Locally refined discrete velocity grids for deterministic rarefied flow simulations

C. Baranger, J. Claudel, N. Hérouard, and L. Mieussens

Citation: AIP Conf. Proc. **1501**, 389 (2012); doi: 10.1063/1.4769549
View online: http://dx.doi.org/10.1063/1.4769549
View Table of Contents: http://proceedings.aip.org/dbt/dbt.jsp?KEY=APCPCS&Volume=1501&Issue=1
Published by the American Institute of Physics.

Locally Refined Discrete Velocity Grids for Deterministic Rarefied Flow Simulations

C. Baranger*, J. Claudel*, N. Hérouard* and L. Mieussens†

*CEA-CESTA 15 av Sablières CS 60001 33116 LE BARP CEDEX. celine.baranger@cea.fr
†Univ. Bordeaux, IMB, UMR 5251, F-33400 Talence, France.CNRS, IMB, UMR 5251, F-33400 Talence, France. Luc.Mieussens@math.u-bordeaux1.fr

Abstract. In the context of atmospheric re-entry studies, the CEA-CESTA needs to perform simulations of hypersonic steady rarefied gas flows. In the range of 60-120 km altitude, the air is in a transitional regime and is described by the Bhatnagar-Gross-Krook (BGK) kinetic model. As a deterministic method is used in order to solve the BGK model, such simulations require large memory storage and CPU time, especially for 3D configurations with high Mach numbers. Local refinements of meshes is a way to decrease those requirements in memory and CPU time ; however, if space mesh refinements are rather standard, velocity grid refinements are more unusual. The work presented here proposes an algorithm that allows to build such a locally refined discrete velocity grid. This algorithm uses the properties of the BGK collision operator to predict the aspect of local distributions in each zone of the velocity space by using macroscopic temperature and velocity fields computed in the physical space. On this basis, it determines an appropriate discretization of the whole velocity space and constructs the corresponding grid with a recursive method. This grid is used for the whole computation.

Keywords: rarefied flow simulation, hypersonic flows, re-entry problem, transition regime, BGK model, refined grid
PACS: 51.10.+y, 47.45.-n, 47.40.Ki

INTRODUCTION

The description of a flow surrounding a flying object at hypersonic speed in the rarefied atmosphere (more than 60 km altitude) is still a challenge in the atmospheric Re-Entry community. When this flow is in a rarefied state, that is when the Knudsen number (which is the ratio $Kn = \frac{\lambda}{L}$ between the mean free path λ of particle and a characteristic macroscopic length L) is larger than 0.01, the flow cannot be accurately described by the compressible Navier-Stokes equations of Gas Dynamics. In this case, the kinetic theory has to be used. The evolution of the molecules of the gas is then described by a mass density distribution in phase space, which is a solution of the Boltzmann equation. In the transitional regime, this equation can be replaced by the simpler Bhatnagar-Gross-Krook (BGK) model.

In order to be able to compute parietal heat flux and aerodynamic coefficients for the CEA (French Alternative Energies and Atomic Energy Agency) applications in the range of 60-120 km, a kinetic description of the stationary flow is necessary.

The most popular numerical method to simulate rarefied flows is the Direct Simulation Monte Carlo method (DSMC) [1]. However, it is well known that this method is very expensive in transitional regimes, in particular for flows in the range of altitude we are interested in. The efficiency of DSMC can be improved by using coupling strategies (see [2]) or implicit schemes (see [3]), but these methods are still not very well suited for stationary computations. In contrast, deterministic methods (based on a numerical discretization of the stationary kinetic model) can be more efficient in transitional regimes.

In our team, we developed several years ago the code KISS (Kinetic Implicit Supersonic Scheme), to make 2D plane and axisymmetric simulations of rarefied flows based on the BGK model (see a description in [4, 5, 6]). This code has recently been extended to 3D computations, for polyatomic gases. Due to the physical model (polyatomic gases), the space discretization (block structured mesh), and the parallelization (space domain decomposition with MPI and inner parallelization with openMP), this code is rather different from the other existing 3D codes recently presented in the literature for the same kind of problems (the 3D code of Titarev [7] for example), even if space domain decomposition have already been used for unsteady simulation (see [8]).

