Two-fluid compressible simulations on GPU cluster

Philippe Helluy, Jonathan Jung

Two-fluid compressible simulations on GPU cluster

Philippe Helluy* and Jonathan Jung†

Abstract

In this work we propose an efficient finite volume approximation of two-fluid flows. Our scheme is based on three ingredients. We first construct a conservative scheme that removes the pressure oscillations phenomenon at the interface. The construction relies on a random sampling at the interface [6, 5]. Secondly, we replace the exact Riemann solver by a faster relaxation Riemann solver with good stability properties [4]. Finally, we apply Strang directional splitting and optimized memory transpositions in order to achieve high performance on Graphics Processing Unit (GPU) or GPU cluster computations.

1 Introduction

We are studying the numerical resolution of a two-fluid compressible fluid flow. The model is the Euler equations with an additional transport equation of a color function φ. The function φ is equal to 1 in the gas and 0 in the liquid. It allows locating the two-fluid interface. We consider the system

$$\partial_t W + \partial_x F(W) + \partial_y G(W) = 0, \tag{1}$$

where

$$\begin{aligned} W &= (\rho, \rho u, \rho v, \rho E, \rho\varphi)^T, \\ F(W) &= (\rho u, \rho u^2 + p, \rho uv, (\rho E + p)u, \rho u\varphi)^T, \\ G(W) &= (\rho u, \rho uv, \rho v^2 + p, (\rho E + p)v, \rho v\varphi)^T. \end{aligned}$$

The pressure law p is a stiffened gas pressure law

$$p(\rho, e, \varphi) = (\gamma(\varphi) - 1)\rho e - \gamma(\varphi) p_\infty(\varphi), \tag{2}$$

where $e = E - \frac{u^2+v^2}{2}$, and

$$(\gamma, p_\infty)(\varphi) = \begin{cases} (\gamma_{gas}, p_{\infty,gas}), \text{ if } \varphi = 1, \\ (\gamma_{liq}, p_{\infty,liq}), \text{ if } \varphi = 0. \end{cases}$$

*IRMA, Université de Strasbourg & TONUS, INRIA Nancy - Grand Est
†LJLL, Université Pierre et Marie Curie (UPMC), Paris VI

Recall that the system (1) coupled with the pressure law (2) is hyperbolic on the domain

$$\Omega : = \left\{ W = (\rho, \rho u, \rho v, \rho E, \rho\varphi) \in \mathbb{R}^5,\ \rho > 0, \right.$$
$$\left. \varphi \in [0;1],\ p\left(\rho, E - \frac{u^2+v^2}{2}, \varphi\right) + p_\infty(\varphi) > 0 \right\}.$$

The main point is that the hyperbolic set Ω is generally not convex if $p_{\infty,1} \neq p_{\infty,2}$ (see [12]).

Classic conservative finite volume schemes generally produce an artificial diffusion of the mass fraction φ. In the numerical approximation we may observe a mixture zone $1 > \varphi > 0$. This artificial mixing zone implies a loss of velocity and pressure equilibrium at the interface. It is possible to recover a better equilibrium by relaxing the conservation property of the scheme as in [17].

Independently of the pressure oscillations, problem of stability appears for classic schemes. This comes from the non-convexity of the hyperbolic set Ω [12]. In this paper, we use a random sampling projection at the two-fluid interface. It allows preserving the non convex hyperbolic domain. The method was first proposed in [6] for a particular traffic flow model. It does not introduce a mixture zone and extends the one-dimensional method described in [2]. More precisely, we construct a numerical scheme that preserves the hyperbolic domain without diffusion $\mathcal{H}$,

$$\mathcal{H} := \Omega_0 \cup \Omega_1, \tag{3}$$

where for all $\varphi_0 \in [0;1]$, the convex set Ω_{φ_0} is

$$\Omega_{\varphi_0} : = \left\{ W = (\rho, \rho u, \rho v, \rho E, \rho\varphi) \in \mathbb{R}^5,\ \rho > 0, \right.$$
$$\left. \varphi = \varphi_0,\ p\left(\rho, E - \frac{u^2+v^2}{2}, \varphi\right) + p_\infty(\varphi) > 0 \right\}.$$

