in (16) to get the first order explicit scheme

$$\begin{aligned} F_{i,j,k,p}^{n+1,*} &= \frac{V_{i,j,k}^n}{V_{i,j,k}^{n+1,*}} F_{i,j,k,p}^n - \frac{\Delta t}{V_{i,j,k}^{n+1,*}} \sum_{(i',j',k') \in \sigma_{i,j,k}^n} \left[\left(\mathcal{F}_{i'+\frac{1}{2},j',k',p}^n - \mathcal{F}_{i'-\frac{1}{2},j',k',p}^n \right) + \left(\mathcal{F}_{i',j'+\frac{1}{2},k',p}^n - \mathcal{F}_{i',j'-\frac{1}{2},k',p}^n \right) \right. \\ &\quad \left. + \left(\mathcal{F}_{i',j',k'+\frac{1}{2},p}^n - \mathcal{F}_{i',j',k'-\frac{1}{2},p}^n \right) + \mathcal{F}_{i',j',k',p}^n \right] + \frac{V_{i,j,k}^n}{V_{i,j,k}^{n+1,*}} \frac{1}{\tau_{i,j,k,p}^n} (\mathcal{E}_{i,j,k,p}^n - F_{i,j,k,p}^n), \end{aligned} \quad (31)$$

where the upwind numerical flux $\mathcal{F}_{i+\frac{1}{2},j,k,p}^n$, $\mathcal{F}_{i,j+\frac{1}{2},k,p}^n$, $\mathcal{F}_{i,j,k+\frac{1}{2},p}^n$ and $\mathcal{F}_{i,j,k,p}^n$ are:

$$\begin{aligned} \mathcal{F}_{i+\frac{1}{2},j,k,p}^n &:= |S_{i+\frac{1}{2},j,k}^n| [\min(v_{p1}, 0) F_{i+1,j,k,p}^n + \max(v_{p1}, 0) F_{i,j,k,p}^n] \\ \mathcal{F}_{i,j+\frac{1}{2},k,p}^n &:= |S_{i,j+\frac{1}{2},k}^n| [\min(v_{p1}, 0) F_{i,j+1,k,p}^n + \max(v_{p1}, 0) F_{i,j,k,p}^n] \\ \mathcal{F}_{i,j,k+\frac{1}{2},p}^n &:= |S_{i,j,k+\frac{1}{2}}^n| [\min(v_{p1}, 0) F_{i,j,k+1,p}^n + \max(v_{p1}, 0) F_{i,j,k,p}^n] \\ \mathcal{F}_{i,j,k,p}^n &:= |S_{i,j,k}^n| [\min([\bar{v}_p - \bar{u}_w(t^n, \bar{r}_{i,j,k}^n)] \cdot \bar{n}_w(t^n, \bar{r}_{i,j,k}^n), 0) F_w(t^n, \bar{r}_{i,j,k}^n, \bar{v}_p) \\ &\quad + \max([\bar{v}_p - \bar{u}_w(t^n, \bar{r}_{i,j,k}^n)] \cdot \bar{n}_w(t^n, \bar{r}_{i,j,k}^n), 0) F_{i,j,k,p}^n] \end{aligned}$$

7. This average value of the distribution function is distributed to the virtual cells $\overline{\Omega}_{i,j,k}^{n+1}$ merged into $C_{i,j,k}^n(t^{n+1})$ by $\overline{F}_{i,j,k,p}^{n+1} := F_{i,j,k,p}^{n+1,*}$.

5.1. Octree procedure

Full 3D simulations are computationally very expensive. For instance, based on the 2D computations presented in section 4.2, a 3D simulation of the Crookes radiometer requires 200^3 degrees of freedom for the space discretization and 30^3 for the velocity discretization. In addition to these 2160×10^8 grid points, 10 000 time steps are expected to reach the stationary rotational velocity of the vanes. In conclusion, even a massively parallel computing is not sufficient to do to this simulation in a reasonable computational time (i.e. less than several weeks).

In order to reduce the computational time, an octree procedure is implemented. First, a coarse Cartesian mesh is initialized and refined around boundaries. This means that all the cells that are close enough to the boundary are divided in 8 smaller cells (or 4 in 2D). These new cells can be divided again if necessary; and we define the depth of a cell as the number of divisions that lead to this cell. The splitting criterion is the following:

$$\text{if } \phi < 2\sqrt{2}\frac{\Delta x}{2^d} \text{ and } d \leq d_{\max} \text{ the cell is divided,} \quad (32)$$

where ϕ is the distance from the cell center to the nearest boundary and d_{\max} is a prescribed maximum depth. For the 3D simulation of the Crookes radiometer, the coarse mesh is made of 50^3 cells and $d_{\max} = 2$. With these parameters, the spatial mesh only contains 240 000 cells. Since the boundary moves, the mesh is adapted to the new location of the boundary at the beginning of each time iteration. Note that it may happen that 8 cells have to merge during this process, if all of them no longer respect criterion (32).

When the mesh changes – i.e. when cells split or merge – the distribution function has to be interpolated on the new mesh. This is done by assuming that the distribution function is constant over a cell and by using a standard restriction/prolongation method (by average and 0th order interpolation). Since every cells are cuboid, scheme (31) can be applied by using an appropriate data structure to access the neighboring cells of each numerical interfaces: here, we use the standard Z-ordering which is very efficient for that.

5.2. Parallel implementation

We describe here two natural strategies for a parallel implementation of a kinetic solver with the *Message Passing Interface* (MPI) library, and we propose our own hybrid technique.

The first method is velocity parallelization, or decomposition domain method in the velocity space: each processor computes the distribution function in the whole space domain, but for only a part of the discrete velocities. This approach is for instance used in [2] and more recently in [52]. For a given discrete velocity, the scheme is independent of the other velocities: then it is used independently by each processor, and each of them compute partial moments (by using its own reduced set of discrete velocities). To compute the full moments, all processors gather the sum of the partial moments.

