

GPU-accelerated numerical simulations of the Knudsen gas on time-dependent domains

Florian de Vuyst, Francesco Salvarani

▶ **To cite this version:**

Florian de Vuyst, Francesco Salvarani. GPU-accelerated numerical simulations of the Knudsen gas on time-dependent domains. 2012. hal-00687566v2

HAL Id: hal-00687566
https://hal.science/hal-00687566v2

Preprint submitted on 14 Sep 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

GPU-ACCELERATED NUMERICAL SIMULATIONS OF THE KNUDSEN GAS ON TIME-DEPENDENT DOMAINS

FLORIAN DE VUYST AND FRANCESCO SALVARANI

Abstract. We consider the long-time behaviour of a free-molecular gas in a time-dependent vessel with absorbing boundary, in any space dimension. We first show, at the theoretical level, that the convergence towards equilibrium heavily depends on the initial data and on the time evolution law of the vessel. Subsequently, we describe a numerical strategy to simulate the problem, based on a particle method implemented on general-purpose graphics processing units (GPGPU). We observe that the parallelization procedure on GPGPU allows for a marked improvement of the performances when compared with the standard approach on CPU.

1. Introduction

In this paper we consider an ideal rarefied gas contained in a deformable vessel with absorbing boundary, kept at a uniform temperature.

When the mean free path of the gas molecules has the same order of magnitude of a characteristic macroscopic length (for example, the size of the vessel), continuum fluid dynamics is not valid, and we need to use a kinetic description, as described, for example, in [5, 6].

However, since kinetic equations are defined on the phase space of the system, the number of independent variables makes, from a computational point of view, their numerical simulation much more expensive than the numerical computations of standard fluid mechanics.

A popular strategy for the numerical study of kinetic equations is the particle method, which has been declined in a number of interesting variants, such as, for example, the Bird method [4] and the Nanbu-Babowski method [7, 2].

The particle method is based on the physical ground that leads to write the equations (and not on the equations themselves); among the main quantities used in the simulations, indeed, one can find the mean free path between collisions, the particle free flow and the collision frequency.

In practice, the particle method simulates a gas, composed by a number of particle that is not far from the Avogadro number, by a reduced set of particles (whose number can be handled by a computing machine) that simulate the global behaviour of the whole gas.

It is clear that such a method can obtain great benefits from a parallelization procedure.

Date: September 14, 2012.

Key words and phrases. Transport equation, Particle methods, GPGPU, moving domains.

Indeed, the evolution of the particles is caused by the free flow (which can be treated in a completely parallel way) and, in the case of collisional gases, by the mutual interactions between particles, which are instantaneous in time, local in space, and involving only a pair of particles in each collision.

In the situation studied in the paper, we consider the case where the gas is so rarefied that the interaction between the gas molecules is neglected. Such a gas is called the free-molecular gas or the Knudsen gas and, in the case of moving domains, exhibits a variety of different behaviours in what concerns its long-time evolution.

After a theoretical analysis of the long-time asymptotics of the Knudsen gas in a moving vessel, we implement a simulation algorithm of the problem suitable to run on a general purpose GPU (GPU-based architecture dedicated to high performance computing).

GPUs today reach a performance up to 3 TFLOPS at a fraction of the cost of CPU-based architectures of similar performance. The use of this kind of hardware architectures allows one to handle a greater number of numerical particles (up to 10^8 per board) and hence to noticeably improve both the speed and the precision of the numerical results.

Finally, we notice that, the free transport equation being the first step of a splitting method concerning the numerical simulation of the Boltzmann equation, the results presented here can be useful in the simulation on GPU of the full Boltzmann equation on moving domains.

The paper is organized as follows: in the next section we provide the theoretical background of the problem; then, in Section 3, we explain the numerical method. Finally, in Section 4, we show and analyse our numerical simulations.

2. Basic mathematical properties of the problem

In order to give a solid benchmark framework for our computations, we first set up the theoretical aspects of the problem, and prove the main results that we will reproduce in the numerical simulations.

Consider an open bounded domain $\Omega^0 \subset \mathbb{R}^d$, $d \in \mathbb{N}$, with a regular boundary $\Gamma^0 = \partial\Omega^0$, at least of class C^1, which evolves with respect to time.

