Simplified adaptive mesh refinement algorithm based on dual cell-boundary trees for multiphase compressible flows

Kevin Schmidmayer, Fabien Petitpas, Eric Daniel

Simplified Adaptive Mesh Refinement algorithm based on dual cell-boundary trees for multiphase compressible flows

Kevin Schmidmayer[a,*], Fabien Petitpas[a], Eric Daniel[a]

[a] *Aix Marseille Univ, CNRS, IUSTI, Marseille, France*

Abstract

An adaptive mesh refinement method is proposed for finite volume framework. The novelty of the method resides in using a dual data structure with two trees: a classical one for the computational cells and an extra one dedicated to computational cells boundaries. This new dual structure makes easier the method to implement as it simplifies algorithm. It results in an efficient adaptive mesh refinement method that improves computational efficiency while preserving an acceptable memory cost. This new AMR method is then applied on compressible multiphase flows in the framework of diffuse interface methods. Efficiency of the method is demonstrated thanks to computational results for different applications: transport, shock tube, capillary flows and shock/droplet interaction, in 1D, 2D and 3D. The test cases are performed with quantitative comparisons regarding non-AMR methods to analyze benefits.

Keywords: adaptive mesh refinement, diffuse interface, multiphase flow, shock wave

1. Introduction

In computational fluids dynamics, the accuracy of results is conditioned by the refinement level of the computational grid. Nevertheless, the finer is the grid, the more expensive is the computational cost regarding CPU time as well as memory. For the computations of steady flows, the use of unstructured grid and mesh refinement technics at well-defined locations can lead to very accurate results (see for example simulations around hydrofoils [12]). For the computations of strongly unsteady flows with shock waves or traveling interfaces, achievement of accurate results is conditioned by the use of a very small cell size. Thus, a large amount of computational time is wasted to compute solutions in cells where almost nothing occurs (example in Figure 1).

When dealing with multiphase flows, especially in the framework of diffuse interface methods [23], the previous remark is amplified by the complexity of the model:

*Corresponding author

Email addresses: kevin.schmidmayer@gmail.com (Kevin Schmidmayer), fabien.petitpas@univ-amu.fr (Fabien Petitpas), eric.daniel@univ-amu.fr (Eric Daniel)

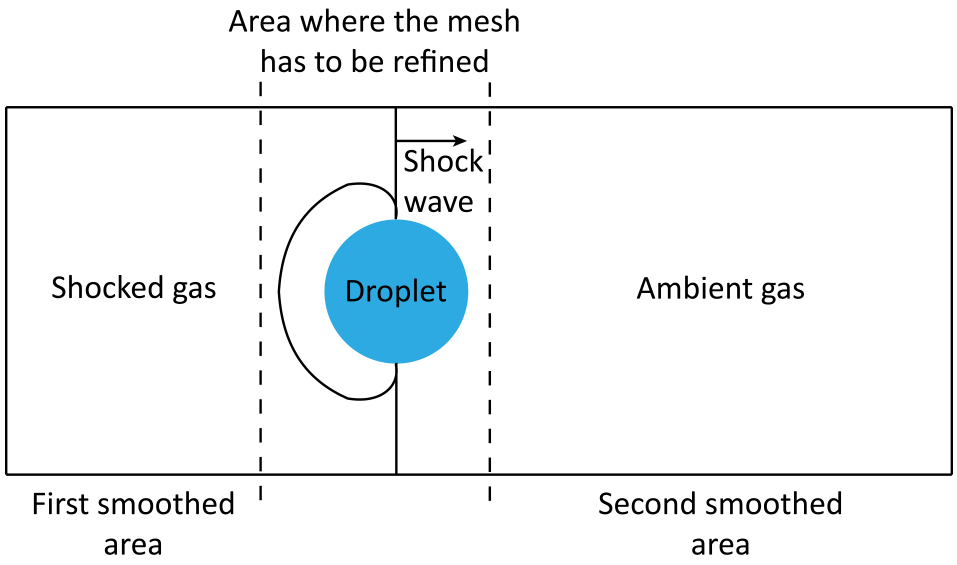

Figure 1: Sketch of a choc-droplet interaction where two parts are distinguished: one where the mesh has to be refined and one where it does not.

- an important number of evolution equations may be solved (Hank et al. [11], Petitpas et al. [20]),
- iterative solvers for relaxation procedure may also be required (Saurel et al. [25], Han et al. [10]),
- expensive Riemann solvers are sometimes needed to account for real material effects (Le Métayer et al. [14]).

On the strength of these observations, the use of Adaptive Mesh Refinement (AMR) technics represents an interesting option to reduce the CPU time cost when dealing with complex multiphase compressible flows. This is the aim of this study.

When AMR is embedded in a computational fluid dynamics code, the computational grid is dynamically adapted in order to be refined where it is necessary and to maintain a coarse grid elsewhere. These AMR methods are suitable and already massively used for many applications:

- magnetohydrodynamics (Anderson et al. [2], Dumbser et al. [7]),
- uncompressible multiphase flows as for the droplet motions in a microchannel and the atomization of liquid impinging jets (Chen and Yang [5], Popinet and Rickard [21]),
- compressible flows in porous media as for the leakage of gas from a liquefied petroleum gas storage cavern (Pau et al.[18]).

The analysis of literature shows that there are mainly two existing approaches in the context of finite volume scheme:

- The first one is an approach where cells are organized in Cartesian grids (Berger and Oliger [4], Berger and Colella [3]). The entire computational domain is represented by a coarse base grid and when more resolution is required, finer and nested grids are laid over coarser grids. This defines a grid hierarchy from the coarsest to the most refined grid that may be organized as a "grid tree". The grid generation or destruction occurs when the local relative error between calculations done on the current grid and a finer or a coarser grid, respectively, passes a given criterion. The main advantage of this approach is that the flow solver is independent of the grid, thus the same flow solver can be applied whatever the grid considered. That also means that any single-grid fluid flow solver can be used without important modifications for AMR. The drawbacks is due to the rigid structure of the grid. It induces that the simulation of complex flows requires the covering with an important number of grids and several grids of the same level of refinement may then overlap. A duplication of cells is inherit from this overlapping and also a substantial number of new computational cells can be wasted in smooth flow regions. Besides, the periodic rebuilding of the entire grid hierarchy is required when the flow evolves with time.