The second method is a more standard space domain decomposition, see [23,27]. Each processor computes the distribution function for the whole discrete velocity grid, but only for a subdomain in the position space. To compute the numerical fluxes across the interfaces between different subdomains, the method requires communications: it is sufficient that each processor sends the distribution of its interface cells to its neighbors. We refer to [50] for a comparison of these two strategies.

Our implementation combines these ideas. Each processor uses the scheme on a space subdomain, for a partial set of discrete velocities. The space domain decomposition is made on the initial mesh, before the refinement procedure: the Cartesian structure of this mesh makes the portioning very easy. For problems with moving boundaries, it is difficult to ensure a good dynamic load balancing between different processors: the number of cells of a subdomain can change a lot due to the space refinement induced by the displacement of the solid obstacle. In order to optimize the workload distribution, we use a small number of subdomains. Groups of processors are given to each subdomain, and each processor will apply the scheme for a partial set of discrete velocities. The advantage of this technique is that we can use a large number of processors without a two large number of space subdomains. This makes the workload well balanced during the simulation for each subdomain, at least for the Crookes radiometer presented in the following section, since there is always one vane in each subdomain. For the corresponding 3D simulation, our technique is quite efficient, since it has been made with 240 processors for a CPU time lower than 12 hours.

Note that there is an other kind of hybrid parallelization which uses both MPI and OpenMP libraries (see [4]), but this is not what is used here.

5.3. Numerical example: the Crookes radiometer

First, the implementation of the method has been checked with a 3D simulation of the plane 2D radiometer similar to the one presented in section 4.2: this 2D geometry is extruded to get a cylinder shaped radiometer (see Fig. 10, left), and

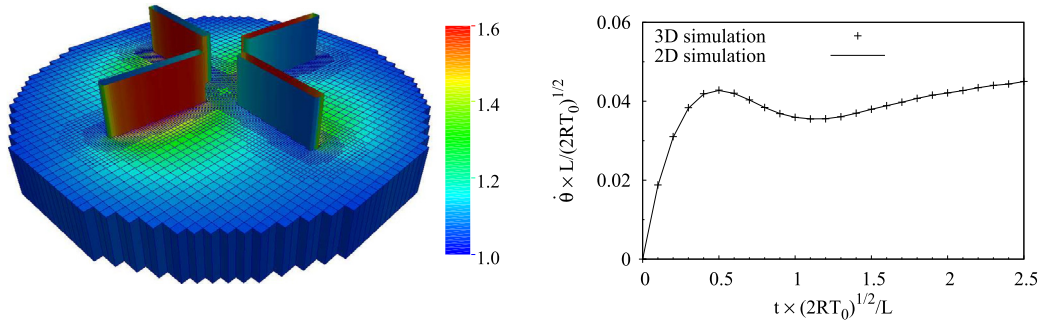


Fig. 10. 3D extruded radiometer: temperature field T/T_0 and mesh (left) and radial velocity profile as a function of time for 2D and 3D simulations (right).

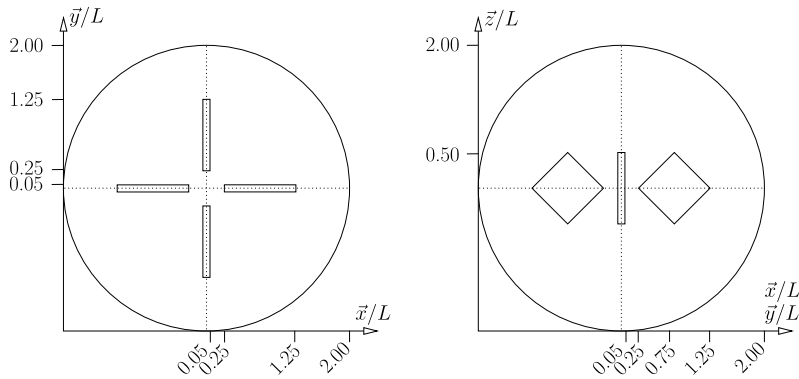


Fig. 11. 3D radiometer: cross section in the plane xOy (left) and in the plane xOz (right). See Fig. 12 for a 3D view.

periodic boundary conditions are imposed at the upper and lower boundaries to simulate the infinite vanes. In this case, the moment of inertia is $97\rho_0 L^5/370$, where L is the height of the extruded vanes. For a Knudsen number of 0.5, 2D and 3D simulations give exactly the same results, as it is shown in Fig. 10, right. For this comparison, the AMR technique is used for both simulations, and a plane section of the 3D mesh is the same as the 2D mesh.

Now the real 3D radiometer is made of four square shaped vanes. The dimensions of the vanes are L for the diagonal and $L/10$ for the width. They are immersed in a sphere of radius $2L$, and their centers are in the plane $z=0$, at a distance $0.75L$ from the center of the sphere. The corresponding geometry is shown in Fig. 11. The moment of inertia of the vanes is computed with a material density ρ_0 equal to the mass density of the surrounding gas which gives $J = \frac{97}{750} \rho_0 L^5$. This density is not realistic, but it makes the vanes faster, and it is easier to observe their movement.

At $t=0$, the radial velocity of the vanes is zero and the temperature T_0 in the domain is uniform. The gas is governed by the BGK model, where the relaxation time is $\tau = \frac{\sqrt{\pi}}{2} \frac{\rho_0}{\rho} \frac{\text{Kn}L}{\sqrt{2RT_0}}$. All the boundary conditions are diffuse reflections with constant temperatures: the sphere boundary is maintained at temperature T_0 and the white side of the vanes as well, while their black side is maintained at temperature $2T_0$. On the edge of the vanes, the temperature is discontinuous (T_0 on one part and $2T_0$ on the other part).