The deformation of Ω^0 that leads to the domain $\Omega^t \subset \mathbb{R}^d$, $t \in \mathbb{R}^+$ is described by a function

$$\Phi : \mathbb{R}^+ \times \mathbb{R}^d \to \mathbb{R}^d.$$

In other words, for any $x_0 \in \Gamma^0$, the vector $x_0^t = \Phi(t, x_0)$ defines the surface $\Gamma^t = \partial\Omega^t$, which is the boundary of the domain of the problem at time t.

We suppose that Φ is regular, at least of class C^1, and moreover that $\|\partial_t\Phi(t, x_0)\|_{\mathbb{R}^d} \leq c$ ($c > 0$), uniformly for all $x_0 \in \Omega^0$ and for all $t \in \mathbb{R}^+$. In other words, we suppose that the velocity of the boundary is finite.

The problem considered in the paper consists in studying the time evolution of a collisionless gas, described by the classical transport equation

$$\frac{\partial f}{\partial t} + v \cdot \nabla_x f = 0, \qquad (t, x, v) \in \mathbb{R}^+ \times \Omega^t \times \mathbb{R}^d, \tag{2.1}$$

where $f := f(t,x,v)$ is the density of particles which at time t and point x move with velocity v.

The problem is supplemented with the initial and boundary conditions

$$f(0,x,v) = f^{\text{in}}(x,v) \in L^p(\Omega^0 \times \mathbb{R}^3), \qquad f(t,x,v)|_{x\in\Gamma^t} = 0 \tag{2.2}$$

where $1 \leq p \leq +\infty$.

Restrictions on the regularity of the initial data and on the properties of the domain Ω^t will be consider later on, when focusing on some special features of the problem.

A crucial tool in studying Equation (2.1) is the *forward exit time* for a particle starting from $x \in \Omega^0$ in the direction $v \in \mathbb{R}^d$, defined as

$$\tau_{\Omega^t}(x,v) = \inf\{t > 0 \ : \ x + tv \in \Gamma^t\}.$$

In what follows, we denote by $B_q(s)$ the ball centred in $s \in \mathbb{R}^d$ of radius $q \in \mathbb{R}^+$ and its volume by $\text{meas}(B_q(s))$, that is

$$\text{meas}(B_q(s)) = \frac{\pi^{d/2}}{\Gamma(n/2+1)}.$$

The proof of the following theorem is immediate:

Theorem 2.1. *Suppose that $\|\partial_t \Phi(t,x_0)\|_{\mathbb{R}^d} \leq c$ uniformly for all $x_0 \in \Omega^0$ and for all $t \in \mathbb{R}^+$. Then, Problem (2.1)-(2.2) admits a unique solution for any $p \in [1,+\infty]$ and for all $t \in \mathbb{R}^+$. Moreover, the only stationary solution of this problem in the class of L^p-functions is $f = 0$.*

Proof. The unique explicit solution of the problem is

$$f(t,x,v) = f^{\text{in}}(x - vt, v)\mathbb{1}_{\tau_{\Omega^t}(x,v)>t}, \tag{2.3}$$

as shown by the method of characteristics.

It is finally straightforward to prove that the only possible stationary solution of Equation (2.1), with vanishing boundary condition is $f = 0$, since the stationary version of equation (2.1) reduces to $v \cdot \nabla_x f = 0$.

Hence the theorem is fully proven.

□

Another important feature of the problem is its long-time asymptotics.

The situation is more complex than the corresponding problem in a fixed domain, and a variety of behaviours occurs: sometimes the convergence to zero is achieved in a finite time, in some other cases the speed of convergence is algebraic, sometimes there is no convergence towards equilibrium.

We point out that the possibility of algebraic convergence is not surprising, as shown by Bernard and Salvarani in [3] for the linear transport equation and by Aoki and Golse [1] for the Knudsen gas in a vessel whose wall is kept at a uniform and constant temperature, assuming diffuse reflection on the vessel wall.