All the codes designed for steady flows have a common feature: they are based on a "discrete ordinate" like approach, and use a global velocity grid. This grid is generally a Cartesian grid with a constant step size. The number of points of this grid is roughly proportional to the Mach number of the flow in each direction, and hence can be

28th International Symposium on Rarefied Gas Dynamics 2012
AIP Conf. Proc. 1501, 389-396 (2012); doi: 10.1063/1.4769549

prohibitively large for hypersonic flows, even with parallel computers. To compute realistic cases (3D configurations with Mach number larger than 20), the velocity space discretization has to be modified in order to decrease CPU time and memory storage requirements. It has already been noticed that a refinement of the grid around small velocities can improve the accuracy and reduce the cost of the computation (especially for large Knudsen numbers in flows close to solid boundaries, see [7]). However, up to our knowledge, there is no general strategy in the literature that helps us to reduce the number of discrete velocities of a velocity grid for any rarefied flow, even if some works on adaptive velocity grid have already been presented for a shock wave problem (see [9]).

The main contribution of this article is to propose an algorithm for an automatic construction of a locally refined velocity grid that allows a dramatic reduction of the number of discrete velocities, with the same accuracy as a standard Cartesian grid. This algorithm uses a compressible Navier-Stokes pre-simulation to obtain a rough estimation of the macroscopic fields of the flow. These fields are used to refine the grid wherever it is necessary by using some indicators of the width of the distribution functions in all the computational domain. This strategy allows us to simulate hypersonic flows that can hardly be simulated by standard methods, since we are indeed able to apply our method to our code KISS to simulate a re-entry flow at Mach 25 and for temperature and pressure conditions of an altitude of 90 km. Note that preliminary results have already been presented in [10].

The outline of this article is the following. First, we briefly present the kinetic description of a rarefied gas. Then we discuss the problems induced by the use of a global velocity grid, and our algorithm is presented. Finally, our approach is illustrated in section with numerical tests.

BOLTZMANN EQUATION AND CARTESIAN VELOCITY GRID

Kinetic Description of Rarefied Gases

In kinetic theory, a monoatomic gas is described by the distribution function $f(t,\mathbf{x},\mathbf{v})$ defined such that $f(t,\mathbf{x},\mathbf{v})d\mathbf{x}d\mathbf{v}$ is the mass of molecules that at time t are located in an elementary space volume $d\mathbf{x}$ centered in $\mathbf{x}=(x,y,z)$ and have a velocity in an elementary volume $d\mathbf{v}$ centered in $\mathbf{v}=(v_x,v_y,v_z)$.

Consequently, the macroscopic quantities as mass density ρ, momentum $\rho\mathbf{u}$ and total energy E are defined as the five first moments of f with respect to the velocity variable, namely:

$$(\rho(t,\mathbf{x}),\rho\mathbf{u}(t,\mathbf{x}),E(t,\mathbf{x}))=\int_{\mathbb{R}^3}(1,\mathbf{v},\frac{1}{2}|\mathbf{v}|^2)f(t,\mathbf{x},\mathbf{v})\,d\mathbf{v}. \tag{1}$$

The temperature T of the gas is defined by the relation $E=\frac{1}{2}\rho|\mathbf{u}|^2+\frac{3}{2}\rho RT$, where R is the gas constant defined as the ratio between the Boltzmann constant and the molecular mass of the gas.

When the gas is in a thermodynamical equilibrium state, it is well known that the distribution function f is a Gaussian function $M[\rho,\mathbf{u},T]$ of $\mathbf{v}$, called Maxwellian distribution, that depends only on the macroscopic quantities :

$$M[\rho,\mathbf{u},T]=\frac{\rho}{(2\pi RT)^{\frac{3}{2}}}\exp(-\frac{|\mathbf{v}-\mathbf{u}|^2}{2RT}). \tag{2}$$

It can easily be checked that M satisfies relations (1).