We also use the 2D simulation to estimate the resolution required by the 3D computation. The difference between the results obtained with a 2D mesh of 50^2 cells refined by the AMR technique with a maximum depth $d_{\max}=2$ and the fine Cartesian structured mesh of 500^2 cells is less than 5%, which is considered as sufficiently accurate here. Consequently, a 3D AMR mesh of 50^3 cells with a depth of 2 should be accurate enough for a 3D simulation. This is computationally possible, since this mesh contains 240 000 cells, which is much smaller than the equivalent Cartesian mesh of $200^3 = 8\,000\,000$ cells. The corresponding simulation is shown in Fig. 12 at different times.

We have made other simulations with three Knudsen numbers 0.1, 0.3, and 0.5 until the steady state is reached. In Fig. 13(a) is shown the evolution of the radial velocity: we clearly see the convergence to a constant velocity, which is larger for $\text{Kn}=0.5$. A comparison between 2D and 3D geometries is shown in Fig. 13(b): the 3D vanes are clearly faster than the 2D vanes.

Note that this test is just shown to illustrate the potential of our method. We are not aware of any similar simulation in the literature so far, and we are not able to present any comparison. Moreover, we do not claim this is a realistic simulation, since the moment of inertia of the vanes is too small, and their width is too large. However, we do not know any experimental measures of the motion of the radiometer. If any, it would probably be necessary to make a more intensive simulation, since the refinement should be stronger around the vanes that are generally very thin.

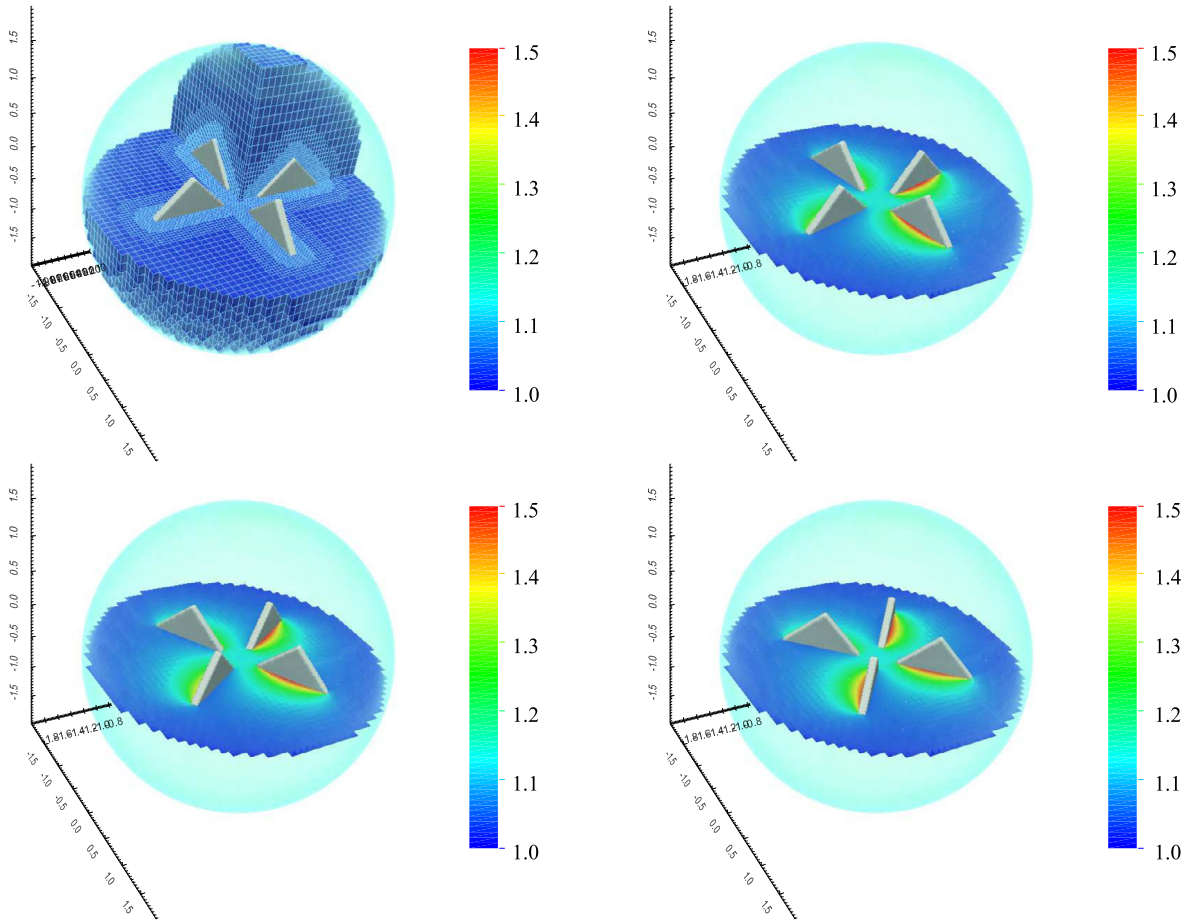


Fig. 12. 3D Crookes radiometer for $Kn = 0.5$: the temperature field T/T_0 on the plane $z = 0$ is shown at times $t \times L / \sqrt{2RT_0} = 0, 5, 10$ and 15 (from left to right and from top to bottom). The mesh is shown at $t = 0$ (top-left).