2 An ALE-projection scheme with a random numerical method

2.1 Introduction

In order to compute the two-dimensional numerical solution, we use dimensional splitting. It means that each time step is split into two stages. In the first stage we solve $\partial_t W + \partial_x F(W) = 0$ and in the second stage we solve $\partial_t W + \partial_y G(W) = 0$. In addition, in our application, thanks to the rotational invariance of the Euler equations, the two resolutions are equivalent if we simply exchange the space variables x and y and the velocity components u and v. It is thus enough to construct a scheme for solving the one dimensional system

$$\partial_t W + \partial_x F(W) = 0. \tag{4}$$

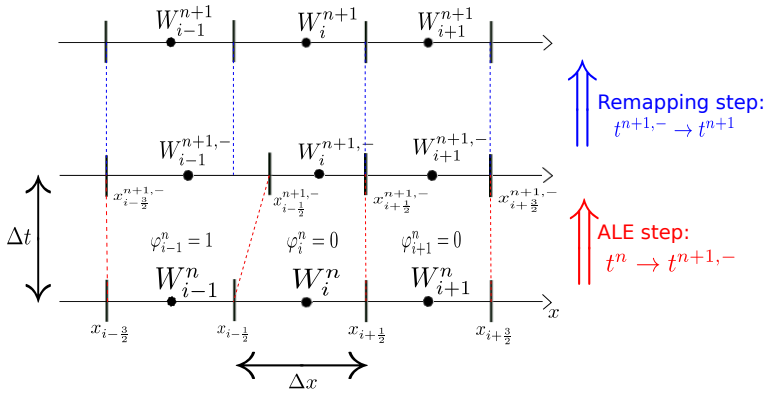


Figure 1: Structure of the ALE-projection scheme.

We generalize the Lagrange-projection scheme. We replace the Lagrange step by an Arbitrary Lagrangian Eulerian step (ALE step). It allows us to switch between a Lagrangian approach at the liquid-gas interface and a Eulerian approach in the pure phases.

We want solve (4) on $[a;b] \times \mathbb{R}^+$. We consider a sequence of times t_n, $n \in \mathbb{N}$ such that the time step $\Delta t_n := t_{n+1} - t_n > 0$. We consider also a space step $h = \frac{b-a}{N}$, where N is a positive integer. We define the cell centers by $x_i = a + \left(i + \frac{1}{2}\right) h$, $i = 0 \cdots N+1$. The cell C_i is the interval $\left]x_{i-\frac{1}{2}}; x_{i+\frac{1}{2}}\right[$, where $x_{i+\frac{1}{2}} = x_i + \frac{h}{2}$. We now focus on an approximation $W_i^n \approx W(x_i; t_n)$. The boundary cell $x_{i+\frac{1}{2}}$ moves at speed $\xi_{i+\frac{1}{2}}^n$ between time t_n and t_{n+1}^-

$$x_{i+\frac{1}{2}}^{n+1,-} = x_{i+\frac{1}{2}} + \Delta t_n \xi_{i+\frac{1}{2}}^n.$$

A time iteration of the ALE-projection scheme includes two steps (see Figure 1):

- the ALE step: with W_i^n on C_i, we obtain $W_i^{n+1,-}$ on the cell $C_i^{n+1,-} = \left]x_{i-\frac{1}{2}}^{n+1,-}; x_{i+\frac{1}{2}}^{n+1,-}\right[$,
- a projection step to obtain the Euler variables at time t_{n+1} on the original cell C_i.

We use the notation $\cdot^{n+1,-}$ to characterize the value of $\cdot$ at time t_{n+1}^-, just before the projection step.

2.2 ALE step

2.2.1 Finite volume scheme

Integrating the conservation law (4) on the space-time trapezoid

$$\left\{(x,t), \quad x_{i-\frac{1}{2}} + (t-t_n)\xi^n_{i-\frac{1}{2}} < x < x_{i+\frac{1}{2}} + (t-t_n)\xi^n_{i+\frac{1}{2}}, \quad t_n < t < t^-_{n+1}\right\},$$

we obtain the finite volume scheme

$$h_i^{n+1,-}W_i^{n+1,-} = hW_i^n - \Delta t_n \left(F(W_i^n, W_{i+1}^n, \xi^n_{i+\frac{1}{2}}) - F(W_{i-1}^n, W_i^n, \xi^n_{i-\frac{1}{2}})\right), \tag{5}$$

where $h_i^{n+1,-} = x^{n+1,-}_{i+\frac{1}{2}} - x^{n+1,-}_{i-\frac{1}{2}} = h + \Delta t_n(\xi^n_{i+\frac{1}{2}} - \xi^n_{i-\frac{1}{2}})$ and $F(W_L, W_R, \xi)$ is the conservative ALE numerical flux.

2.2.2 Choice for the velocity $\xi_{i+\frac{1}{2}}$ of the boundary $x_{i+\frac{1}{2}}$

The choice consists to move the boundary at the speed of the fluid only at the liquid-gas interface (see Figure 1). It means that

$$\xi^n_{i+\frac{1}{2}} = \begin{cases} u^n_{i+\frac{1}{2}} & \text{if} \quad \left(\varphi^n_i - \frac{1}{2}\right)\left(\varphi^n_{i+1} - \frac{1}{2}\right) < 0, \\ 0 & \text{otherwise,} \end{cases}$$

where $u^n_{i+\frac{1}{2}}$ is the velocity of the contact discontinuity obtained in the resolution of the Riemann problem

$$\begin{aligned} \partial_t W + \partial_x F(W) &= 0, \\ W(x,0) &= \begin{cases} W^n_i, \text{ if } x < 0, \\ W^n_{i+1}, \text{ otherwise.} \end{cases} \end{aligned}$$