If the gas is not in a thermodynamical equilibrium state, its evolution is described by the following kinetic equation

$$\partial_t f+\mathbf{v}\cdot\nabla_{\mathbf{x}}f=Q(f), \tag{3}$$

which means that the total variation of f (described by the left-hand side) is due to the collisions between molecules (described by the right-hand side). The most realistic collision model is the Boltzmann operator but it is still very computationally expensive to use. In this paper, we use the simpler BGK model [11]

$$Q(f)=\frac{1}{\tau}(M[\rho,\mathbf{u},T]-f) \tag{4}$$

which models the effect of the collisions as a relaxation of f towards the local equilibrium corresponding to the macroscopic quantities defined by (1). The relaxation time is defined as $\tau=\frac{\mu}{\rho RT}$, where μ is the viscosity of the

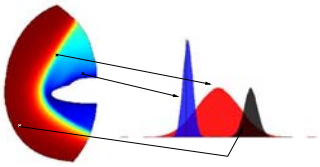

FIGURE 1. Three distribution functions in different space points of a computational domain for a re-entry problem (note that the parameters of this simulation presented here do not correspond to any real physical problem)

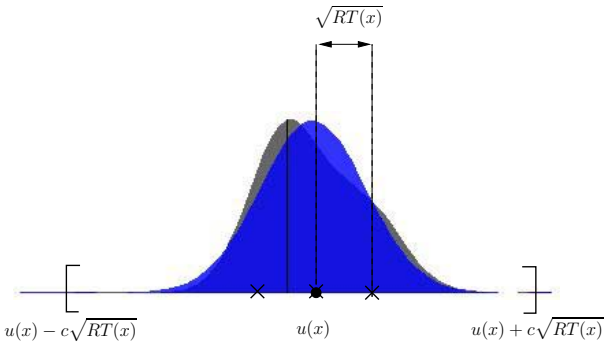

FIGURE 2. Support of a distribution function $f(x,.)$: the distribution (in black), its corresponding Maxwellian (in blue), and the corresponding support based on the bulk velocity u and the temperature T.

gas. This definition allows to match the correct viscosity in the Navier-Stokes equations obtained by the Chapman-Enskog expansion. This viscosity μ is usually supposed to fit the law $\mu = \mu_{ref}(\frac{T}{T_{ref}})^{\omega}$, where μ_{ref} and T_{ref} are reference viscosity and temperature determined experimentally for each gas, as well as the exponent ω (see a table in [1]).

The interactions of the gas with solid boundaries are described with the standard reflection models, like diffuse, specular or Maxwell reflection models.

Velocity Discretization on Cartesian Grid

In most of existing computational codes for the Boltzmann equation that use a deterministic method [7, 8], the velocity variable is discretized with a *global* Cartesian grid, that is to say the same grid for every point of the space mesh. Consequently, it is necessary to compute a priori a velocity grid that fits for every distribution function that may appear in the computation (see Fig. 1).

This means that the grid must be:

- large enough to contain the main part of the distribution functions *for every position in the computational domain*
- fine enough to capture the core of the distribution functions *for every position in the computational domain*

The corresponding parameters (bounds and step of the grid) can be determined with the following idea: at each point $\mathbf{x}$ of the computational domain, the macroscopic velocity $\mathbf{u}(\mathbf{x})$ and temperature $T(\mathbf{x})$ give information on the support of the distribution function $f(\mathbf{x},.)$. Indeed, if $f(\mathbf{x},.)$ is "not too far" from its corresponding Maxwellian, as supposed in the BGK model, its support is centered around $\mathbf{u}(\mathbf{x})$, and is essentially localized between bounds that depend on $\mathbf{u}(\mathbf{x})$ and $T(\mathbf{x})$, corresponding to the standard deviation of the Maxwellian ; that is to say $f(\mathbf{x},.)$ is very small outside (see Fig. 2 where these bounds are $\mathbf{u}(\mathbf{x}) - c\sqrt{RT(\mathbf{x})}$ and $\mathbf{u}(\mathbf{x}) + c\sqrt{RT(\mathbf{x})}$ in 1D).

When several distribution functions have to be discretized on the same grid, their supports are reasonably well approximated if the bounds are

$$v^{\alpha}_{max} = \max_{x\in\Omega}\{u^{\alpha}(\mathbf{x}) + c\sqrt{RT(\mathbf{x})}\}, \qquad v^{\alpha}_{min} = \min_{\mathbf{x}\in\Omega}\{u^{\alpha}(\mathbf{x}) - c\sqrt{RT(\mathbf{x})}\}, \tag{5}$$

and if the grid step is

$$\Delta v = a \min_{\mathbf{x}\in\Omega} \sqrt{RT(\mathbf{x})}, \tag{6}$$

where Ω is the computational domain (in space), and $\alpha = x, y, z$.