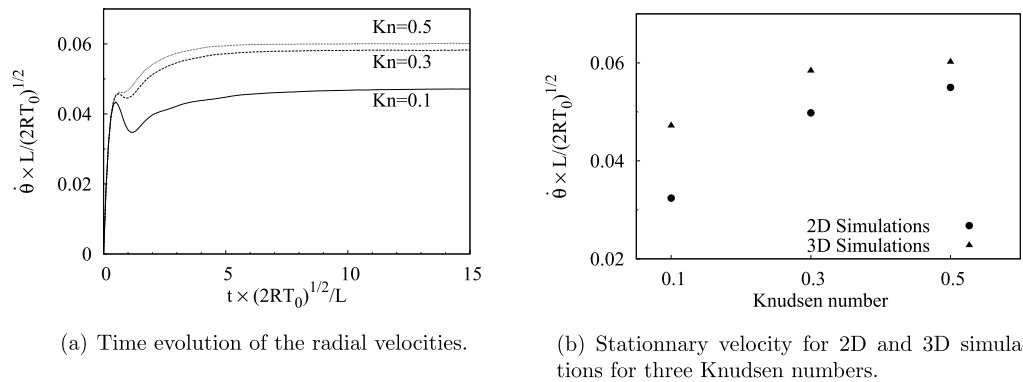


Fig. 13. Radial velocity profiles for 3D simulations with 3 different Knudsen numbers.

6. Conclusion

A numerical method for solving kinetic equations with moving obstacles has been presented. This method is an extension to the kinetic theory of the cut cell technique used in computational fluid dynamics. The main advantage of this algorithm is that it combines the simplicity of the Cartesian grid based methods to the accuracy of the body fitted methods, which ensures exact mass conservation. The method is easily extended to 3D flows, and its accuracy has been proved with the simulation of a Crookes radiometer. Another advantage of our approach is a simple and generic treatment of all kinds of cut cells. This is essential, especially for 3D problems in which there are many different kinds of cut cells.

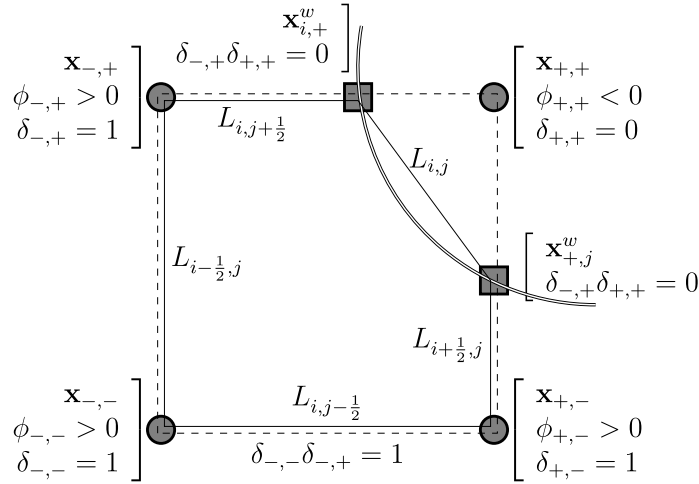


Fig. 14. Summary of the notations used in Appendix A: the cell $\Omega_{i,j}$ is shown with a dotted line, its corresponding virtual cell $\bar{\Omega}_{i,j}$ with a solid line, and the solid boundary with a double line.

Our goal is now to improve the accuracy of our method by using a second order scheme. Since the mesh is Cartesian, the main difficulty is to approximate the gradient of the distribution function on the cut cells with enough accuracy. Such an extension has already been done (see [15]), but while it works well for non-moving obstacles, it is not efficient enough for general problems. An other perspective is the validation of the 3D algorithm, in particular by using experimental data.

Acknowledgements

Experiments presented in this paper were carried out using the PlaFRIM experimental testbed, being developed under the Inria PlaFRIM development action with support from LABRI and IMB and other entities: Conseil Régional d'Aquitaine, FeDER, Université de Bordeaux and CNRS (see <https://plafrim.bordeaux.inria.fr/>). Computer time for this study was also provided by the computing facilities MCIA (Mésocentre de Calcul Intensif Aquitain) of the Université de Bordeaux and of the Université de Pau et des Pays de l'Adour. This study has been carried out in the frame of “the Investments for the future” Programme IdEx Bordeaux – CPU (ANR-10-IDEX-03-02).

Appendix A. Computation of the cell geometric parameters

We describe below how these parameters are computed for 2D problems. There is no specific difficulty to extend our algorithms to 3D problems: however, the formula are a bit long, and to shorten the paper, this extension is left to the reader.

The computation is made with the following four steps:

1. identification of the type of each cell (solid, gas, our cut cell);
2. for each virtual cell, computation of the lengths of its five edges, and computation of its normal vector of its fifth edge (the one which is a linear approximation of the solid boundary);
3. identification of the cut cells that have to merge, and computation of the set $\sigma_{i,j}$;
4. computation of the area of each virtual cells and corresponding control volumes.

Before we describe these steps in the following sections, we point out that a level-set signed distance function $\phi : \mathbf{x} \rightarrow \phi(\mathbf{x})$ is systematically used. It gives the shortest distance between a point \mathbf{x} in the computational domain and the solid boundary. This function is negative if \mathbf{x} is inside the solid, and positive if it is inside the gas, and hence the zero level-set of ϕ is the solid boundary. In our algorithm, the values of ϕ are computed analytically at each node $\mathbf{x}_{i\pm\frac{1}{2},j\pm\frac{1}{2}}$ of the Cartesian grid, and we set $\phi_{i\pm\frac{1}{2},j\pm\frac{1}{2}} := \phi(\mathbf{x}_{i\pm\frac{1}{2},j\pm\frac{1}{2}})$. Since solid boundaries move, it is necessary to update these values at each time step.

From now on, we consider a single cell $\Omega_{i,j}$, and in order to simplify the notations, indices i and j of variables defined at the vertices of this cell will be omitted. For instance, the values $\mathbf{x}_{i+\frac{1}{2},j+\frac{1}{2}}$ and $\phi_{i+\frac{1}{2},j+\frac{1}{2}}$ will be denoted by $\mathbf{x}_{+,+}$ and $\phi_{+,+}$. All the notations used here are shown in Fig. 14.