2.2.3 Numerical flux

In order to compute the numerical fluxes we could use an exact Riemann solver, but it is not adapted to GPU computations. Indeed, the exact solver algorithm relies on many branch tests, which are not treated efficiently on GPU compute units. We prefer to use a two-fluid Lagrangian relaxation Riemann solver developed in [10, 12]. It is an adaptation of the entropic one-fluid Eulerian relaxation solver proposed in [4]. The numerical flux $F(W_L, W_R, \xi)$ can be written as

$$F(W_L, W_R, \xi) = \begin{cases} F(W_L) - \xi W_L, \text{ if } \xi < u_L - \frac{a_L}{\rho_L}, \\ F_1 - \xi W_1, \text{ if } u_L - \frac{a_L}{\rho_L} \le \xi < u_1 = u_2, \\ F_2 - \xi W_2, \text{ if } u_1 = u_2 \le \xi < u_R + \frac{a_R}{\rho_R}, \\ F(W_R) - \xi W_R, \text{ if } u_R + \frac{a_R}{\rho_R} \le \xi, \end{cases},$$

$$\text{where } \begin{cases} W_1 = (\rho_1, \rho_1 u_1, \rho_1 v_1, \rho_1 E_1, \rho_1 \varphi_1)^T, \\ W_2 = (\rho_2, \rho_2 u_2, \rho_2 v_2, \rho_2 E_2, \rho_2 \varphi_2)^T, \\ F_1 = u_1 W_1 + (0, \pi_1, 0, \pi_1 u_1, 0)^T, \\ F_2 = u_2 W_2 + (0, \pi_2, 0, \pi_2 u_2, 0)^T, \end{cases}$$

with

$$\begin{array}{rclcrcl}
\frac{1}{\rho_1} & = & \frac{1}{\rho_L}+\frac{a_R(u_R-u_L)+\pi_L-\pi_R}{a_L(a_L+a_R)}, & & \frac{1}{\rho_2} & = & \frac{1}{\rho_R}+\frac{a_L(u_R-u_L)+\pi_R-\pi_L}{a_R(a_L+a_R)},\\
u_1 & = & u_2=\frac{\pi_L-\pi_R+a_Lu_L+a_Ru_R}{a_R+a_L}, & & \pi_1 & = & \pi_2=\frac{a_L\pi_R+a_R\pi_L+a_La_R(u_L-u_R)}{a_L+a_R},\\
v_1 & = & v_L, & & v_2 & = & v_R,\\
e_1 & = & e_L-\frac{\pi_L^2-\pi_1^2}{2a_L^2}, & & e_2 & = & e_R-\frac{\pi_R^2-\pi_2^2}{2a_R^2},\\
E_1 & = & e_1+\frac{u_1^2+v_1^2}{2}, & & E_2 & = & e_2+\frac{u_2^2+v_2^2}{2},\\
\varphi_1 & = & \varphi_L, & & \varphi_2 & = & \varphi_R,
\end{array}$$

where a_L and a_R are defined by

$$\text{if } p_R-p_L \geq 0,\ \begin{cases} \frac{a_L}{\rho_L}=c_L+\alpha\max\left(\frac{p_R-p_L}{\rho_Rc_R}+u_L-u_R,0\right),\\ \frac{a_R}{\rho_R}=c_R+\alpha\max\left(\frac{p_L-p_R}{a_L}+u_L-u_R,0\right), \end{cases} \tag{6}$$

$$\text{if } p_R-p_L \leq 0,\ \begin{cases} \frac{a_R}{\rho_R}=c_R+\alpha\max\left(\frac{p_L-p_R}{\rho_Lc_L}+u_L-u_R,0\right),\\ \frac{a_L}{\rho_L}=c_L+\alpha\max\left(\frac{p_R-p_L}{a_R}+u_L-u_R,0\right), \end{cases} \tag{7}$$

with $\alpha=\frac{1}{2}\max\big(\gamma(\varphi_L)+1,\gamma(\varphi_R)+1\big)$ and where p_L, p_R, c_L et c_R are given by

$$\begin{array}{rclcrcl}
p_L & = & p(\rho_L,e_L,\varphi_L), & & c_L & = & c\,(\rho_L,e_L,\varphi_L)=\sqrt{\gamma(\varphi_L)\frac{p_L+p_\infty(\varphi_L)}{\rho_L}},\\
p_R & = & p(\rho_R,e_R,\varphi_R), & & c_R & = & c\,(\rho_R,e_R,\varphi_R)=\sqrt{\gamma(\varphi_R)\frac{p_R+p_\infty(\varphi_R)}{\rho_R}}.
\end{array}$$

Remark 2.1. *Generally, $\pi_1\neq p(\rho_1,e_1,\varphi_1)$ and $\pi_2\neq p(\rho_2,e_2,\varphi_2)$ and it is not possible to write $F(W_L,W_R,\xi)=F(W_*)-\xi W_*$ for some $W_*\in\mathbb{R}^5$.*

2.3 Projection step

The second part of the time step is needed for returning to the initial mesh. We have to average on the cells C_i the solution $W_i^{n+1,-}$, which is naturally defined on the moved cell $C_i^{n+1,-}=\left]x_{i-\frac{1}{2}}^{n+1,-};x_{i+\frac{1}{2}}^{n+1,-}\right[$.