In (5)-(6), the parameters c and a can be chosen as follows. For c, statistical argument suggest that values between 3 and 4 are needed, and so $c = 3$ seems to be a good compromise between accuracy and computational cost.The parameter a allows us to adjust the grid step: it must be at most equal to 1(which ensures that there are at least six points inside the core of every distribution function), but a smaller value might be necessary for a more accurate computation.

To compute these bounds, it is necessary to a priori estimate the values of $\mathbf{u}$ and T in the computational domain. A simple way is to use a compressible Navier-Stokes pre-simulation (for which the CPU time is neglectable as compared to the Boltzmann simulation). This simulation gives the fields $\mathbf{u}^{CNS}$ and T^{CNS} in Ω, and the bounds and the grid step can be defined by formulas (5)-(6) applied to these fields, that is to say

$$v^{\alpha}_{max} = \max_{\mathbf{x}\in\Omega} |u^{\alpha,CNS}(\mathbf{x}) + c\sqrt{RT^{CNS}(\mathbf{x})}|, \qquad v^{\alpha}_{min} = \min_{\mathbf{x}\in\Omega} |u^{\alpha,CNS}(\mathbf{x}) - c\sqrt{RT^{CNS}(\mathbf{x})}|, \tag{7}$$

and

$$\Delta v = a \min_{\mathbf{x}\in\Omega} \sqrt{RT^{CNS}(\mathbf{x})}. \tag{8}$$

However, the values of the bounds are mainly determinated by the temperatures in the shock area, which can reach thousands of Kelvin. One must be careful that, at high altitude and high speeds, numerical computations of Navier-Stokes equations can underestimate those temperatures, which would lead to inappropriate bounds.

It is quite clear that the size of the velocity grid increases with the Mach number. Indeed, a large Mach number implies large upstream velocities and large temperature in the shock wave, which lead to very large bounds (see (5)), while the temperature of the body remains small, thus leading to small grid step (see (6)). For realistic re-entry problems, this can lead to prohibitively large grids. For instance, for the flow around a cylinder of radius 0.1 m at Mach 20 and altitude 90 km, formulae (7) and (8) used with a Compressible Navier-Stokes pre-simulation lead to a velocity grid of $52 \times 41 \times 41$ points and around 350 GB memory requirements with a coarse 3D mesh in space.

However, this is mainly due to the use of a Cartesian grid with a constant step. In order to decrease the size of the grid, it is attractive to refine it only wherever it is necessary, and to coarsen it elsewhere. In the following section, we show that this can be done automatically with our new algorithm.

This approach may not be well designed for cases where f is very far from it corresponding Maxwellian (like in shock interactions problem). However, for reentry problems, our assumption is realistic for transitional regime we are interested in.

AN ALGORITHM TO DEFINE LOCALLY REFINED DISCRETE VELOCITY GRIDS

Since we have to represent many distribution functions on the same grid, it is natural to refine the grid in the cores of theses distributions, and to coarsen it in the tails. (see Fig. 3).

To achieve this goal, we define the concept of "support function", and we use it to design an AMR (Adaptive Mesh Refinement) velocity grid.

The Support Function

At each point $\mathbf{x}$ of the computational space domain Ω, $\mathbf{u}(\mathbf{x})$ and $T(\mathbf{x})$ give information on the "main" support of the distribution function $f(\mathbf{x},.)$ in the velocity space: this support is centered at $\mathbf{u}(\mathbf{x})$ and of size $2c\sqrt{RT(\mathbf{x})}$ (see (5)). We define the *support function* ϕ in the velocity space as follows: for every $\mathbf{v}$ in the velocity space, we set $\phi(\mathbf{v}) = \sqrt{RT(\mathbf{x})}$ if there exists in Ω such $\mathbf{x}$ that $||\mathbf{v} - \mathbf{u}(\mathbf{x})|| \leq c\sqrt{RT(\mathbf{x})}$, that is to say that $\mathbf{v}$ is inside a support of a distribution, considered as a sphere of center $\mathbf{u}(\mathbf{x})$ and of radius $c\sqrt{RT(\mathbf{x})}$. This function gives an incomplete mapping of the velocity domain. Now we want to extend this function to the whole velocity space, so that it can be used as a refinement criterion to design an optimal velocity grid.