- In the second approach, the refinement occurs on an individual cell and this directly defines a tree of cells (cells tree method) (Young et al. [27]). Each cell can be refined or unrefined independently of others. Moreover, the mesh refinement occurs locally where it is necessary, then at every level of the tree, the mesh may have a non-uniform shape. The main advantage of the cells tree approach is the flexibility in refinement and unrefinement. This flexibility is paid by the fact that standard grid-based solvers cannot be used directly on a tree. Fluxes calculation and time stepping strategy on a tree is different from that on a grid, that means that the flow solver is slightly dependent on the level computed. In addition, an access to neighboring cells is more difficult in a tree than in an array and scanning the tree to access the nearest neighbors is a difficult procedure to vectorize and parallelize. Moreover, the memory cost oscillates between an additional or a lesser cost than the first approach because there are fewer cells to be stored but a tree generates an additional cost to be maintained.

  In Khokhlov [13], the use of a different structure of data allowed to circumvent these two last problems. The structure of Khokhlov involves a fully threaded tree where cells have not only knowledge of their child cells but also of their parent cells and neighboring cells. This thread provides an efficient parallel access to information on a tree. Khokhlov also improves the memory cost for maintaining the tree by regrouping cells in a so-called "oct" structure. Besides, the refinement criterion is based on physical variations between neighboring cells. This dynamical refinement is generally ensured by tracking discontinuities (as shock waves or contact discontinuities). The refinement evolves with the flow features which makes it more suitable for resolution of highly dynamical flows.

In this work, we retain the second approach for its ability to easily adapt the mesh for unsteady flows. Thus, a modified version of Khokhlov's method is presented. It is important to mention that the original method was developed in the late nineties and the goal was to compute single phase compressible flows using AMR method with the best performance concerning the efficiency and the memory. Even if the development of an efficient algorithm is still a key problem nowadays, the constraints on memory are not as crucial as they were. Today's problems are linked to multiphysics complex flows (including several phases, solids, phase transition, chemical reactions, viscosity, capillarity...) that are well described by complex mathematical models for which the computational cost represents the limitation, much more than memory. That is why two specificities of Khokhlov method have to be pointed out:

- The first one is that, the fully threaded tree involves a lot a operations to find the neighboring cells. This becomes a critical point when the algorithm is used to solve complex model coupled with high-order method where the procedure to find the neighbors occurs many times (flux calculations at each step of the high-order method, gradient calculations for numerical purpose or physical description, etc.).

- The second one is that if the AMR method is implemented in a code where the mesh can be either Cartesian or not, finding the neighbors could involve many complex operations.

The new method we propose has two advantages: the general AMR algorithm is first simple and also cell neighboring searching is improved. The key point is the role plays by cells boundaries (geometrical contour): obviously for the flux calculations and also because these boundaries naturally define neighbors between two cells. This new method extends Khovhlov approach on two points: the cell tree structure is slightly modified and a second tree is used to store some cell boundaries information. Addition of this second tree boundaries structure presents the advantages to reduce the number of operations during the time step integration and to simplify the algorithm. The drawback is to reasonably increase the memory involved. Reasonably because the number of additional information stored for each cell boundary is relatively small in comparison to what is needed in a cell.

The paper is then organized as follows: First, the extended AMR data structure is described. Second, the general AMR algorithm in the context of finite volume scheme (coupled with Riemann solvers for fluxes calculations) is presented. The time-stepping strategy, the advancing and the mesh refinement procedures are detailed. Third, the extension of the AMR method to the multiphase flow model of Schmidmayer et al. [26] is presented. Its application on different tests - transport, shock tube, capillary flow and water droplet atomization in 1D, 2D and 3D - is performed with quantitative comparisons regarding exact solutions or non-AMR method results in order to analyze the benefit of this new method.

2. Description of AMR data structure

Let us first recall the data structure of AMR method based on cells trees. A computational cell is represented by a node at a given level in a tree. Each node of a tree is linked thanks to edges to:

- A parent node representing a computational cell at lower level. The root of a tree is a particular node with no parent.
- A given number of child nodes representing computational cells at higher level. The number of child nodes depends on the geometry and the dimension of the problem. A leaf of a tree is a particular node with no child and the calculation of physical quantities (not linked to AMR) only occur on a node that is a leaf.

Each cell in AMR method may be split in a given number of child cells. An example of possible splitting in 1D/2D/3D Cartesian grid is shown on the right part of Figure 2 and an example of a tree representing data for a 1D AMR method is shown in on its left part.

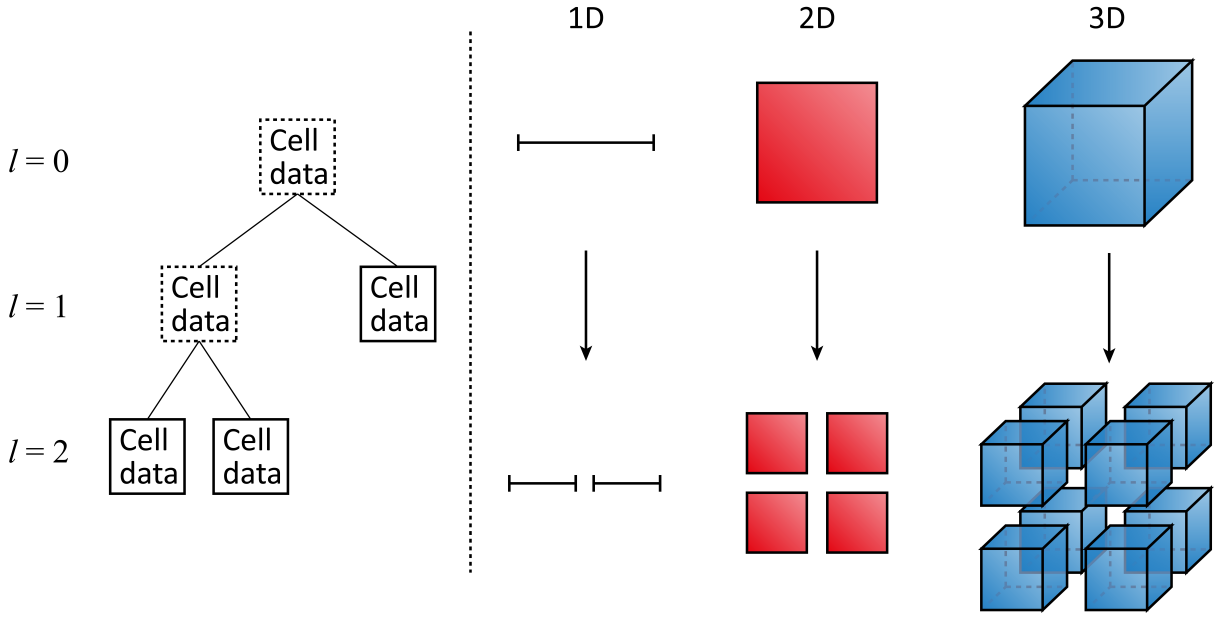

Figure 2: Figure split in two parts. On the left, an example of a tree representing the data structure for a 1D AMR method. "l" accounts for the level in the tree. On the right, possible cell splitting. A 1D, 2D and 3D cell will give birth in a Cartesian grid to 2, 4 and 8 child cells, respectively.