We consider a pseudo random sequence $\omega_n\in]0;1[$ and we perform a pseudo-random averaging

$$W_i^{n+1}=\begin{cases} W_{i-1}^{n+1,-},\ \text{si } \omega_n<\frac{\xi_{i-\frac{1}{2}}^n\Delta t_n}{h},\\ W_i^{n+1,-},\ \text{si } \frac{\xi_{i-\frac{1}{2}}^n\Delta t_n}{h}\leq\omega_n\leq 1+\frac{\xi_{i+\frac{1}{2}}^n\Delta t_n}{h},\\ W_{i+1}^{n+1,-},\ \text{si } \omega_n>1+\frac{\xi_{i+\frac{1}{2}}^n\Delta t_n}{h}. \end{cases} \tag{8}$$

A good choice for the pseudo-random sequence ω_n is the $(5;3)$ van der Corput sequence [7]. Note that the averaging step has simpler expression if the cell does not touch the liquid gas interface. More precisely, if the cell is not at the interface, i.e. if

$$\left(\varphi_i^{n+1,-}-\frac{1}{2}\right)\left(\varphi_{i+1}^{n+1,-}-\frac{1}{2}\right)>0 \text{ and } \left(\varphi_{i-1}^{n+1,-}-\frac{1}{2}\right)\left(\varphi_i^{n+1,-}-\frac{1}{2}\right)>0,$$

as the velocities $\xi^n_{i-\frac{1}{2}}$ and $\xi^n_{i+\frac{1}{2}}$ of the boundaries $x_{i-\frac{1}{2}}$ and $x_{i+\frac{1}{2}}$ are zero, $C_i^{n+1,-} = C_i$ and we obtain

$$W_i^{n+1} = W_i^{n+1,-}.$$

2.4 Properties

Proposition 2.2. *Assume that ω follows a uniform law on $]0;1[$ and that the time step Δt_n satisfies the CFL condition*

$$\Delta t_n \max_i \left(\max \left(\left| u_i^n - \frac{a_{i+\frac{1}{2},L}}{\rho_i^n} \right|, \left| u_{i+1}^n + \frac{a_{i+\frac{1}{2},R}}{\rho_{i+1}^n} \right| \right) \right) \leq \frac{1}{2} h,$$

where $a_{i+\frac{1}{2},L}$ and $a_{i+\frac{1}{2},R}$ are given by (6)-(7) with $W_L = W_i^n$ and $W_R = W_{i+1}^n$. The ALE-projection scheme described above has the following properties:

- *it is conservative on Ω_0 and Ω_1,*
- *it is statistically conservative on $\mathcal{H}$,*
- *it is Ω_0-stable and satisfies a discrete entropy inequality on Ω_0,*
- *it is Ω_1-stable and satisfies a discrete entropy inequality on Ω_1,*
- *it is $\mathcal{H}$-stable and satisfies a statistically discrete entropy inequality on $\mathcal{H}$,*
- *it preserves constant u and p states at the two-fluid interface.*

For precisions or a proof, we refer to [12]. Remark that generally we have $a_{i-\frac{1}{2},R} \neq a_{i+\frac{1}{2},L}$. An extension to second order is proposed in [12].

3 GPU and MPI implementations

3.1 OpenCL and GPU implementation

For performance reasons, we decided to implement the 2D scheme on recent multicore processor architectures, such as a Graphic Processing Unit (GPU). A modern GPU is made of a global memory ($\approx$ 1 GB) and compute units ($\approx$ 27). Each compute unit (or work-group in the OpenCL terminology) is made of processing elements ($\approx$ 8, also called work-items) and a local cache memory ($\approx$16 kB). The same program (a kernel) can be executed on all the processing elements at the same time. There are some rules to respect. All the processing elements have access to the global memory but have only access to the local memory of their compute unit. The access to the global memory is slow while the access to the local memory is fast. The access to global memory is much faster if two neighboring processing elements read (or write) into two neighboring memory locations, in this case we speak about "coalescent memory access".

Our OpenCL implementation is described in [12, 10]. We recall only the most important steps of a time iteration.

- Computation of the CFL time step Δt_n. We compute a local time step $(\Delta t_n)_{i,j}$ on each cell and we use a *reduction algorithm* (see [3]) in order to compute $\Delta t_n = \min_{i,j}(\Delta t_n)_{i,j}$.