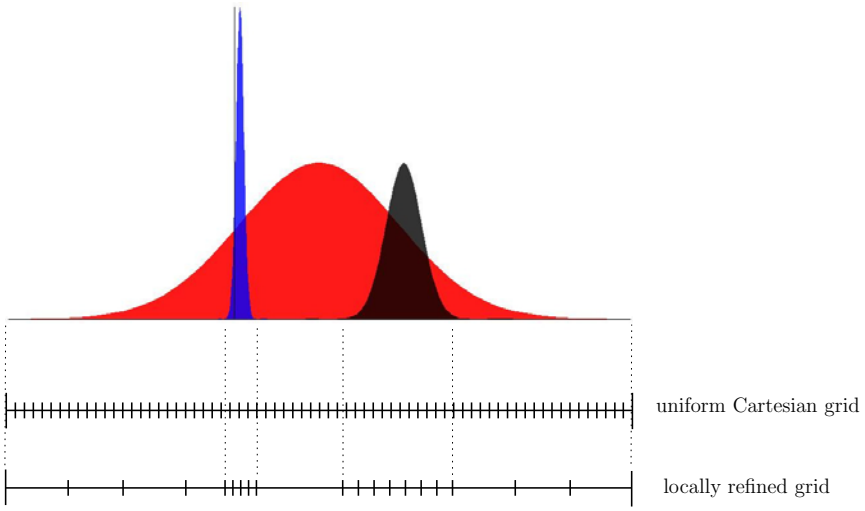

FIGURE 3. Three different distribution functions (top), the corresponding uniform Cartesian velocity grid (middle), and the grid locally refined in the support of the distributions (coarsened elsewhere).

Indeed, so far, this function is not defined for every $\mathbf{v}$, especially for large ones, as we cannot find for every $\mathbf{v}$ an appropriate pair $(\mathbf{u}(\mathbf{x}),T(\mathbf{x}))$ to match the definition of the support function. Moreover, this function might be multivalued, as there might be two different space points $\mathbf{x}_1$ and $\mathbf{x}_2$ with same velocity $\mathbf{u}(\mathbf{x}_1)=\mathbf{u}(\mathbf{x}_2)$, but with different temperatures $T(\mathbf{x}_1)\neq T(\mathbf{x}_2)$. Since our goal is to obtain for every $\mathbf{v}$ the minimum size of the support of every distribution centered around $\mathbf{v}$, these two problems can be avoided as follows :

(a) we set $\phi(\mathbf{v})=\min(\sqrt{RT(\mathbf{x}_1)},\sqrt{RT(\mathbf{x}_2)})$ if $||\mathbf{v}-\mathbf{u}(\mathbf{x}_1)||\leq c\sqrt{RT(\mathbf{x}_1)}$ and $||\mathbf{v}-\mathbf{u}(\mathbf{x}_2)||\leq c\sqrt{RT(\mathbf{x}_2)}$ with $(\mathbf{u}(\mathbf{x}_1),T(\mathbf{x}_1))\neq(\mathbf{u}(\mathbf{x}_2),T(\mathbf{x}_2))$. Indeed, the size of the grid around $\mathbf{v}$ is constrained by the distribution centered on $\mathbf{v}$ that has the smallest support, hence ϕ must have the corresponding smallest possible value.

(b) if there is no $\mathbf{x}$ such that $||\mathbf{v}-\mathbf{u}(\mathbf{x})||\leq c\sqrt{RT(\mathbf{x})}$, then $\mathbf{v}$ is in the tail of every distribution function, and there is no reason to refine the grid there, so we set $\phi(\mathbf{v})$ to its largest possible value $\phi(\mathbf{v})=\max\sqrt{RT(\mathbf{x})}$.

These ideas lead to the following algorithm for an automatic construction of ϕ. Note that below, the computational domain in space is supposed to be discretized with a mesh composed of cells numbered with three indices (i,j,k). This is not a restriction of our algorithm.

Algorithm 0.1 (Construction of the support function).