A.1. Identification of cell types

It is clear that $\Omega_{i,j}$ is a gas cell if the values of ϕ at its four vertices are positive. At the contrary, it is a solid cell if these values are all negative. Finally, if these values have different signs, the cell is cut by the solid boundary. These three types are identified by looking at the sign of $m := \min(\phi_{-,-}, \phi_{+,-}, \phi_{+,+}, \phi_{-,+})$ and $M := \max(\phi_{-,-}, \phi_{+,-}, \phi_{+,+}, \phi_{-,+})$:

$$m > 0 \Leftrightarrow \Omega_{i,j} \text{ is a gas cell,}$$

$$M < 0 \Leftrightarrow \Omega_{i,j} \text{ is a solid cell,}$$

$$M > 0 \text{ and } m < 0 \Leftrightarrow \Omega_{i,j} \text{ is a cut cell.}$$

Of course, if a cell is crossed by a solid object thinner than Δx , it will be considered as a gas cell: it is important that the Cartesian grid is fine enough to resolve all the solid objects.

Finally, the case $\phi_{\pm,\pm} = 0$ is complex and is avoided by using the modification $\phi := \text{sign}(\phi) \times \max(|\phi|, 10^{-10} \Delta x)$. This means that grid points that are exactly on the solid boundary are numerically considered as moved on a distance of $10^{-10} \Delta x$. This has no influence on the accuracy of the results.

A.2. Lengths of edges and normal vector of the virtual cell

To each of the four edges of $\Omega_{i,j}$ are associated the four points $\mathbf{x}_{\pm,j}^w$ et $\mathbf{x}_{i,\pm}^w$ defined as follows. If it is a cut cell, two of these points are intersection points of an edge with the solid boundary, and the two others are not used by the algorithm. When $\Omega_{i,j}$ is a gas or solid cell, none of these four points is used. These points are defined by linear approximations

$$\mathbf{x}_{\pm,j}^w = \begin{bmatrix} x_{i\pm\frac{1}{2}} \\ y_{j-\frac{1}{2}} - \Delta y \frac{\phi_{\pm,-}}{\phi_{\pm,+} - \phi_{\pm,-}} \end{bmatrix} \quad \text{and} \quad \mathbf{x}_{i,\pm}^w = \begin{bmatrix} x_{i-\frac{1}{2}} - \Delta x \frac{\phi_{-,\pm}}{\phi_{+,\pm} - \phi_{-,\pm}} \\ y_{j\pm\frac{1}{2}} \end{bmatrix}.$$

To compute the lengths of the edges of $\Omega_{i,j}$, we use the symbol $\delta_{\pm,\pm}$ that is 1 if $\mathbf{x}_{\pm,\pm}$ is inside the gas, and 0 if it is inside the solid. This value is given by $\delta_{\pm,\pm} = \max(\phi_{\pm,\pm}, 0)/|\phi_{\pm,\pm}|$. Also note that looking at the products $\delta_{\pm,-}\delta_{\pm,+}$ (or $\delta_{-,\pm}\delta_{+,\pm}$) tells us if left and right edges (respectively, upper and lower edges) are crossed by the solid boundary. This notation allows us to easily write the four Cartesian edge lengths of the virtual cell $\bar{\Omega}_{i,j}$:

$$|L_{i\pm\frac{1}{2},j}^n| = \delta_{\pm,-}\|\mathbf{x}_{\pm,j}^w - \mathbf{x}_{\pm,-}\| + \delta_{\pm,+}\|\mathbf{x}_{\pm,+} - \mathbf{x}_{\pm,j}^w\| + \delta_{\pm,-}\delta_{\pm,+}\|\mathbf{x}_{\pm,+} - \mathbf{x}_{\pm,-}\|,$$

$$|L_{i,j\pm\frac{1}{2}}^n| = \delta_{-,\pm}\|\mathbf{x}_{i,\pm}^w - \mathbf{x}_{-,\pm}\| + \delta_{+,\pm}\|\mathbf{x}_{+,\pm} - \mathbf{x}_{i,\pm}^w\| + \delta_{-,\pm}\delta_{+,\pm}\|\mathbf{x}_{+,\pm} - \mathbf{x}_{-,\pm}\|.$$

The length $|L_{i,j}^n|$ of the fifth edge (that fit with the solid boundary) is the norm of the vector defined by the two solid boundary/edge intersection points.

Finally, the normal vector to this edge is computed by a Green formula, which naturally gives the correct outward direction:

$$\mathbf{n} = \frac{1}{|L_{i,j}^n|} \left(|L_{i+\frac{1}{2},j}^n| - |L_{i-\frac{1}{2},j}^n| \right) \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{1}{|L_{i,j}^n|} \left(|L_{i,j+\frac{1}{2}}^n| - |L_{i,j-\frac{1}{2}}^n| \right) \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

A.3. Cell merging: computation of $\sigma_{i,j}$

We consider one virtual cell $\bar{\Omega}_{i,j}$. The set $\sigma_{i,j}$ collects the indices of all the virtual cells that merge with $\bar{\Omega}_{i,j}$. In this set, let us denote by (i', j') the indices of the unique virtual cell whose center $\mathbf{x}_{i',j'}$ is inside the gas, that is to say

$$\frac{1}{4}(\phi_{-,-} + \phi_{+,-} + \phi_{+,+} + \phi_{-,+}) > 0. \quad (33)$$

We point out that the rule given in section 3.2.3 ensures that this “master” virtual cell is unique. Its indices (i', j') are determined by the following algorithm