- We perform the ALE-projection update in x-direction. We compute the fluxes balance in the x-direction for each cell of each row of the grid: a row or a part of a row is associated to one compute unit and one cell to one processor. As of October 2012, the OpenCL implementations generally impose a limit (typically 1024) for the number of work-items inside a work-group [8]. This forces us to split the rows for some large computations. The values in the cells are then loaded into the local cache memory of the compute unit. It is then possible to perform the ALE-projection algorithm with all the data into the cache memory in order to achieve the highest performance. The memory access is coalescent for reading and writing.

- We transpose the data matrix (exchange x and y) with an optimized memory transfer algorithm [16]. The optimized algorithm includes four steps:

  - the data matrix is split into smaller tiles of size 32×32. Each tile is associated to a compute unit,
  - each tile is copied line by line from the global memory to the local memory of the compute unit. Memory access is coalescent because two successive processors read in two neighboring memory locations,
  - we transpose each 32×32 tile in the local memory,
  - each tile is copied line by line from the local memory to the global memory. The memory access is coalescent for writing.

- We perform the ALE-projection update in y-direction. The memory access is coalescent because of the previous transposition,

- We transpose again the data matrix for the next time step.

The repartition of the computational time on each kernel is the following: the ALE-projection steps represents 80%, the transposition 11% and the time step computation 9% of the global time computation.

We observe high efficiency (see Table 1) of the GPU implementation. The efficiency is explained by two important points.

- We used an optimized transposition algorithm to have coalescent access in x and y directions. Without this transposition, the computation would be 10 times slower.

- We use a relaxation solver. With this solver, fluxes have a simpler expression than the exact Godunov's flux. Indeed, with the relaxation solver fluxes are directly given from the left and right states (see Section 2.2.3). The exact solver would require solving a non linear equation. This computation involves many tests and then is not efficient on GPU. We also

Hardware	Time (s)	Speedup
AMD A8 3850 (one core)	527	1
AMD A8 3850 (4 cores)	205	2.6
NVIDIA GeForce 320M	56	9.4
AMD Radeon HD 5850	3	175
AMD Radeon HD 7970	2	263

Table 1: Simulations times on different hardware with single precisions. We observe interesting speedups for the GPU simulations compared to the one-core simulation. We also observe that OpenCL is still efficient on standard multicore CPU. The test case corresponds to the computation of 300 time steps of the algorithm on a 1024×512 grid. One numerical flux evaluation corresponds approximately to 500 floating point operations (flop). Four flux evaluations are needed per cell and per time step. The amount of computations for this test is thus of the order of 300 Gflop.

experiment our scheme with the exact solver on GPU. The computation with the exact solver is 50 times slower than with the relaxation solver.

3.2 MPI

The memory of a GPU is limited, typically to 1 gigabyte (GB), which limits us to a number of cells of the order of $25,000,000$. We will couple the parallelization on GPU (using OpenCL) with a subdomain parallelization, which uses the Message Passing Interface (MPI) standard. This allows a computation on several GPUs simultaneously. It will allow us to consider finer meshes and also reduce the computation time [13, 11, 1].

We use a standard subdomain decomposition, a GPU is associated to each subdomain. Thanks to MPI messages, we exchange the values at the subdomains boundaries. As we want a compatible decomposition with the matrix transpose algorithm, we split the initial domain only along the x-direction (see Figure 2). The exchanges between two GPUs will occur at each iteration. The GPU number l will exchange information with GPUs $l-1$ and $l+1$.

Assume that we have a cluster of L GPUs and consider a two-dimensional computation on the domain $[a;b] \times [c;d]$. We split the interval $[a;b]$ as

$$a = a_0 < a_1 < a_2 < \cdots < a_L = b,$$

with $a_l = a + l\frac{b-a}{L}$, for $l = 0, \cdots, L$.

- the computational domain $[a_l; a_{l+1}] \times [c;d]$ is associated to the GPU number $l+1$,
- each subdomain $[a_l; a_{l+1}] \times [c;d]$ is split into $(N_x - gap) \times N_y$ cells, where $gap \in \{2;5\}$ corresponds to the number of cells in the overlap between subdomains.