2.1. Recall of Khokhlov method [13]: FTT

The tree structure is the most obvious structure to define and optimize the cells data storage in an AMR method. Its flexibility allows refinement and unrefinement *via* destruction and reconstruction of chosen nodes in the tree. The cornerstone of such method resides in the chosen way to browse cells in the tree that can be a complex and expensive operation

depending on the links between cells. In its simplest version, a node can only be accessed from browsing the tree from its root. One can easily understand that the simple operation that consists in locating neighbors of a given cell (for example to calculate inter-cell fluxes) rapidly becomes a source of computational waste. An alternative solution is the use of a so called Fully Threaded Tree (FTT) where each nodes has the knowledge of its parent node, child nodes and neighbors nodes. This improvement renders easy the browsing of the tree in every direction (from parent to child, child to parent and even between neighbors). This ability to browse the tree in all directions has nevertheless the drawback to increase considerably the amount of memory (by addition of multiple pointers acting as threads). It also renders maintenance operations on the tree more complicated when refinement and unrefinement occur.

Khokhlov proposes to group cells into octs in order to limit extra storage due to links between nodes and to limit maintenance operations costs. Each oct contains 8 organized cells (in 3D), a pointer to its parent cell at lower level and 6 pointers to parents cells of neighboring octs. Each cell contains physical flow variables and a pointer to a child oct at higher level. Most of the pointers, as well as the geometrical properties (level, position, size, etc.), are thus grouped to be stored into octs rather than in cells. Consequently, the memory costs is significantly reduced (especially in 3D) in comparison to a FTT simple version without oct structure.

The memory cost saving is undeniable when dealing with Euler equations, it is more disputable when dealing with complex models traducing multiphysics problems. In such applications of AMR, the ratio between physical flow variables and geometrical/AMR variables increases drastically. Khokhlov explains that the cost of his oct-FTT AMR version is 2 words of memory per cell instead of 17 words/cell for a non oct version of FTT. These additional memory costs have to be compared with physical flow variables that are necessary to store (5 words/cell for Euler equations but 6N words/cell for a general N-phase flow without extra physics).

In addition to the ratio of memory cost saving that decreases when the model, and then the physic, becomes complicated, another drawback of oct-trees is that the computational time associated with computing cell pointers from oct pointers and oct pointers from cell pointers, or in other words extra time needed to look for neighbors cells, is estimated to about 20% when computing single phase flow with a first order scheme. Here again, when complex models coupled with high-order numerical solvers are of interest (involves neighborhood seeking for each flux calculation of the high order scheme, for each gradient calculation, etc.), this extra computational time involved by the structure may be no longer negligible.

2.2. Basic idea of the new AMR data structure: The extra cells boundaries trees

An efficient way to avoid increasing CPU time and difficulty in searching neighbors in parallel computations is to take benefit of information related to cells boundaries. In the finite volume framework, a cell boundary may be defined as a geometrical contour between two computational cells that is the seat of fluxes calculations. A boundary may be defined as an object that stores two pointers, one for a “left” cell and another one for a “right” cell. Availability of such objects prevents from neighbors seeking when solving inter-cell fluxes in

a computational CFD code. In a more general context of unstructured grids, it also prevents from using a connectivity table. It implies that finite volume algorithms using such data structure can be easily used whatever the grid structure is.

In addition to cells trees, we thus propose to define boundaries trees. In these boundaries trees, cells boundaries are represented by nodes that are linked to other boundary nodes by edges. The duality of cells tree and boundaries trees represents a complex data structure that greatly simplifies the algorithm and reduces computational costs. Up to this remark, the new data structure is composed of:

- cells that are organized in tree structures (oct tree or not). They may also be linked to boundaries.
- boundaries that are also organized in trees structures. They can also be grouped in quad tree to mimic Khokhlov's oct cell trees structures (a boundary will be split in up to 4 child boundaries in 3D).

Boundaries trees in the AMR method implies additional memory costs. Nevertheless, with this new data structure, calculations at boundaries (fluxes, gradients, etc.) are naturally accessed without seeking for neighbors. This important point renders the method easily extensible to unstructured meshes. Moreover, the oct structure used in Khokhlov's work can be kept to regroup information regarding geometrical properties in the Cartesian framework.

2.3. Detailed description of trees

For convenience in presentation, data structure is presented for non-oct trees. The alert reader will easily extend the method to oct-trees if needed.

The main tree of the method is quite similar to those of a FTT classical method. The computational cells constitute the nodes of the cells tree. In particular they contain the physical flow properties (depending on the flow model under interest) as well as geometrical data. These cells nodes also includes additional data specific to the AMR method:

* an integer for its level (0 for the root, > 0 otherwise),
* a pointer for each of its child cells nodes (up to 8 in Cartesian 3D),
* an additional pointer for each of its boundary (up to 6 in Cartesian 3D),
* a pointer for root of each new internal boundary tree (up to 12 in Cartesian 3D). The particular case of the internal boundaries is presented in the following.

Compared to a classical FTT method, the novelty resides in the pointers to boundaries that represent an additive memory cost of maximum 11 pointers/cell in 3D (12 new pointers for internal boundaries, 1 less because pointer to parent cell is no longer needed in the method). Up to this point, a given cell can be either split or not (if its pointers to child cells nodes are null).

The second novel data structure is represented by new boundaries trees. A cell boundary is a new object that includes two cell pointers (one for the left cell and one for the right

cell). The interest of the presence of such objects in a finite volume method resides in a better access to fluxes calculation between two computational cells. In the present AMR method, cells boundaries constitute nodes of new boundaries trees. These boundaries nodes then includes additional data specific to the AMR method:

* an integer for its level,

* a pointer to each of its child boundaries nodes (up to 4 in 3D).