We have the following result:

Theorem 2.2. *Let us consider the initial-boundary value problem (2.1)-(2.2). The following behaviours concerning the long-time asymptotics of the problem are possible:*

1) *If Ω^t is uniformly bounded for all $t \in \mathbb{R}^+$ by an hypersphere $B_R(0)$, with $R > 0$, and there exists $V > 0$ such that $\mathrm{supp}\,(f^{\mathrm{in}}) \subset \Omega^0 \times (\mathbb{R}^d \setminus B_V(0))$, then the extinction time of the solution f is finite.*
2) *If Ω^t is uniformly bounded for all $t \in \mathbb{R}^+$ by an hypersphere $B_R(0)$, with $R > 0$, and $f^{\mathrm{in}} \in L^\infty(\Omega^0 \times \mathbb{R}^d)$, the solution converges to zero. Moreover, we have the following estimate on the speed of convergence towards the asymptotic state:*

$$\|f(t,\,\cdot\,,\,\cdot\,)\|_{L^1(\Omega^t\times\mathbb{R}^d)} \leq \mathrm{meas}(B_R(0))\|f^{\mathrm{in}}\|_{L^\infty(\Omega^0\times\mathbb{R}^d)} \left(\frac{4R}{t}\right)^d.$$

3) *If Ω^t is not uniformly bounded by an hypersphere $B_R(0)$, with $R > 0$, then it may exist no stationary state.*

Proof. The strategy of proof is different in the three cases.

1) Since Ω^t is uniformly bounded for all $t \in \mathbb{R}^+$ by the hypersphere $B_R(0)$, and $v \in (\mathbb{R}^d \setminus B_V(0))$, by looking at the explicit solution (2.3) we can deduce that the worst forward exit time is $\tau_{B_R(0)}(x,v) \leq 2R/V$, and hence the extinction time is finite.
2) We use a technique based on an upper solution of the problem, defined on a fixed hypersphere $B_R(0)$, with $R > 0$ (which is such that $\Omega^t \subset B_R(0)$ uniformly for all $t \in \mathbb{R}^+$ by hypothesis).

   We hence consider the function g, solution of the problem

$$\frac{\partial g}{\partial t} + v \cdot \nabla_x g = 0, \qquad (t,x,v) \in \mathbb{R}^+ \times B_R(0) \times \mathbb{R}^d,$$

   with the initial conditions $g(0,x,v) = f^{\mathrm{in}}(x,v)$ when $x \in \Omega^0$ and $g(0,x,v) = 0$ when $x \in B_R(0) \setminus \Omega^0$, and with absorbing boundary data

$$g(t,x,v)|_{x\in\partial B_R(0)} = 0.$$

   It is straightforward to prove that

$$g(t,x,v) = f^{\mathrm{in}}(x - vt, v)\mathbb{1}_{\tau_{B_R(0)}(x,v)>t},$$

   and that $g \geq f$ for a.e. $(t,x,v) \in \mathbb{R}^+ \times \Omega^t \times \mathbb{R}^d$. In general, we have that

$$\|g(t,\,\cdot\,,\,\cdot\,)\|_{L^1(B_R(0)\times\mathbb{R}^d)} \leq \|f^{\mathrm{in}}\|_{L^1(\Omega^0\times\mathbb{R}^d)}.$$

   We now suppose that $t > 0$. Thanks to the explicit form of g and the fact that its domain of definition is fixed, we have that

$$\|g(t,\,\cdot\,,\,\cdot\,)\|_{L^1(B_R(0)\times\mathbb{R}^d)} \leq \int_{B_R(0)\times\mathbb{R}^d} f^{\mathrm{in}}(x,v)\mathbb{1}_{x\in\Omega^0}\,dxdv \leq$$

$$\|f^{\mathrm{in}}\|_{L^\infty(\Omega^0\times\mathbb{R}^d)} \int_{B_R(0)} \int_{-2R/t}^{2R/t} \dots \int_{-2R/t}^{2R/t} dxdv_1 \dots dv_1 dv_d =$$

$$\mathrm{Vol}(B_R(0))\|f^{\mathrm{in}}\|_{L^\infty(\Omega^0\times\mathbb{R}^d)} \left(\frac{4R}{t}\right)^d,$$

   which is the desired result.

3) We give the following counterexample: $\Omega^t = B_{1+2t}(0)$ and $f^{\text{in}} = \mathbb{1}_{\|v\|_{\mathbb{R}^d} \leq 1}$ on $B_1(0) \times \mathbb{R}^d$. The explicit form (2.3) of the solution implies that

$$\int_{\mathbb{R}^d \times \mathbb{R}^d} f dx dv = \text{meas}(B_1(0))^2$$

for all time $t > 0$, and hence there exists no extinction time.

□

Remark 2.3. *The situations listed on Theorem 2.2 are not exhaustive nor optimal. They depend, indeed, on the geometry of the moving domain, together with the functional form of the initial conditions.*

3. The numerical strategy

The basic point of a particle method is the discretization of the unknown function f by means of a sum of Dirac masses, centred in $(x_k(t), v_k(t))_{1 \leq k \leq N}$, which represent a set composed by $N \in \mathbb{N}$ macro-particles that evolve in the phase space of a system.