1. CNS velocity and temperature are stored in arrays $\mathbf{u}(i,j,k)$ and $T(i,j,k)$ for $i,j,k=1:i_{max},j_{max},k_{max}$.
2. construction of a (fine) Cartesian velocity grid (based on the previous global criterion (7)-(8)): points $\mathbf{v}(q)$ with $q=0:q_{max}$.
3. computation of the field $\sqrt{RT}$, stored in an descending order in the 1D array $\psi(I)$, for $I=1:i_{max}\times j_{max}\times k_{max}$, so that $\psi(I)=\sqrt{RT((i,j,k)(I))}$.
4. initialization of the array $\phi(0:q_{max})=\max(\psi)$ on the velocity grid (one value per discrete velocity)
5. *do* $I=1:i_{max}\times j_{max}\times k_{max}$ (loop on the cells of the space mesh)
   do $q=0:q_{max}$ (loop on the nodes of the velocity grid)
   – if $\mathbf{v}(q)$ is in the sphere of center $\mathbf{u}((i,j,k)(I))$ and radius $c\psi(I)$, then it is in the support of a distribution function, and we set $\phi(q):=\psi(I)$

This algorithm ensures that the array ϕ satisfies the following property.

Property 0.1. *For every q between 0 and q_{max}, we have :*

$$\phi(q)=\min\left(\min_{i,j,k}\left\{\sqrt{RT(i,j,k)}\ \textit{such that}\ \|\mathbf{v}(q)-\mathbf{u}(i,j,k)\|\leq c\sqrt{RT(i,j,k)}\right\},\max_{i,j,k}\sqrt{RT(i,j,k)}\right)$$

This means that if we take a velocity $\mathbf{v}(q)$ of the fine grid, then among all the distribution functions whose support contains $\mathbf{v}(q)$, one of them has a smallest support, and $2c\phi(q)$ is the size of its support. This support function hence tells us how the fine Cartesian grid should be refined, or coarsened, around $\mathbf{v}(q)$.

The generation of the corresponding adapted grid is described in the following section.

AMR Grid Generation

The idea is to start with a unique cell defined by the bounds of the fine velocity grid (that is the full velocity domain), and then to apply a recursive algorithm: each cell is cut if its dimensions are larger than the minimum of $a\phi$ in the cell. The algorithm is the following:

Algorithm 0.2 (Recursive refinement of a cell $\mathscr{C}$).

1. compute the minimum of the field $a\phi$ in the cell $\mathscr{C}$, that is to say the minimum of the elements of $a\phi$ that have the same indices than the discrete velocities of the fine grid included in this cell $\mathscr{C}$:

$$m := a\min\{\phi(q), \text{ such that } \mathbf{v}(q) \in \text{cell } \mathscr{C}\}$$

2. if one edge of $\mathscr{C}$ is larger than m, then the cell is cut into 8 new subcells on which the refinement algorithm is applied, recursively.
3. else, the cell is kept as it is, and the cell and its vertices are numbered.

At the end of the algorithm, the final grid satisfies the following property: every cell has a size smaller than $a\times$the minimum of the support function in the cell.

Discrete Velocity Approximation on the AMR Velocity Grid

When one wants to transform a standard discrete velocity method based on a Cartesian grid to a method using our AMR grid, two points have to be treated carefully: the computation of the moments of f, and the approximation of the collision operator. In this paper, we only use the BGK collision model of the Boltzmann equation: therefore its approximation (based on the conservative approach of [5]) reduces to the problem of a correct approximation of the moments of f, by choosing a correct quadrature formula (details on the scheme can be found in [4]). First, we propose the standard $\mathbb{Q}_1$ bilinear interpolation of f on each cell of the AMR grid (by using the four corners of the cell). In this case, the quadrature points are the nodes of the grid, and so are the discrete velocities that are used in the transport term. The quadrature weights are $\omega_q = \frac{1}{2^d}\sum_{\text{cell } \mathscr{C}_l \ni \mathbf{v}_q} |\mathscr{C}_l|$, where $d = 3$ in 3D or 2 in 2D, and $|\mathscr{C}_l|$ is the volume (or the area in 2D) of $\mathscr{C}_l$.

The number of discrete velocities can be decreased if we take a simpler $\mathbb{P}_0$ (constant per cell) interpolation. Here, the quadrature points are the centers of the cells, and so are the discrete velocities $\mathbf{v}_q$ that are used in the transport term. The weights are simply the volume (or the area) of the cells: $\omega_q = |\mathscr{C}_q|$. Note that the number of discrete velocities that are used with this approach is equal to the number of cells of the AMR grid, which is smaller than the number of nodes, see our numerical results below. We advocate the use of this latter approach, since it allows to save a lot of computer memory, and that we observed with our tests that the $\mathbb{Q}_1$ interpolation does not give more accurate results.