This new data structure possesses some particular specificity. Indeed, let us consider the example of a 2D Cartesian cell represented in Figure 3. This cell is surrounded by 4 boundaries (blue edges). Refinement of this cell will give birth to 4 new computational cells and 12 new boundaries. Among these 12 boundaries, 8 of them are originated from parent cell's boundaries splitting (dashed red edges) and appear naturally as children of parent boundaries. Also, 4 new boundaries appear inside the parent cell as the result of cell's splitting and are considered as root of new boundaries trees (dashed point green edges). Consequently, splitting of a given cell will act on boundaries trees in two ways:

- It will increase the depth of already existing trees. "External" boundaries of the parent cell, that were leaves before splitting, will become parent of new boundaries (up to 4 in 3D), the last ones are thus leaves.

- In the same time, it will also generate new "internal" boundaries that are roots (and leaves) of new boundaries trees.

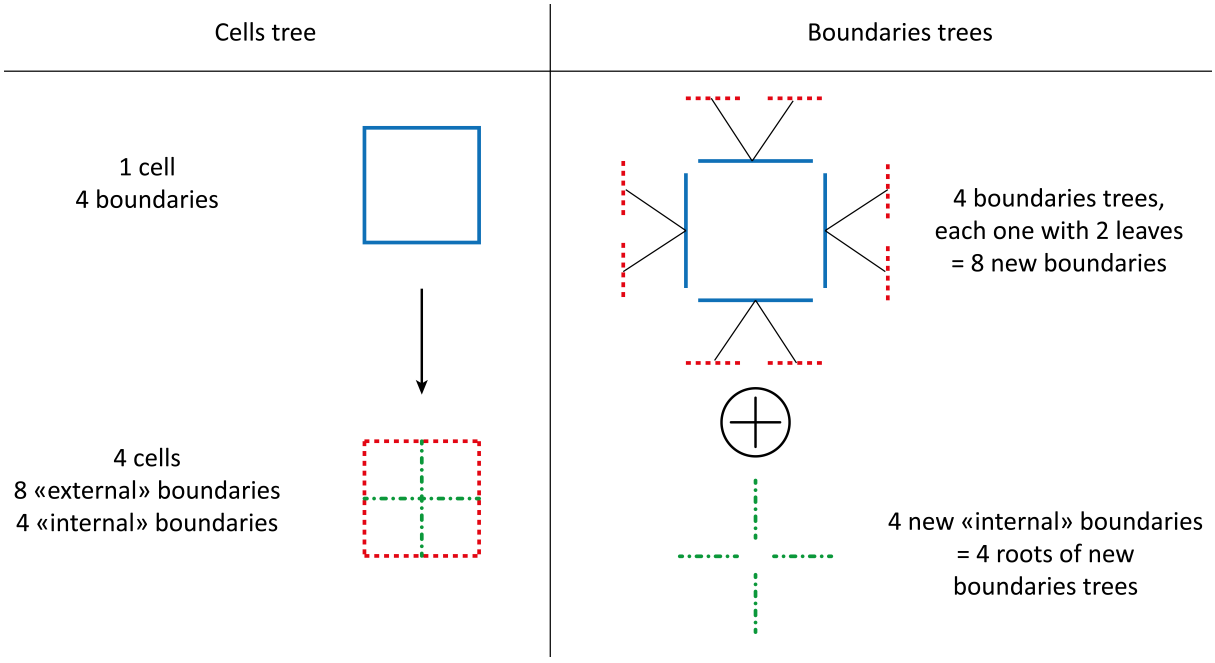

Figure 3: 2D example of cells tree and boundaries trees duality.

A 2D example of the links between cells and boundaries trees is illustrated in Figure 4. In this example, 3 levels are present. The figure is decomposed in three parts:

- The top part shows 2 successive refinements occurring from a given level-0 cell. The following cells and boundaries trees correspond to this particular splitting.
- The middle part shows the corresponding cells tree represented by square nodes and composed with 4 level-1 cells and 4 level-2 cells.
- The bottom part shows the corresponding boundaries trees represented by circles nodes. 4 level-0 boundaries trees, 12 level-1 boundaries (including 4 new level-1 boundaries trees) and 12 level-2 boundaries (including 4 new level-2 boundaries trees) are generated.

The adopted numeration is 'XYZ' with X being C (for cell) or B (for boundary), Y corresponds to level number (here from 0 to 2) and Z is the letter corresponding to the entity (A to N in the present case). Pointers between cells in cell tree are shown with black lines as well as pointers between boundaries in boundaries trees. In order to facilitate comprehension, the pointers between cells and boundaries are non exhaustively presented but only some typical examples:

- In yellow lines, cell C0A will point to the 4 boundaries B0A, B0B, B0C and B0D of level 0. Such pointers are present for each cell and are needed for gradient calculations.
- In dashed point green lines, are represented pointers from a boundary of level 1 (B1N) to two level-1 cells (C1C and C1D).
- In point blue lines, an example of a boundary (B2J) linked to two cells from different levels (C1C and C2D).
- The last example in dashed red lines shows a boundary (B1A) linked to a level-1 cell (C1A) and another cell neighbor (level-0) of cell C0A or one of her child (level-1 cell) not shown in the figure.

This last 3 kind of links are used for fluxes calculation (only if the boundary is a leaf).

One can note that the number of boundaries may be important. Nevertheless, the boundaries trees reasonably increase the memory involved since they only need a few additional pointers for each boundary. Comparison with a classical FTT structure of the required memory is presented in Table 1. The table shows the detailed number of words for each cell and each boundary. The total number of words reported to a cell is also given. For a given 3D hexahedron cell, the new method requires 11 additional words. Noticing that a boundary is common to 2 neighboring cells, a cell requires approximately 3 boundaries (instead of 6), each of them requiring 7 words. The global over-cost of the new method is thus 32 words/cell. This has to be compared to the memory cost for storage of geometrical and physical variables that may represent the larger part of memory costs in a multiphase and multiphysic computation.