More precisely, our working hypothesis is the approximation

$$f = \sum_{k=1}^{N} \omega_k \, \delta(x - x_k(t)) \, \delta(v - v_k(t)),$$

where ω_k is the weight of the k-th particle.

Once the number N of numerical particles has been chosen, we initialize the problem by approximating the initial condition f^{in} by means of

$$f^{\text{in}}(x, v) = \sum_{k=1}^{N} \omega_k \, \delta(x - x_k^0) \, \delta(v - v_k^0),$$

and then the time evolution of the system is obtained by deducing the time evolution of the macro-particles on the characteristic curves of the problem (2.1)-(2.2). In our case, we obtain the following evolution rule for the free flow of the numerical particles:

$$\begin{cases} x_k'(t) = v_k(t), \\ \\ v_k'(t) = 0, \end{cases}$$

under the initial conditions $(x_k(0), v_k(0)) = (x_k^0, v_k^0)$, for all $1 \leq k \leq N$.

The method is naturally conservative and highly parallelizable. In the next section, we shall give the details of the numerical strategy.

4. Implementation and benchmarks

We have produced two different codes, the first one is a classical sequential code, and the second one is based on a thread-based GPU parallelization technique.

The numerical tests on the long-time asymptotics of the problem allow to evaluate the mean GPU speedup factor compared to a sequential monothread CPU computation.

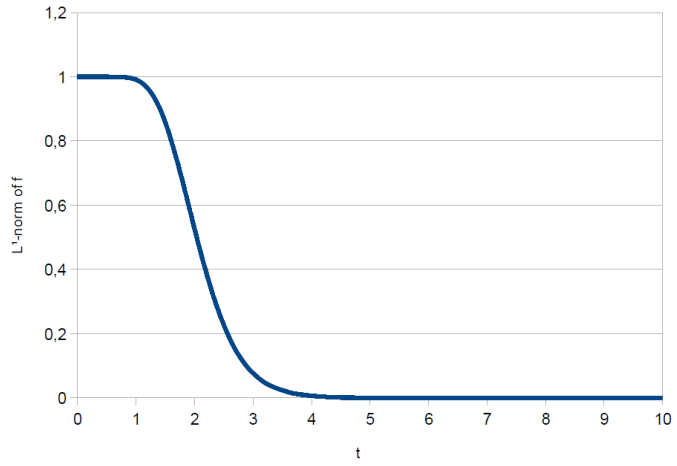


FIGURE 1. Evolution of the L^1-norm of the discrete distribution generated by the initial condition f_1^{in} into the sphere $B_{r(t)}(0)$.

The speedup factor is evaluated as a function of the number of particles N. The hardware being used is a PC with an `Intel Xeon CPU E56202.40 GHz` and a `nVIDIA GPGPU TESLA C2070`, 448 cores with 6 GB memory, 14 multiprocessors, 510 Gflop/s in double precision and 960 Gflop/s in single precision.

All the computations presented here have been performed in single precision. The `nVIDIA CUDA toolkit 4.1` is used.

In all computations, we have considered a three-dimensional spatial problem, i.e., all positions and velocities of the particles belong to a subset of $\mathbb{R}^3$. This means that each particle is individuated at each time instant t by a set of six independent variables in the phase-space $\mathbb{R}^3 \times \mathbb{R}^3$.

We have then performed three numerical tests which represent the three prototypes of the behaviours described in Theorem 2.2.

Let $x = (x_1, x_2, x_3) \in \mathbb{R}^3$ and $v = (v_1, v_2, v_3) \in \mathbb{R}^3$. We first consider the sphere $B_r(0)$, where the radius $r = r(t)$ is given by

$$r^2(t) = e^{-t} + 2(1 - e^{-t}), \quad t \geq 0.$$

We numerically studied the time evolution of the problem with two different initial data,

$$f_1^{\text{in}} = \mathbb{1}_{\{(x,v)\in\mathbb{R}^3\times\mathbb{R}^3\,:\,|x_i|\leq 1/2,\, 1/4\leq v_i\leq 1,\, i=1,2,3\}}$$

and

$$f_2^{\text{in}} = \mathbb{1}_{\{(x,v)\in\mathbb{R}^3\times\mathbb{R}^3\,:\,|x_i|\leq 1/2,\, |v_i|\leq 1,\, i=1,2,3\}},$$

by using $2^{20} = 1,048,576$ numerical particles, and for $t \in [0, 10]$.