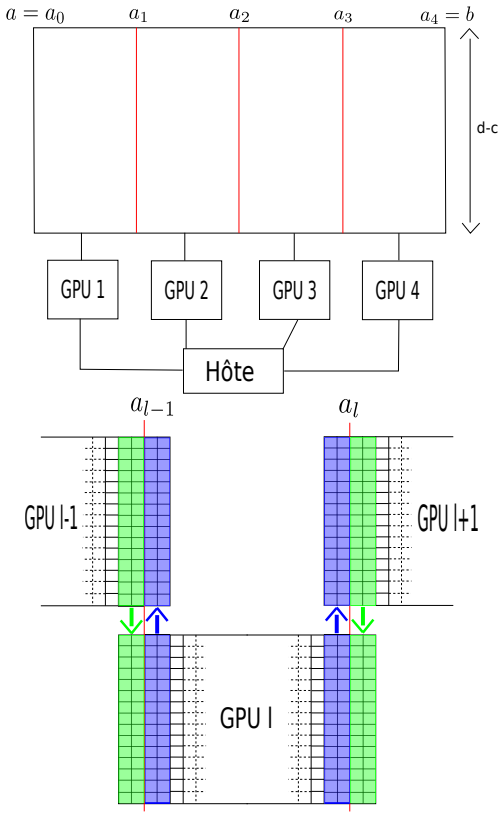


Figure 2: On the left: the computational domain is split into 4 subdomains. GPU l performs the computations for the domain $[a_{l-1}; a_l] \times [c; d]$. On the right: MPI transfers. Between each iteration, GPU l has to exchange the values on the left and right boundaries with GPUs $l-1$ and $l+1$. On the picture we consider an overlap of 2 cells but for a second order implementation we need 5 cells.

  - For a one-order computation, we use $gap = 2$ cells in the overlap: one cell for the fluxes computations and another one for the Glimm projection (8).
  - For a second order computation, we use $gap = 5$ cells. Indeed, in the ALE step, we couple a MUSCL method to a Heun's time integrator, then we need a two-cell overlap (one for the slopes of the MUSCL method and one for the flux) for each of the two steps of the Heun's time integration. We need an additional one-cell overlap for the Glimm projection (8).

- We add $gap \in \{2; 5\}$ columns on the left and right boundaries of each domain (see Figure 2). The GPU number l performs computations on $N_x \times N_y$ cells.

- we compute the stability condition. We compute the time step Δt_n^l on each subdomain l and we take

$$\Delta t_n = \min_{1 \leq l \leq L} \Delta t_n^l.$$

- on each subdomain $l = 1, \cdots, L$
  - we associate a processor to each cell,
  - we perform the time update under x-direction using the ALE- projection scheme,
  - we transpose the data table,
  - we perform the y-direction update.

  For more details, see [12].

- We perform transfers between subdomains: each domain l sends the *gap* columns inside the left edge of the subdomain to the GPU number $l-1$ and the *gap* columns inside the right edge of the subdomain to the GPU number $l+1$ (see Figure 2). The GPU number l receives the *gap* columns from GPU number $l-1$ and the *gap* columns from GPU number $l+1$ (see Figure 2).

- On each subdomain $l = 1, \cdots, L$, we transpose the data table so that the data are aligned in memory according to the x-direction for the next time step.

Remark 3.1. *For MPI communications, we need firstly to copy data from each GPU to its host (CPU). Secondly we send MPI communications between hosts (CPUs). Finally, we copy data from CPU to GPU. As we perform the MPI communications before the data matrix transposition, memory access is coalescent for reading and writing.*

Grid	1 GPU	4 GPUs	Speedup
2048×2048	14 s	14 s	1
4096×2048	22 s	16 s	1.4
4096×4096	77 s	60 s	1.3
8192×4096	150 s ?	61 s	2.5
16384×4096	600 s ?	230 s	2.6

Table 2: Simulations times for the MPI implementation on a cluster of 4 GPUs AMD Radeon HD7970. The computational domain is $[0;2] \times [0;1]$. If the GPU is not fully occupied (meshes smaller than 4096×4096), the computation times on one and on four GPUs are comparable. However if the GPU is fully occupied, for example for a grid of 16384×4096, the MPI implementation goes 2.6 times faster. This computation can not be done on only one GPU, thus some computation times are only estimated from a simple complexity analysis and marked by "?".

The MPI implementation allows considering L times finer mesh but is it faster? We test the method on a cluster of four AMD Radeon HD 7970 GPUs. The MPI communications represents globally 5% of the total time computation. The speedups are presented in the Table 2. The MPI implementation is faster with a factor 2.6.

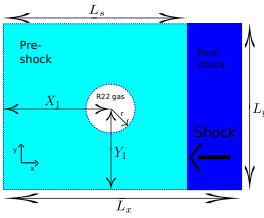
4 Numerical result

We now present a two-dimensional test that consists in simulating the impact of a Mach 1.22 shock traveling through air onto a (cylindrical) bubble of $R22$ gas. The shock speed is $\sigma = 415m.s^{-1}$. This test aims at simulating the experiment of [9] and has been considered by several authors [15, 18, 14]. The initial conditions are depicted in Figure 3: a bubble of $R22$ is surrounded by air within a $L_x \times L_y$ rectangular domain. At $t = 0$, the bubble is at rest and its center is located at (X_1, Y_1). We denote by r the initial radius of the bubble. The planar shock is initially located at $x = L_s$ and moves from right to left towards the bubble. The parameters for this test are

$$L_x = 445\text{mm}, L_y = 89\text{mm}, L_s = 275\text{mm}, X_1 = 225\text{mm}, Y_1 = 44.5\text{mm}, r = 25\text{mm}.$$

Both $R22$ and air are modeled by two perfect gases whose coefficients γ and initial states are given in the table of Figure 3.