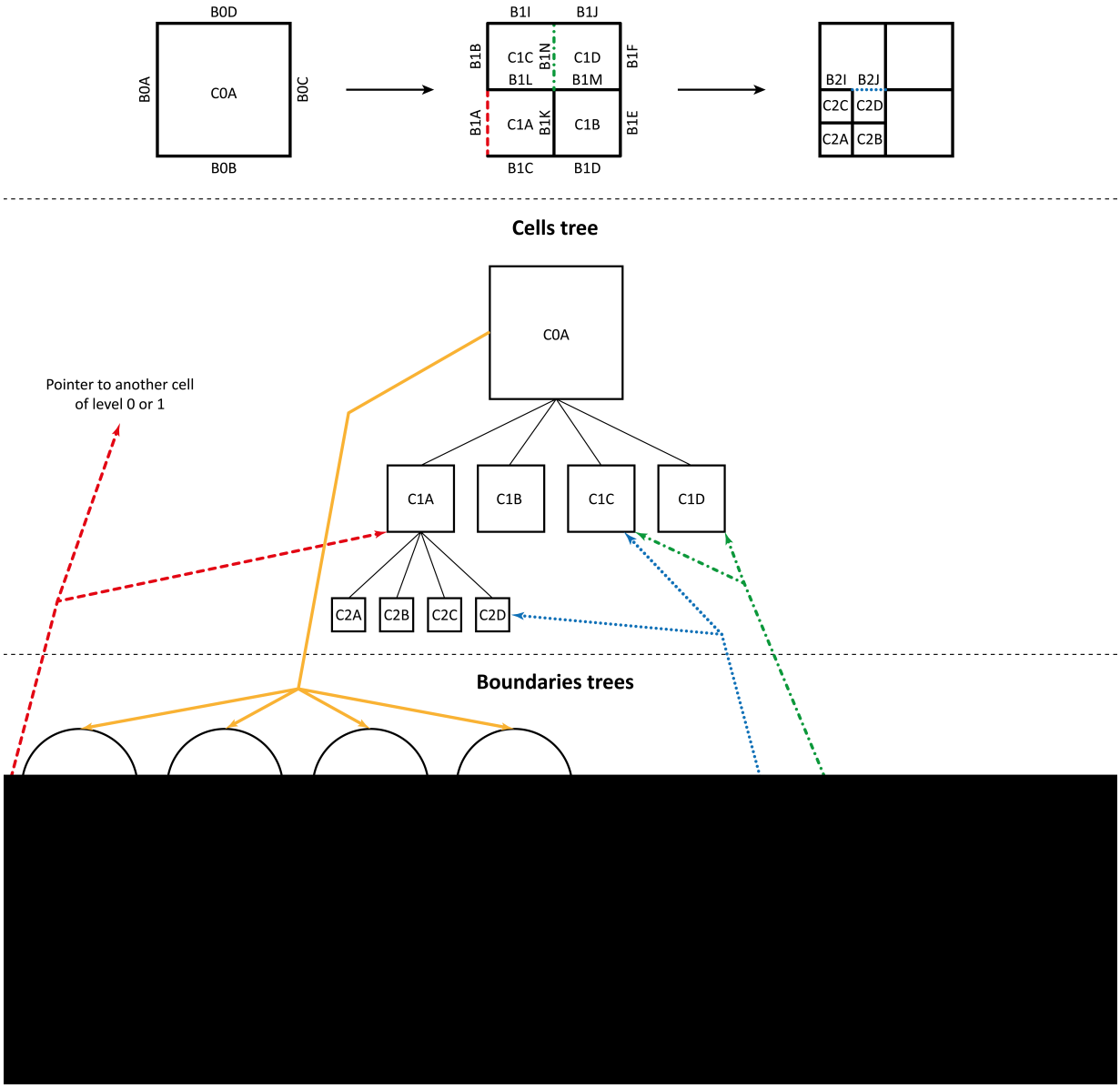

Figure 4: 2D example of links between cells and boundaries trees. Here, some details are given for a cell and its 2 levels of refinement. The top represents the cells appearing from the two successive refinement form a level-0 parent cell. The middle and bottom sketches are representing the cell tree and boundaries trees, respectively. Connections between both cells and boundaries trees are presented in some typical situations.

Saving may be done regarding memory costs by using Khokhlov-like oct tree method for particular Cartesian grids. Indeed, it is possible to group cells in oct ([13]) as well as boundaries, where grouping is also possible: "external" boundaries are grouped in quad and "internal" boundaries are grouped in dodeca. This improvement in term of memory cost is possible but complicates the AMR algorithm. We present in Table 2 these possible savings in memory costs and the global over-cost of the new method is thus reduced to 3.75 words/cell.

	Data type	Classical FTT AMR	New AMR
Words number / Cell data	Cell level	1	1
	Refinement indicator	1	1
	Pointer to parent cell	1	-
	Pointers to child cells	8	8
	Pointers to neighboring cells	6	-
	Pointers to cell boundaries	-	6
	Pointers to internal cell boundaries	-	12
Words number / Cell boundary (x3)	Boundary level	-	1
	Pointers to children boundaries	-	4
	Total number of words / cell	17	43

Table 1: Memory costs comparison between classical AMR FTT method and new AMR method using boundaries trees. For a given 3D hexahedron cell, the new method requires 11 additional words. Noticing that a boundary is common to 2 neighboring cells, a cell requires approximately 3 boundaries (instead of 6), each of them requiring 7 words. The global over-cost of the new method is thus 32 words/cell.

	Data type	Khokhlov AMR	New AMR
Words number / Cell data	Cell level	1/8	1/8
	Refinement indicator	1	1
	Pointer to parent cell	1/8	-
	Pointers to child oct	1	1
	Pointers to neighboring octs	6/8	-
	Pointers to oct's quad boundaries	-	6/8
	Pointers to oct's internal dodeca boundaries	-	1/8
Words number / Cell boundary (x3)	Boundary level	-	1/4
	Pointers to children boundaries	-	1
	Total number of words / cell	3	6.75

Table 2: Possible savings using oct tree for cells and quad/dodeca trees for boundaries.

However, as mentioned in 2.1, these AMR structure memory savings are balanced by the memory needed for physical quantities as well as extra quantities stored for computational conveniences. For these reasons we decide to highlight algorithm simplicity and computational efficiency and thus not retain the oct tree structure.

3. General AMR algorithm

3.1. Finite volume scheme for conservation laws

We consider a system of conservation laws under the following form:

$$\frac{\partial \mathbf{U}}{\partial t} + \boldsymbol{\nabla} \cdot \overline{\overline{F}}\left(\mathbf{U}\right) = \mathbf{S} \tag{1}$$

with $\mathbf{U}$ the conservative variable vector, $\overline{\overline{F}}$ the fluxes tensor and $\mathbf{S}$ the source terms vector. Integration of system 1 on a computational cell of volume V_i delimited by surfaces A of

normal unit vector $\mathbf{n}$ (a two-dimensional example is presented in Figure 5) reads:

$$\frac{\partial}{\partial t}\int_{V_i}\mathbf{U}\mathrm{d}V+\int_{A}\overline{\overline{F}}\left(\mathbf{U}\right)\cdot\mathbf{n}\mathrm{d}A=\int_{V_i}\mathbf{S}\mathrm{d}V, \tag{2}$$