NUMERICAL RESULTS: A 2D EXAMPLE

For the 2D examples shown below, we use a simpler version of the code KISS, called CORBIS. This code is based on the same tools but makes use of the reduced distribution technique to reduce the velocity space to 2D only. This code only uses OpenMP directives to work on parallel computers (see [6]).

Here, we illustrate our approach with the simulation of a steady flow over a cylinder of radius 0.1m at Mach 20, for density and pressure of the air at an altitude of 90 km. The gas considered here is argon (molecular mass= 6.663×10^{-24}kg). Namely, we have $\rho = 3.17\times10^{-6}kg/m^3$, $u = 5.81\times10^3 m/s$, and $T = 242.4K$.

The space mesh uses 50×50 cells, with a refinement such that the first cell at the solid boundary is smaller than one fifth of the mean free path at the boundary.

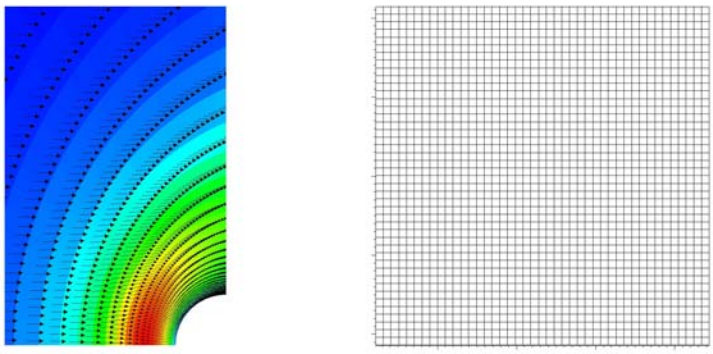

FIGURE 4. CNS velocity and temperature fields (left), corresponding fine Cartesian velocity grid (right).

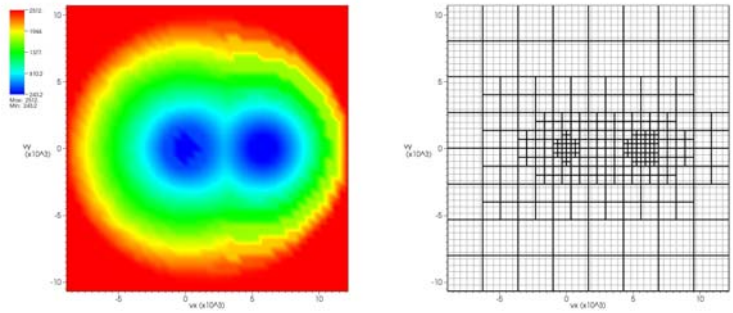

FIGURE 5. Left: support function, Right: velocity grids (solid line: induced AMR velocity grid, dotted line: initial fine Cartesian grid).

A CNS pre-simulation and the use of formula (7-8) with parameters $c = 3$ and $a = 1$ give us a fine velocity grid with 52×41 points (bounds ± 11.500 and ± 8.900 for v_x and v_y with a step of 450, in m.s^{-1}), see Fig. 4. Algorithms 0.1 and 0.2 applied to the CNS fields give the support function and the AMR velocity grid shown in Fig. 5. The AMR velocity grid has 529 points, which is four times smaller than the original fine Cartesian grid. This gain can be further increased if we define the discrete velocities as the centers of the cells of the AMR grid rather than its vertices. This gives 316 discrete velocities, hence with a gain of 6.7 times instead of 4.

First, we compare the CPU time required by the code CORBIS with the different velocity grids. While the number of iterations to reach steady state is approximately the same with both grids, the CPU time required by the original fine Cartesian velocity grid is around 7 times as large as with the new AMR grid, which is almost the same ratio as the ratio of the number of discrete velocities. The memory required with the uniform grid method is around 170 MB whereas with the use of the AMR grid, only 25 MB of memory storage is required. This shows that the new AMR grid leads to a real gain both in memory storage and CPU time.

Then, we compare the accuracy of the results with the two grids for the macroscopic quantities. We compute the normal component of the heat flux to the boundary, which is a quantity of paramount importance in aerodynamic simulations: we find a maximum relative difference lower than 5%, which is reasonably small (see the profile of this flux on Fig. 6).