$(i', j') := (i, j) \quad (\alpha, \beta) := (1, 1)$

while (33) is false, do

$M := \max(|L_{i+\frac{1}{2},j}^n|, \alpha \times |L_{i-\frac{1}{2},j}^n|, |L_{i,j+\frac{1}{2}}^n|, \beta \times |L_{i,j-\frac{1}{2}}^n|)$

if $|L_{i+\frac{1}{2},j}^n| = M$ then $(i', j') := (i' + 1, j')$ $(\alpha, \beta) := (0, 1)$

if $|L_{i-\frac{1}{2},j}^n| = M$ then $(i', j') := (i' - 1, j')$ $(\alpha, \beta) := (1, 1)$

if $|L_{i,j+\frac{1}{2}}^n| = M$ then $(i', j') := (i', j' + 1)$ $(\alpha, \beta) := (1, 0)$

if $|L_{i,j-\frac{1}{2}}^n| = M$ then $(i', j') := (i', j' - 1)$ $(\alpha, \beta) := (1, 1)$

end while

(34)

Now, we comment on this algorithm. First we point out that a given cell can belong to only one control volume. Indeed, our criterion to merge a small cell with its neighbour is that it merges with the one that shares the largest edge given by M . In case of two equal largest edges, only one cell is selected, arbitrarily.

Second, a control volume can contain several small cells. In fact, if a cell merges with another one whose center is still in the solid, both cells have to merge with another one. This procedure is continued until a cell whose center is in the gas is found (this is condition (33) in the do while loop). This might lead to a control volume which is bigger than necessary: we could use a stopping criterion (for instance as soon as the size of the control volume is larger than half the size of a gas cell), but this is not done in this paper.

Third, the coefficients α and β are introduced to prevent the algorithm to go backward (in 3D we would need three coefficients). Indeed, consider a cell A that has to merge with cell B. If B is too small, it has to merge too. It might happen that the largest edge of B is common with A: the coefficients α and β avoid that B merges with A (which would create an infinite loop). They are used to set the selected length edge to zero, so that it cannot be chosen again. Note that this problem only appears in cases for which the mesh is too coarse around a concave boundary.

Note that this algorithm is close to those proposed in [37,36]. It seems that the main difference is the choice of the neighboring cell that has to merge with the small cell. In [37] this cell is the one that is pointed to by the normal vector that starts from the middle of the solid edge. In [36], this cell is the biggest neighboring cell.

Finally, note that the set $\sigma_{i,j}$ is not really computed: practically, we compute the numerical fluxes across the edges of each virtual cells, and these fluxes are directly added to the fluxes of the master cell of indices (i', j') .

A.4. Virtual cell and control volume areas

The area of a virtual cell can be computed with the length of its edges and the coordinates of its vertices by a Green formula:

$$\begin{aligned} \bar{s}_{i,j} &= \int_{\bar{\Omega}_{i,j}} dS = \frac{1}{2} \int_{\bar{\Omega}_{i,j}} \nabla \cdot \mathbf{x} dS = \frac{1}{4} |L_{i+\frac{1}{2},j}^n| \left(\delta_{+,+} \mathbf{x}_{+,+} + \delta_{+,-} \mathbf{x}_{+,-} + \delta_{+,+}^+ \mathbf{x}_{+,j}^w \right) \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &\quad + \frac{1}{4} |L_{i-\frac{1}{2},j}^n| \left(\delta_{-,+} \mathbf{x}_{-,+} + \delta_{-,-} \mathbf{x}_{-,-} + \delta_{-,-}^+ \mathbf{x}_{-,j}^w \right) \cdot \begin{bmatrix} -1 \\ 0 \end{bmatrix} \\ &\quad + \frac{1}{4} |L_{i,j+\frac{1}{2}}^n| \left(\delta_{+,+} \mathbf{x}_{+,+} + \delta_{-,+} \mathbf{x}_{-,+} + \delta_{+,+}^+ \mathbf{x}_{i,+}^w \right) \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ &\quad + \frac{1}{4} |L_{i,j-\frac{1}{2}}^n| \left(\delta_{-,-} \mathbf{x}_{-,-} + \delta_{+,-} \mathbf{x}_{+,-} + \delta_{-,-}^+ \mathbf{x}_{i,-}^w \right) \cdot \begin{bmatrix} 0 \\ -1 \end{bmatrix} \\ &\quad + \frac{1}{4} |L_{i,j}^n| \left(\delta_{+,+}^+ \mathbf{x}_{+,j}^w + \delta_{-,-}^+ \mathbf{x}_{-,j}^w + \delta_{+,+}^+ \mathbf{x}_{i,+}^w + \delta_{-,-}^+ \mathbf{x}_{i,-}^w \right) \cdot \mathbf{n} \end{aligned}$$

Here, we used $\delta_{\pm,+}^{\pm,-} := 1 - \delta_{\pm,+} \delta_{\pm,-}$ which is 1 if left or right edges are crossed by the solid boundary, and 0 else. This formula is nothing but the sum on each edge of the dot product between its normal vector and the vector pointing to the center of the edge, multiplied by the length edge.

Finally, the area of the control volume is computed and stored in the master cell of indices (i', j') defined by algorithm (34), with the following loop along all the cut cells:

$s_{i,j} = 0$ for every cells.

For all (i, j) do :

compute (i', j') with algorithm (34).