We observe the finite in time extinction of the solution for the initial condition f_1^{in} (Figure 1) and the extinction with time rate t^{-3} for the initial condition f_2^{in}, as expected, the latter plotted both in linear and in logarithmic scale (Figures 2 and 3).

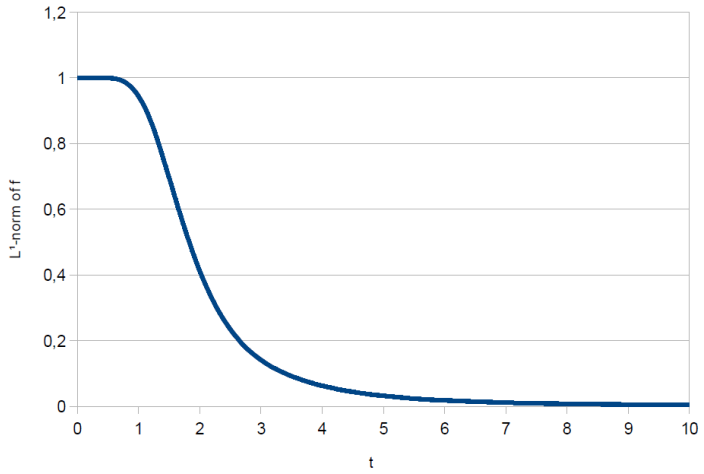


FIGURE 2. Evolution of the L^1-norm of the discrete distribution generated by the initial condition f_2^{in} into the sphere $B_{r(t)}(0)$ (linear scale).

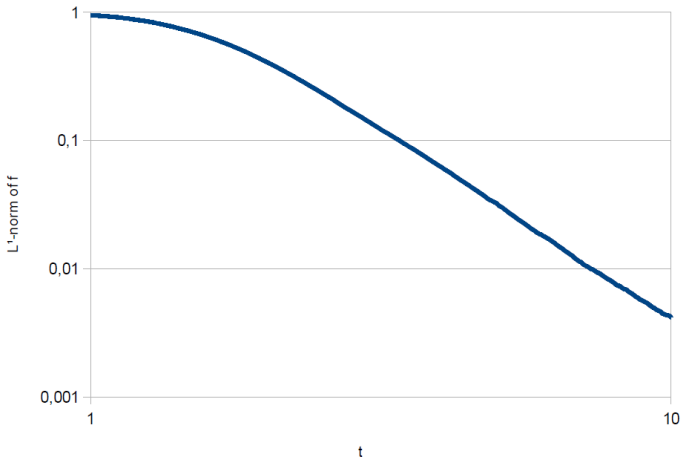


FIGURE 3. Evolution of the L^1-norm of the discrete distribution generated by the initial condition f_2^{in} into the sphere $B_{r(t)}(0)$ (logarithmic scale).

In this case we have also computed the numerical decay rate with a least square methods on the time interval $t \in [6, 10]$. The numerical decay rate is equal to 2.98, a value which is very close to the theoretical one, $d = 3$.

We finally consider the domain $B_{1+3t/5}(0)$, with initial data f_1^{in}. In Figure 4 we recover the behaviour predicted by the third point in Theorem 2.2, namely the absence of an asymptotic equilibrium state (which should have L^1-norm equal to zero). Again, in the simulation shown in Figure 4 we have used 2^{20} numerical particles.

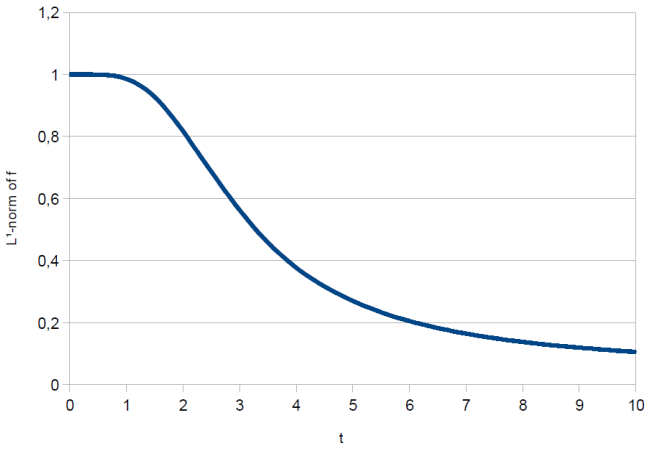


FIGURE 4. Evolution of the L^1-norm of the discrete distribution generated by the initial condition f_1^{in} into the sphere $B_{1+3t/5}(0)$.