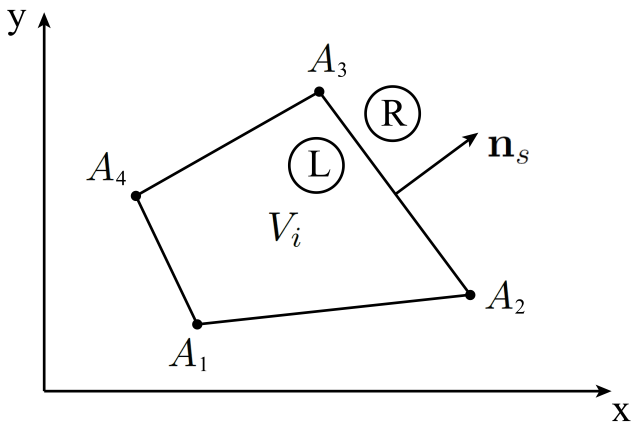

Figure 5: Scheme example for a 2D computational cell

The first and last terms of Eq. (2) are interpreted as the time-rate of change of the conservative variable and source terms vectors volume average:

$$\frac{\partial}{\partial t}\int_{V_i}\mathbf{U}\mathrm{d}V=V_i\frac{\partial\overline{\mathbf{U}}}{\partial t},$$

$$\int_{V_i}\mathbf{S}\mathrm{d}V=V_i\overline{\mathbf{S}}.$$

As boundary A of V_i is the union of N straight segments $[A_s, A_{s+1}]$, where $A_{N+1}=A_1$ and the normal unit vector is expressed by $\mathbf{n}_s$, the second term of (2) becomes:

$$\int_{A}\overline{\overline{F}}\cdot\mathbf{n}\mathrm{d}A=\sum_{s=1}^{N}\int_{A_s}^{A_{s+1}}\overline{\overline{F}}\cdot\mathbf{n}_s\mathrm{d}A,$$

Assuming that the fluxes are constant along each segment, it becomes:

$$\int_{A}\overline{\overline{F}}\cdot\mathbf{n}\mathrm{d}A=\sum_{s=1}^{N}L_s\overline{\overline{F}}_s\cdot\mathbf{n}_s,$$

where L_s is the length of segment $[A_s, A_{s+1}]$ (a surface in 3D).

After time integration, the evolution of the conservative part of system (2) is given for cell i by the scheme:

$$\mathbf{U}_i^{n+1}=\mathbf{U}_i^n-\frac{\Delta t}{V_i}\sum_{s=1}^{N}L_s\overline{\overline{F}}_s^{\star}\cdot\mathbf{n}_s+\Delta t\mathbf{S}_i, \tag{3}$$

where $\overline{\overline{F}}_s^{\star}$ represents the fluxes tensor solution of the Riemann problem between left (L) and right (R) states separated by the segment $[A_s, A_{s+1}]$ with respect to normal $\mathbf{n}_s$.

3.2. Time-stepping strategy

The efficiency of an AMR method necessitates the implementation of a specific time-stepping strategy. Following the works of Khokhlov [13], this time-stepping strategy is based on two key points:

- Cells at different levels evolve with different time-steps according to their level of refinement. In order to maintain the global time-step coherence for unsteady simulations, if cells of level l evolve at a given time-step, cells of level $l+1$ will then evolve 2 times with a time-step 2 times smaller. It results in CPU time saving.
- This time-stepping strategy permits interleaving between time integration and tree refinement. It results in memory save as it limit excessive buffer layer of refinement ahead of a discontinuity [13].

The global time-step is determined using the minimum tree level where there are leaf cells (l_{min}) and the CFL condition:

$$\Delta t = \Delta t\left(l_{min}\right) = cfl \frac{L}{2^{l_{min}} \max\left(\left|\left(\mathrm{u}+\mathrm{a}\right)_{\mathrm{s}}^{*}\right|\right)}$$

where $cfl < 1$ is a constant, L is the characteristic length of the coarser cells (at level $l = 0$) and the maximum speed is determined going through each leaf cell boundary where u is the fluid velocity in the corresponding face direction and a is the sound waves speed. Time-steps at various levels are:

$$\Delta t\left(l\right) = 2^{l_{min}-l} \Delta t.$$

The general integration procedure occurs at the different levels of the tree as an interleaving of advances and refinements. It is expressed as a recursive procedure $I(l_{min})$ with:

$$\begin{array}{lcll} I\left(l_{min}\right) & = & A\left(l_{min}\right) I\left(l_{min}+1\right) R\left(l_{min}\right) & \\ I\left(l\right) & = & A\left(l\right) I\left(l+1\right) A\left(l\right) I\left(l+1\right) R\left(l\right), & \text{for} \quad l \neq l_{min}, l_{max} \\ I\left(l_{max}\right) & = & A\left(l_{max}\right) A\left(l_{max}\right) R\left(l_{max}\right) & \end{array} \tag{4}$$

where $A(l)$ represent the advancement procedure of level l described in Section 3.3 and $R(l)$ is the refinement/unrefinement procedure of level l detailed in Section 3.4. All procedures in (4) are performed from right to left, *i.e.*, $R\left(l\right)$ first, and $A\left(l\right)$ last. An example of the sequence generated by (4) could be, for $l_{min} = 0$ and $l_{max} = 2$:

$$[R\left(0\right) \quad [R\left(1\right) \quad [R\left(2\right) A\left(2\right) A\left(2\right)] \quad A\left(1\right) \quad [R\left(2\right) A\left(2\right) A\left(2\right)] \quad A\left(1\right)] \quad A\left(0\right)].$$

One can note that the generality of the new method in the present finite volume framework simplifies the recursive integration procedure in comparaison to [13] where directional time-step splitting is computed. Indeed, the procedure of the lower level l_{min} is simplified, *i.e.*, only one advancing procedure is undertaken, and each advancing procedure is identical (only the referred level changes). The last one has to be compared with [13] where two

advancing procedures have to be undertaken, one for the sequence of XYZ one-dimensional sweeps and one for its reversed.

Because of the recursive evolution algorithm, trees browsing will be traduce by an important amount of test to detect cells and boundaries levels. A possibility to avoid going through all the trees and then accelerate the procedures is to add lists for cells and boundaries and for each level of the simulation. Then loops on cells or boundaries become straightforward and constantly efficient.