$s_{i',j'} := s_{i',j'} + \bar{s}_{i,j}$

end do

References

- [1] Ann S. Almgren, John B. Bell, Phillip Colella, Tyler Marthaler, A Cartesian grid projection method for the incompressible Euler equations in complex geometries, *SIAM J. Sci. Comput.* 18 (5) (1997) 1289–1309.
- [2] V.V. Aristov, S.A. Zabelok, A deterministic method for solving the Boltzmann equation with parallel computations, *Comput. Math. Math. Phys.* 42 (3) (2002) 406–418.
- [3] Robert R. Arslanbekov, Vladimir I. Kolobov, Anna A. Frolova, Immersed boundary method for Boltzmann and Navier–Stokes solvers with adaptive Cartesian mesh, *AIP Conf. Proc.* 1333 (1) (2011) 873–877.
- [4] C. Baranger, J. Claudel, N. Hérouard, L. Mieussens, Locally refined discrete velocity grids for stationary rarefied flow simulations, *J. Comput. Phys.* A 257 (2014) 572–593.
- [5] M. Berger, R.J. LeVeque, An adaptive Cartesian mesh algorithm for the Euler equations in arbitrary geometries, *AIAA paper AIAA-89-1930*, 1989.
- [6] Florian Bernard, Efficient asymptotic preserving scheme for BGK and ES-BGK models on Cartesian grid, PhD thesis, Polytechnico di Torino and Université de Bordeaux, 2015.
- [7] P.L. Bhatnagar, E.P. Gross, M. Krook, A model for collision processes in gases: small amplitude processes in charged and neutral one-component systems, *Phys. Rev.* 94 (1954) 511–525.
- [8] G.A. Bird, *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*, Oxford Engineering Science Series, vol. 42, Oxford Science Publications, 1994.
- [9] Songze Chen, Kun Xu, A Cartesian grid-based unified gas kinetic scheme, *AIP Conf. Proc.* 1628 (1) (2014) 995–1002.
- [10] Songze Chen, Kun Xu, Cunbia Lee, The dynamic mechanism of a moving Crookes radiometer, *Phys. Fluids* 24 (2012).
- [11] Songze Chen, Kun Xu, Cunbia Lee, Qingdong Cai, A unified gas-kinetic scheme with moving mesh and velocity space adaptation, *J. Comput. Phys.* 231 (2012) 6643–6664.
- [12] C.K. Chu, Kinetic-theoretic description of the formation of a shock wave, *Phys. Fluids* 8 (1965) 12–22.
- [13] D.K. Clarke, H.A. Hassan, M.D. Salas, Euler calculations for multielement airfoils using Cartesian grids, *AIAA J.* 24 (3) (1986) 353–358, 2016/02/18.
- [14] W. Crookes, On attraction and repulsion resulting from radiation, *Philos. Trans. R. Soc. Lond.* 164 (1874) 501–527.
- [15] G. Dechristé, Méthodes numériques pour la simulation d'écoulements de gaz raréfiés autour d'obstacles mobiles, PhD thesis, Université de Bordeaux, 2014, in French.
- [16] Guillaume Dechristé, Luc Mieussens, A cut cell method for the 3d simulation of Crookes radiometer, in: *Proceedings of the 29th International Symposium on Rarefied Gas Dynamics*, in: *AIP Conf. Proc.*, vol. 1628, 2014, p. 988.
- [17] G. Dimarco, L. Pareschi, Numerical methods for kinetic equations, *Acta Numer.* 23 (2014) 369–520.
- [18] Francis Filbet, Chang Yang, Inverse Lax–Wendroff method for boundary conditions of Boltzmann type models, *J. Comput. Phys.* 245 (2013).
- [19] M. Gad-el Hak, *The MEMS Handbook*, Mechanical and Aerospace Engineering Series, Taylor & Francis, 2001.
- [20] C.W. Hirt, An arbitrary Lagrangian–Eulerian computing technique, in: Maurice Holt (Ed.), *Proceedings of the Second International Conference on Numerical Methods in Fluid Dynamics*, in: *Lecture Notes in Physics*, vol. 8, Springer, Berlin, Heidelberg, 1971, pp. 350–355.
- [21] C.W. Hirt, A.A. Amsden, J.L. Cook, An arbitrary Lagrangian–Eulerian computing method for all flow speeds, *J. Comput. Phys.* 14 (3) (1974) 227–253.
- [22] Lowell H. Holway, New statistical models for kinetic theory: methods of construction, *Phys. Fluids* (1958–1988) 9 (9) (1966) 1658–1673.
- [23] Murat Ilgaz, Ismail H. Tuncer, Parallel implementation of a gas-kinetic BGK method on unstructured grids for 3-d inviscid missile flows, in: *Parallel Computational Fluid Dynamics 2007*, in: *Lecture Notes in Computational Science and Engineering*, vol. 67, Springer, Berlin, Heidelberg, 2009, pp. 449–456.
- [24] D.M. Ingram, D.M. Causon, C.G. Mingham, Developments in Cartesian cut cell methods, *Math. Comput. Simul.* 61 (2003) 561–572.
- [25] G. Karniadakis, A. Beskok, N. Aluru, *Microflows and Nanoflows*, Interdisciplinary Applied Mathematics, vol. 29, Springer, 2000.
- [26] Andrew Ketsdever, Natalia Gimelshein, Sergey Gimelshein, Nathaniel Selden, Radiometric phenomena: from the 19th to the 21st century, in: *Vacuum Gas Dynamics: Theory, Experiments and Practical Applications*, *Vacuum* 86 (11) (2012) 1644–1662.
- [27] V.I. Kolobov, R.R. Arslanbekov, V.V. Aristov, A.A. Frolova, S.A. Zabelok, Unified solver for rarefied and continuum flows with adaptive mesh and algorithm refinement, *J. Comput. Phys.* 223 (2) (2007) 589–608.
- [28] G.J. LeBeau, A parallel implementation of the direct simulation Monte Carlo method, *Comput. Methods Appl. Mech. Eng.* 174 (3–4) (1999) 319–337.
- [29] L. Mieussens, Discrete-velocity models and numerical schemes for the Boltzmann–BGK equation in plane and axisymmetric geometries, *J. Comput. Phys.* 162 (2000) 429–466.
- [30] Mieussens Luc, A survey of deterministic solvers for rarefied flows, in: *Proceedings of the 29th International symposium on Rarefied Gas Dynamics*, in: *AIP Conf. Proc.*, vol. 1628, 2014, p. 943.
- [31] Rajat Mittal, Gianluca Iaccarino, Immersed boundary methods, *Annu. Rev. Fluid Mech.* 37 (2005) 239–261.
- [32] Rajat Mittal, Veeraraghavan Seshadri, Holavanahalli S. Udaykumar, Flutter, tumble and vortex induced autorotation, *Theor. Comput. Fluid Dyn.* 17 (3) (2004) 165–170.
- [33] Taku Ohwada, Masashi Kunihiya, Direct simulation of a flow produced by a plane wall oscillating in its normal direction, *AIP Conf. Proc.* 663 (1) (2003) 202–209.
- [34] Cem Pekardan, Sruti Chigullapalli, Lin Sun, Alina Alexeenko, Immersed boundary method for Boltzmann model kinetic equations, *AIP Conf. Proc.* 1501 (1) (2012) 358–365.
- [35] Sandra Pieraccini, Gabriella Puppo, Implicit–explicit schemes for BGK kinetic equations, *J. Sci. Comput.* 32 (2007) 1–28.
- [36] Stephane Popinet, Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries, *J. Comput. Phys.* 190 (2) (2003) 572–600.
- [37] James J. Quirk, An alternative to unstructured grids for computing gas dynamic flows around arbitrarily complex two-dimensional bodies, *Comput. Fluids* 23 (1) (1994) 125–142.
- [38] D.J. Rader, M.A. Gallis, J.R. Torczynski, DSMC moving-boundary algorithms for simulating MEMS geometries with opening and closing gaps, *AIP Conf. Proc.* 1333 (1) (2011) 760–765.
- [39] Osborne Reynolds, On certain dimensional properties of matter in the gaseous state, part I & part II, *Philos. Trans. R. Soc. Lond.* 170 (1879) 727–845.
- [40] Giovanni Russo, Francis Filbet, Semilagrangian schemes applied to moving boundary problems for the BGK model of rarefied gas dynamics, *Kinet. Relat. Models* 2 (1) (2009) 231–250.
- [41] N. Selden, C. Ngalande, N. Gimelshein, S. Gimelshein, A. Ketsdever, Origins of radiometric forces on a circular vane with a temperature gradient, *J. Fluid Mech.* 634 (2009) 419.
- [42] N. Selden, C. Ngalande, S. Gimelshein, E.P. Muntz, A. Alexeenko, A. Ketsdever, Area and edge effects in radiometric forces, *Phys. Rev. E* 79 (Apr. 2009) 041201.
- [43] E.M. Shakhov, Generalization of the Krook kinetic relaxation equation, *Fluid Dyn.* 5 (1968) 142–145.
- [44] Samir Shrestha, Sudarshan Tiwari, Axel Klar, Steffen Hardt, Numerical simulation of moving rigid body in rarefied gases, arXiv:1406.5176, 2014.
- [45] Stefan Stefanov, Peter Gospodinov, Carlo Cercignani, Monte Carlo simulation and Navier–Stokes finite difference calculation of unsteady-state rarefied gas flows, *Phys. Fluids* (1994–present) 10 (1) (1998) 289–300.