In all cases, the GPU results are identical to the CPU ones up to the rounding error.

On each time step the particles are moved according to their velocity. The particle weight is set to zero once a particle leaves the sphere of interest.

In the parallelized code, this task is done in parallel on the GPU. For each discrete instant, we also compute the L^1-norm of the distribution according to the discrete formula:

$$\|f(t,\cdot,\cdot)\|_{L^1(\mathbb{R}^3\times\mathbb{R}^3)} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{(\|x_i(t)\|\le r(t))}.$$

This GPU parallel reduction requires particular attention for performance purpose. Actually we use the GPU shared memory for partial reduction on each `CUDA` block. We based our implementation from the `CUDA GPU Computing SDK 4.1`, reduction example. The numerical complexity of a reduction operation (sum, max, ...) on an array of size N over P processors is known to be $O(N/P + \log(N))$.

The `CUDA` kernel function of the particle positions and the weight update are given by the following instructions:

```
dim3 grid( 448, 1, 1 );
dim3 block( 512, 1, 1 );
k = (float)0.0;
while (k<Tmax)
    {
k += dt;
c = (float)exp(-1.0*k)*1.0 + (1.0-exp(-1.0*k))*2.0;
//
xupdate<<<grid, block>>> (dev_x1, dev_x2, dev_x3, dev_v1,
                          dev_v2, dev_v3, dev_weight, c);
reduce<float>(N, numThreads, numBlocks, whichKernel,
```

```
              dev_weight, d_odata);
//
// sum partial sums from each block on CPU
// copy result from device to host
    // (of size numBlocks*sizeof(float))
    cudaMemcpy(h_odata, d_odata, numBlocks*sizeof(float),
               cudaMemcpyDeviceToHost);
gpu_result = (float)0.0;
// small loop
for (int i=0; i<numBlocks; i++)  {gpu_result += h_odata[i];
}
```

This has to be compared with the C code of the time loop including the call of the particle update kernel and the reduction kernel for the computation of the L^1-norm:

```
__global__ void xupdate(float *x1, float *x2, float *x3,
                        float *v1, float *v2, float *v3,
                        float *weight, float c)  {
  int tid = threadIdx.x + blockIdx.x * blockDim.x;
  while (tid < N) {
    x1[tid] += dt * v1[tid];
    x2[tid] += dt * v2[tid];
    x3[tid] += dt * v3[tid];
    weight[tid] *= (x1[tid]*x1[tid] + x2[tid]*x2[tid] +
                    x3[tid]*x3[tid] < c);
    tid += blockDim.x * gridDim.x;
  }
}
```

We have then compared the results of the parallel code with respect to a standard C sequential code.

The GPU parallel code does not take into account the reduction of the active particles due to the absorbing moving boundary.

Indeed, a dynamical particle table would require many memory accesses to be managed, that would lead either to a loss of performance with respect to the code without particle reduction or a strong implementation effort, requiring high GPU programming skills.

On the other hand, the sequential code can be easily optimized by remapping the particles with positive weight and by eliminating the particles that are absorbed by the boundary.

The algorithm that reduces the current number N of active particles has been implemented in C language as follows:

```
   j=0;
    for(i=0;i<N;i++)
    {
if ( weight[i] > 0.0)
{
        x1[j] = x1[i];
        v1[j] = v1[i];
        x2[j] = x2[i];
```

```
        v2[j] = v2[i];
        x3[j] = x3[i];
        v3[j] = v3[i];
        weight[j]= weight[i];
        j=j+1;
}
    }
    N=j;
```

In what follows, we consider only the simulation generated by the initial condition f_2^{in} into the sphere $B_{r(t)}(0)$. The other tests give very similar results, and hence they are not reported here.

On Table 1, we give the mean speedup factor obtained on the `C2070` board, where the number of particles N varies from $N = 2^{19} = 524,288$ to $N = 2^{23} = 33,554,432$ particles, with $t \in [0, 4]$. In all tests, the number of particles with positive weight at the final time $t = 4$ is only the 6.37% of the initial particles.