The domain is discretized with a $20\,000 \times 5\,000$ regular mesh. Top and bottom boundary conditions are set to solid walls while we use constant state boundary conditions for the left and right boundaries. In Figure 4, we plot the density ρ at the final time $t_1 = 600\mu s$ on the domain $[0; 0.445] \times [0; 0.089]$. The computation needs 3 hours on the four AMD Radeon HD 7970 GPUs cluster. The shocks are well resolved in the air and in the bubble. We can localize



Quantities	Air (post-shock)	Air (pre-shock)	$R22$
ρ $(kg.m^{-3})$	1.686	1.225	3.863
u $(m.s^{-1})$	-113.5	0	0
v $(m.s^{-1})$	0	0	0
p (Pa)	1.59×10^5	1.01325×10^5	1.01325×10^5
φ	0	0	1
γ	1.4	1.4	1.249
p_∞	0	0	0

Figure 3: Air-$R22$ shock/cylinder interaction test. Description of the initial conditions on the left and initial data on the right.

the position of the shock wave that impinges the bubble on the left side of the domain. On the second figure we zoom on the bubble. On the third one we zoom on the Rayleigh-Taylor instabilities that appears at the bubble interface. With a coarser mesh, we could not see these instabilities. For other pictures or for applications to liquid-gas flows, we refer to [12, 10].

5 Conclusion

We have proposed a method for computing two-dimensional compressible flows with interface. Our approach is based on a robust relaxation Riemann solver, coupled with a very simple random choice sampling projection at the interface. The resulting scheme has properties that are not observed in other conservative schemes of the literature: it preserves velocity and pressure equilibrium at the two-fluid interface, it is conservative in mean, it does not diffuse the mass fraction φ, it preserves the non convex hyperbolic domain $\mathcal{H}$.

In addition, the algorithm is easy to parallelize on recent multicore architectures. We have implemented the scheme in the OpenCL environment. Compared to a standard CPU implementation, we observed that the GPU computations are more than hundred times faster. This factor is essentially due to the optimized transposition that we perform between x and y update and to the relaxation solver that gives a robust but simple expression of the numerical fluxes. As the computation is very fast, the limiting factor becomes the memory size of a GPU. The MPI version permits to treat very fine meshes. We test the code on R22/Air shock bubble interaction, thanks to a very fine mesh we can observe Rayleigh-Taylor instabilities at the bubble interface.

References

[1] D. Aubert and R. Teyssier. Reionization simulations powered by graphics processing units. i. on the structure of the ultraviolet radiation field. *The Astrophysical Journal*, 724:244–266, 2010.