3.3. Advancing procedure

The advancing procedure A is called at each time-step ($\Delta t\,(l)$) to advance the solution at the time $t + \Delta t\,(l)$ using the numerical scheme (3). This advancement procedure is decomposed in 3 steps:

- The first step is solving the hyperbolic part of System 3. A loop is first performed on leaves boundaries of level l where fluxes are estimated (using Riemann solvers) and stack in a buffer flux variable (initially set to 0) in each of "left" (L subscript) and "right" (R subscript) neighboring cells. This fluxes buffer is denoted by $\widetilde{\mathbf{F}}$. It is important to notice that possible contribution to these buffers comes from higher boundaries level (during preceding advancement procedures at higher levels).
  At the end of this boundary loop, buffers for cells of level l are complete. Indeed whatever the neighbors level are, fluxes have been stacked either in these advancement procedure or in those of higher levels. Then, conservative variable for leaves cells of level l should be evolved and corresponding buffers fluxes reset to 0 for next time step.

- The second step consists in upgrading leaves cells of level l using source terms integration.

- The third step is an averaging procedure consisting in updating split cells of level l. This step is useful for the correct computation of the refinement procedure (presented in details in Section 3.4).

The $A\,(l)$ procedure is described in a form of the following pseudocode:

1. — Hyperbolic resolution —

   for (leaf boundaries s of level l) {
   Compute the hyperbolic fluxes tensor $\overline{\overline{F}}^{\star}_{s} = \overline{\overline{F}}^{\star}_{s}\,(\mathbf{U}^n_L, \mathbf{U}^n_R)$;
   $\widetilde{\mathbf{F}}_L = \widetilde{\mathbf{F}}_L - l_{diff,L} L_s \overline{\overline{F}}^{\star}_{s} \cdot \mathbf{n}_s$;
   $\widetilde{\mathbf{F}}_R = \widetilde{\mathbf{F}}_R + l_{diff,R} L_s \overline{\overline{F}}^{\star}_{s} \cdot \mathbf{n}_s$;
   }
   for (leaf cells i of level l) {
   $\mathbf{U}^1_i = \mathbf{U}^n_i + \frac{\Delta t}{V_i} \widetilde{\mathbf{F}}_i$;
   $\widetilde{\mathbf{F}}_i = \mathbf{0}$;
   }

2. — Source terms resolution —

   for (leaf cells i of level l) {
   $\mathbf{U}_i^{n+1} = \mathbf{U}_i^1 + \Delta t \mathbf{S}_i(\mathbf{U}_i^1)$;
   }

3. — Averaging of the children for the parent cells —

   for (parent cells i of level l) {
   for (child cells j of parent cell i) {
   $\widetilde{\mathbf{F}}_i = \widetilde{\mathbf{F}}_i + \mathbf{U}_j^{n+1}$;
   }
   $\mathbf{U}_i^{n+1} = \widetilde{\mathbf{F}}_i /$ Number of child cells;
   $\widetilde{\mathbf{F}}_i = \mathbf{0}$;
   }

The l_{diff} factor appearing in fluxes buffer stacking takes into account potential level differences between "left" and "right" cells on the considered boundary. It would take the value $l_{diff} = 1$ if the two neighboring cells have the same level or if the flux is applied to the cell with the higher level, and the value $l_{diff} = 0.5$ if the flux is applied to the cell with the lower level. In the example of cells C2D and C1B in Figure 4, the one with the higher level (C2D) will have 2 time-step integrations while the one with the lower level (C1B) will have just 1. In that case, for the fluxes calculation between these two cells, $l_{diff,L} = 1$ for cell C2D and $l_{diff,L} = 0.5$ for cell C1B. In that way, it makes a time average flux in the cell with the lower level.

Extension of this advancement procedure will be done in Section 4.4 for non conservative system of multiphase flows.

3.4. Mesh refinement procedure

If the data structure and integration algorithm represents key points to ensure AMR simulations efficiency, the ability to refine or unrefine at required locations is an other key point obviously linked to the quality of numerical results. This is also the most problem-dependent part and the choice for the refinement criteria is undeniably the most difficult point for the user of an AMR method. This point will be discussed in the results part. The cells refinement is always linked to a refinement indicator $0 \leqslant \xi \leqslant 1$ which is computed and stored for every computational cell. This indicator will be used to detect which are the cells that needs to be refined or unrefined:

- If a leaf cell has $\xi \geqslant \xi_{split}$, it indicates that the corresponding cell must be refined.
- If a split cell has $\xi < \xi_{join}$, the corresponding cell can be unrefined.

where ξ_{split} and ξ_{join} are two predefined constant parameters controlling the cell refinement dynamics forward and backward a discontinuity. We also impose an extra condition to control refinement: two neighboring cells cannot possess more than one level difference.

3.4.1. ξ indicator setup

The approach we use to calculate the refinement ξ indicator is based on locations of significant gradients (Coirier [6], Aftosmis et al. [1], Melton et al. [16]) and it acts in two steps:

- For each computational cell, ξ is calculated by:

$$\begin{aligned} \xi &= 1 \quad \text{if}: \quad \frac{|(\mathrm{X})_{\mathrm{Nb(i,j)}} - (\mathrm{X})_{\mathrm{i}}|}{\min\left((\mathrm{X})_{\mathrm{Nb(i,j)}}, (\mathrm{X})_{\mathrm{i}}\right)} > \epsilon, \\ \xi &= 0 \quad \textit{otherwise} \end{aligned} \tag{5}$$

  where X can be pertinent physical variable (for example p, $\|\mathbf{u}\|$, ρ, α). $Nb(i,j)$ represents neighboring cells (j accounts for a corresponding neighbor of cell i). The choice of the variable will discriminate shocks, contact discontinuities, interfaces or any kind of gradients. ϵ is a constant parameter that controls the limit in term of stiffness of the detected gradients. Attention should be paid with velocity to avoid division by zero. A combination of several gradients may also been used to improve detection.

- The second step consists in smoothing the refinement indicator. This operation is very important for several reasons. First it prevents cells from being falsely refined (mesh trashing). Secondly, smoothing will allow cells forward a discontinuity to be refined before the discontinuity arrival and by this way prevent oscillations as well as a loss of precision. To perform smoothing, we consider the ξ indicator obeys to a diffusion equation:

$$\frac{\delta\xi}{\delta\tilde{t}} = K\nabla^2\xi, \tag{6}$$

  where $\tilde{t}$ is a fictive diffusion time only used to advance the solution for the diffusion of ξ into the domain. So, this diffusion has no link or impact on the treated physical characteristic. $K = 2^{-2l}L^2$ is a constant diffusion coefficient. This equation is solved with an explicit time advancement where the time step is chosen to preserve the diffusion stability ($dt = CFL_{diff}K/2$, where CFL_{diff} corresponds to the CFL condition pf the diffusion equation). Note that when this equation is solved, the number of fictive time iterations indirectly gives the number of cells where the indicator will be diffused. Typically, 3 or 4 time iterations are enough.