We also compute the differences for the density, temperature and pressure in the whole computational domain: the mean quadratic relative difference over all the cells of the computational domain is small (5% for the density, 1% for the temperature, 0.6% for the horizontal velocity, and 1.2% for the vertical velocity), whereas some high values appears in the maximum relative differences (40% for the density and 157% for the velocity reached at the solid wall, and 69% for the temperature in the upstream flow).

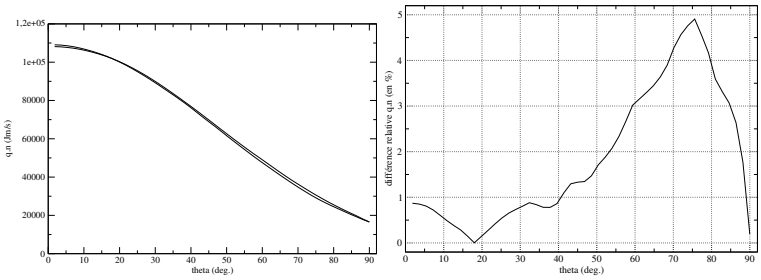

FIGURE 6. Comparison of the normal component of the heat flux to the solid wall: fluxes obtained with the fine Cartesian grid and the AMR grid (left), relative difference in percent (right).

This difference is quite large, and can be explained as follows: the smallest cells of our AMR velocity grid (that are around small velocities, like velocities at the solid wall) turn out to be smaller than the cells of the Cartesian grid (size 330 instead of 450). This means that our results with the AMR grid are probably *more accurate* than the results of the Cartesian grid. Consequently, the Cartesian grid results should not be considered as a reference for this comparison.

To confirm this analysis, we make a new simulation with a Cartesian grid with a uniform step of 330 (like the smallest step of the AMR grid). We observe that relative differences between the Cartesian grid and the AMR grid are much smaller (for example,12% for the maximum relative difference in density). The maximum relative difference is still too large for the velocity (80%, at the boundary), but this quantity is very small in this zone and probably requires smaller velocity cells (that is a smaller parameter a for both grids). However, this inaccuracy does not deteriorates the results on the other quantities, in particular for the heat flux at the boundary. Indeed, the comparison of the heat fluxes is even better, since we find a relative difference lower than 2.5%, which is excellent. The comparison in terms of CPU time and memory storage is of course more favorable to the AMR grid here.

CONCLUSION AND PERSPECTIVES

In this work, we present a new technique to determine a velocity discretization adapted for the simulation of hypersonic flows in rarefied atmosphere. The mean idea was to describe the link between the local refinement in velocity and quantities like macroscopic velocity and temperature. With this point of view, an algorithm was written in order to automatically create the support function that can be used as a refinement criterion to design an optimal velocity grid. A test case in 2D was produce to demonstrate the gain in CPU and memory storage (ratio 7). With work is naturally extended and some 3D test cases as to be produced (ratio 30 expected). As the BGK model will be extended to an ES-BGK model (to deal with the correct Prandtl number), the AMR grid technique will be derived to take into account directional temperature.

REFERENCES

1. G. Bird, *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*, Oxford Science Publications, 1994.
2. J. Burt, and I. Boyd, *J. Comput. Phys.* **28**, 460–475 (2009).
3. L. Pareschi, and G. Russo, *SIAM J. Sci. Comput.* **23**, 1253–1273 (2001).
4. L. Mieussens, *Math. Models Methods Appl. Sci.* **8**, 1121–1149 (2000).
5. L. Mieussens, *J. Comput. Phys.* **162**, 429–466 (2000).
6. K. Aoki, P. Degond, and L. Mieussens, *Comm. Comput. Phys.* **6**, 919–954 (2009).
7. V. A. Titarev, *Comm. Comput. Phys.* **12**, 162–192 (2012).
8. V. Kolobov, R. Arslanbekov, V. Aristov, A. Frolova, and S. Zabelok, *J. Comput. Phys.* **223**, 589 – 608 (2007).
9. V. Aristov, *USSR J. Comput. Math. Math. Phys.* **17(4)**, 261–267 (1977).
10. C. Baranger, J. Claudel, N. Hérouard, and L. Mieussens, "Deterministic Rarefied Flow Simulation using non Cartesian Velocity Grids," ICIAM 2011, Vancouver, Canada, 2011, minisymposium "Advanced Numerical Methods for Kinetic Simulations and Their Applications".
11. P. Bhatnagar, E. Gross, and M. Krook, *Phys. Rev.* **94**, 511–525 (1954).