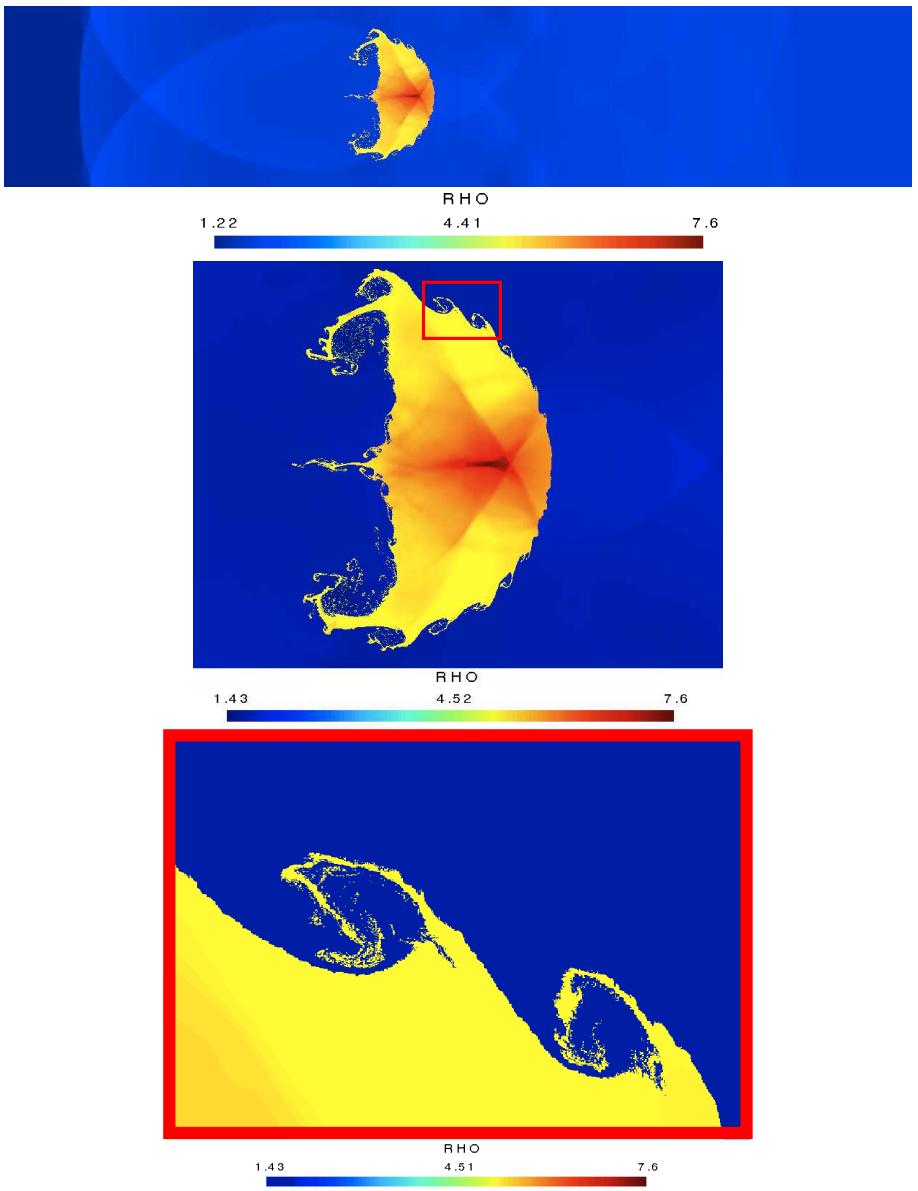


Figure 4: Density at final time $t_1 = 600\mu s$. On the first picture we represent all the domain, on the second one we do a zoom on the bubble and on the third we zoom on the Rayleigh-Taylor instability. On this picture, we can observe the precision of the computation.

[2] M. Bachmann, P. Helluy, J. Jung, H. Mathis, and S. MÃŒller. Random sampling remap for compressible two-phase flows. *Computers and Fluids*, 86:275–283, 2013.

[3] G.E. Blelloch. Scans as primitive parallel operations. *IEEE Transactions on Computers*, 38(11):1526–1538, 1989.

[4] F. Bouchut. *Nonlinear Stability of Finite Volume Methods for Hyperbolic Conservation Laws and Well-Balanced Schemes for Sources*. Frontiers in Mathematics series, Birkhauser, 2004.

[5] C. Chalons and F. Coquel. Computing material fronts with a lagrange-projection approach. *HYP2010 Proc.*, `http://hal.archives-ouvertes.fr/docs/00/54/89/38/PDF/chalons_coquel_hyp2010.pdf`, 2010.

[6] C. Chalons and P. Goatin. Godunov scheme and sampling technique for computing phase transitions in traffic flow modeling. *Interfaces and Free Boundaries*, 10(2):197–221, 2008.

[7] P. Colella. Glimm's method for gas dynamics. *SIAM, J. Sci. Stat. Comput.*, 3(1), 1982.

[8] Khronos Group. Opencl online documentation. `http://www.khronos.org/opencl/`.

[9] J. F. Haas and B Sturtevant. Interaction of weak shock waves with cylindrical ans spherical gas inhomogeneities. *J. Fluid Mechanics*, 181:41–71, 1987.

[10] P. Helluy and J. Jung. Opencl simulations of two-fluid compressible flows with a random choice method. *IJFV International Journal On Finite Volumes*, 10:1–38, 2013.

[11] D. A. Jacobsen, J.-C. Thibault, and I Senocak. An mpi-cuda implementation for massively parallel incompressible flow computations on multi-gpu clusters. *48th AIAA Aerospace Sciences Meeting and Exhibit*, 2010.

[12] J. Jung. *Schémas numériques adaptés aux accélérateurs multicoeurs pour les écoulements bifluides*. PhD thesis, University of Strasbourg, 2013.

[13] V. V. Kindratenko, J. Enos, M. T. Showerman, G. W. Arnold, J. E. Stone, J. C. Phillips, and W.-M. Hwu. Gpu clusters for high- performance computing. *Proc. Workshop on Parallel Programming on Accelerator Clusters, IEEE Cluster 2009*, pages 1–8, 2009.

[14] S. Kokh and F. LagoutiÃšre. An anti-diffusive numerical scheme for the simulation of interfaces between compressible fluids by means of the five-equation model. *J. Computational Physics*, 229:2773–2809, 2010.

[15] J. J. Quirk and S. Karni. On the dynamics of a shock-bubble interaction. *J. Fluid Mechanics*, 318:129–163, 1996.

[16] G. Ruetsch and P. Micikevicius. Optimizing matrix transpose in cuda. *NVIDIA GPU Computing SDK*, pages 1–24, 2009.

[17] R. Saurel and R. Abgrall. A simple method for compressible multifluid flows. *SIAM J. Sci. Comput.*, 21(3):1115–1145, 1999.

[18] K. M. Shyue. A wave-propagation based volume tracking method for compressible multicomponent flow in two space dimensions. *J. Comput. Phy.*, 215:219–244, 2006.