One can observe a speedup factor of over 25 for $N = 2^{25}$ particles. For relatively small numbers of particles, the speedup varies between 17 and 25. This behaviour is certainly due to a stronger rate of memory communication.

Nb of particles N	GPU time (sec)	GPU/CPU speedup factor
2^{19}	0.99	17.6
$2^{20} = 1,048,576$	1.96	18.0
2^{21}	3.89	18.7
2^{22}	7.37	22.6
2^{23}	14.78	24.4
2^{24}	29.60	24.7
$2^{25} = 33,554,432$	60.39	26.3

TABLE 1. Total GPU time computation and speedup factor relatively to a sequential (optimized) CPU computation.

We would like to end this section with a general comment and a deep question about code comparability.

It is, of course, difficult to define an absolute measure of performance speedup between a sequential code and a GPU-accelerated one.

It has indeed been shown that code optimizations which may be done on the sequential code are not always relevant for GPU implementations: the constraints on the algorithm are strongly depending on the physical behaviour of the problem and some strategies, that are very efficient in a sequential code, can be counter-productive on a parallel code (for example, when they would imply a lot of memory rearrangement).

An example of this behaviour can be seen when compared the sequential code that we used here with the same code without particle reduction: we found that the speedup factors are very sensitive to the initial condition, on the evolution law of the moving domain and the on time interval.

By using as initial condition f_2^{in} into the sphere $B_{r(t)}(0)$, with 10^5 particles, the code with particle reduction is 1.45 times faster than the code without reduction when the simulation covers the time interval [0,5], whereas the code without reduction is twice faster than the code with reduction when the simulation covers the time interval [0,1].

Moreover, because CPU are generally made of 4 or 6 cores, researchers are disputing the fact that performance speedup should be measured from a monothread computation rather than a multithreaded run (using `openMP` for example).

For that reason, it is of course of paramount importance to clearly specify the comparison methodology.

5. Conclusions

We have presented an implementation of a GPU-accelerated particle method to numerically simulate a Knudsen gas in a vessel with moving boundary with absorbing boundary conditions.

After proving some possible dynamics of convergence to equilibrium, we have recovered, in the numerical simulations, the theoretical results.

The parallelized code gives a considerable improvement of the performances of the code (up to a factor 26) and allows us to handle a greater number of numerical particles than those allowed by standard sequential programming. Since the free transport step is also an essential part of any particle method for space-inhomogeneous kinetic system, this improvement of the performances will be useful also for the simulation of collisional gases.

Acknowledgements

The authors would like to thank `nVIDIA` for the *TESLA equipment Grant* in 2011 that made this work possible.

References

[1] Kazuo Aoki and François Golse. On the speed of approach to equilibrium for a collisionless gas. *Kinet. Relat. Models*, 4(1):87–107, 2011.

[2] H. Babovsky. On a simulation scheme for the Boltzmann equation. *Math. Methods Appl. Sci.*, 8(2):223–233, 1986.

[3] Étienne Bernard and Francesco Salvarani. On the convergence to equilibrium for degenerate transport problems, 2012.

[4] G. A. Bird. *Molecular gas dynamics and the direct simulation of gas flows*, volume 42 of *Oxford Engineering Science Series*. The Clarendon Press Oxford University Press, New York, 1995. Corrected reprint of the 1994 original, With 1 IBM-PC floppy disk (3.5 inch; DD), Oxford Science Publications.

[5] Carlo Cercignani. *Rarefied gas dynamics*. Cambridge Texts in Applied Mathematics. Cambridge University Press, Cambridge, 2000. From basic concepts to actual calculations.

[6] Carlo Cercignani, Reinhard Illner, and Mario Pulvirenti. *The mathematical theory of dilute gases*, volume 106 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 1994.

[7] Kenichi Nanbu. Direct simulation scheme derived from the Boltzmann equation I. monocomponent gases. *J. Phys. Soc. Japan*, 49(5):2042–2049, 1980.

F.D.V: Centre de Mathématiques et de leurs Applications, UMR 8536, École Normale Supérieure de Cachan, 61 avenue du Président Wilson, 94235 Cachan France

E-mail address: devuyst@cmla.ens-cachan.fr

F.S.: Dipartimento di Matematica F. Casorati, Università degli Studi di Pavia, Via Ferrata 1, I-27100 Pavia, Italy

E-mail address: francesco.salvarani@unipv.it