- [46] S. Taguchi, K. Aoki, Rarefied gas flow around a sharp edge induced by a temperature field, *J. Fluid Mech.* 694 (2012) 191.
- [47] Satoshi Taguchi, Kazuo Aoki, A simple model for flows around moving vanes in Crookes radiometer, *AIP Conf. Proc.* 1501 (1) (2012) 786–793.
- [48] Satoshi Taguchi, Kazuo Aoki, Motion of an array of plates in a rarefied gas caused by radiometric force, *Phys. Rev. E* 91 (Jun. 2015) 063007.
- [49] Tao Tang, Moving mesh methods for computational fluid dynamics, *Contemp. Math.* 383 (2005) 141–173.
- [50] V. Titarev, M. Dumbser, S. Utyuzhnikov, Construction and comparison of parallel implicit kinetic solvers in three spatial dimensions, *J. Comput. Phys.* 256 (2014) 17–33.
- [51] V.A. Titarev, Numerical method for computing two-dimensional unsteady rarefied gas flows in arbitrarily shaped domains, *Comput. Math. Math. Phys.* 49 (2009) 1197–1211.
- [52] V.A. Titarev, Efficient deterministic modelling of three-dimensional rarefied gas flows, *Commun. Comput. Phys.* 12 (1) (2012) 162–192.
- [53] Tetsuro Tsuji, Kazuo Aoki, Moving boundary problems for a rarefied gas: spatially one-dimensional case, *J. Comput. Phys.* 250 (2013) 574–600.
- [54] P.G. Tucker, Z. Pan, A Cartesian cut cell method for incompressible viscous flow, *Appl. Math. Model.* 24 (8–9) (2000) 591–606.
- [55] H.S. Udaykumar, R. Mittal, P. Rampunggoon, A. Khanna, A sharp interface Cartesian grid method for simulating flows with complex moving boundaries, *J. Comput. Phys.* 174 (1) (2001) 345–380.
- [56] G. Yang, D.M. Causon, D.M. Ingram, Calculation of compressible flows about complex moving geometries using 3D Cartesian cut cell method, *Int. J. Numer. Methods Fluids* 33 (2000) 1121–1151.
- [57] T. Ye, R. Mittal, H.S. Udaykumar, W. Shyy, An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries, *J. Comput. Phys.* 156 (2) (1999) 209–240.
- [58] S. Zabelok, R. Arslanbekov, V. Kolobov, Adaptive kinetic–fluid solvers for heterogeneous computing architectures, *J. Comput. Phys.* 303 (December 2015) 455–469.
- [59] Darren De Zeeuw, Kenneth G. Powell, An adaptively refined Cartesian mesh solver for the Euler equations, *J. Comput. Phys.* 104 (1) (1993) 56–68.
- [60] Chonglin Zhang, Thomas E. Schwartzentruber, Robust cut-cell algorithms for DSMC implementations employing multi-level Cartesian grids, *Comput. Fluids* 69 (2012) 122–135.