The splitting and joining criteria (explained previously) are then used to determine if the cell has to be refine or unrefine. The refinement around the contact discontinuity is presented in Figure 6 as a typical example.

3.4.2. Refinement and unrefinement of cells and boundaries

Due to the dual data structure, refinement (unrefinement) acts in two steps: first the refinement (unrefinement) of the cells and second of the boundaries.

Once the ξ indicator of every cell of the current level l is smoothed, the refinement of a cell occurs if $\xi \geqslant \xi_{split}$ and then the two steps are:

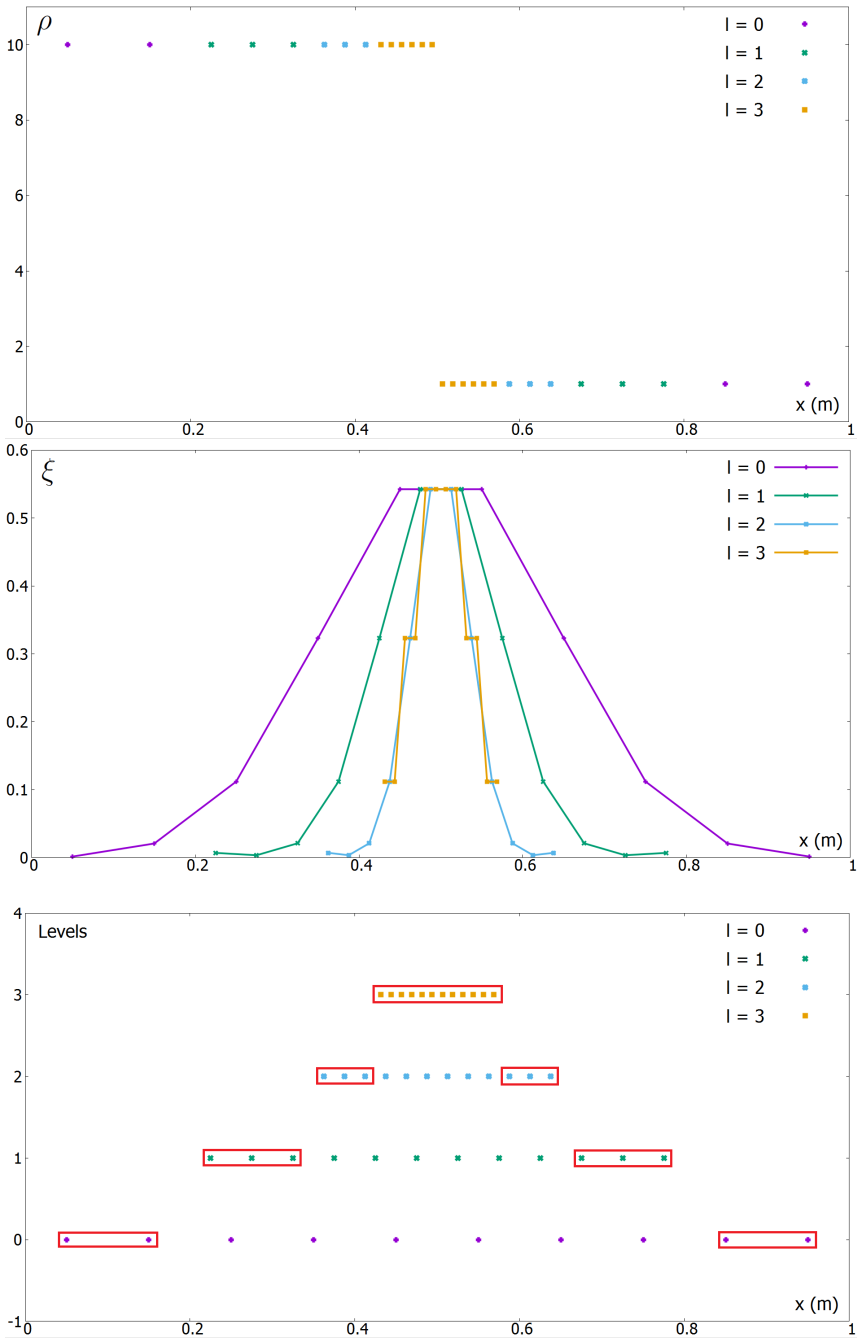

Figure 6: Result of the successive mesh refinement procedures around a density discontinuity. The top plot shows the density discontinuity, the center plot shows the values of the indicator for each level and the bottom plot shows the cells levels distribution (the red surrounded leaf cells are the cells where the integration occurs).

- First, the cell refinement. It does not involve special difficulties as it follows the scheme of Figure 2. A refined cell will give birth to up to 8 child cells (in Cartesian 3D) of level $l+1$ and each child will be built with the same physical characteristics than its parent cell.

- Second, the boundaries refinement. It is also performed in two steps for each splitting cell. The first one is the creation of the internal child boundaries of level $l+1$ that belong to the parent cell and then creates new boundaries trees (up to 12 in Cartesian 3D). In the second step, the creation of the external child boundaries of level $l+1$ for each of the boundaries of the parent cell (level l) is executed only if it was not already done by the corresponding neighboring cell. The child boundaries belong to their parent boundary (see Figure 3).

For the unrefinement, it occurs if $\xi < \xi_{join}$ and then the corresponding two steps are:

- First, the cell unrefinement. The physical characteristics of the child cells are conservatively averaged to overwrite the ones of the parent cell (see point 3 of Section 3.3). Then, child cells are removed.

- Second, the boundaries unrefinement. Again it is performed in two steps for each joining cell. The first one is the removal of the internal child boundaries that belong to the parent cell. In the second step, the removal of the external child boundaries is done only if the corresponding neighboring cell is not split.

All the pointers are obviously redirected to the corresponding cell or boundary if necessary.

3.4.3. Mesh refinement procedure pseudocode

The $R(l)$ refinement procedure is thus described as the following pseudocode:

1. — ξ setup —

```
for (cells i of level l) {
    ξ = 0;
    if (one of the gradient criteria is respected) { ξ = 1; }
}
```

2. — Smoothing of ξ —

```
for (x diffusion iterations) {
    for (cells i of level l) {
        Compute the diffusion equation for ξ (Eq. (6));
    }
}
```

3. — Refinement —

if ($l < l_{max}$) {
  for (non-split cells i of level l) {
    if ($\xi \geqslant \xi_{split}$ & level of each neighboring cells $> l-1$) { Cell i is refined; }
  }
}

4. — Unrefinement —

if ($l < l_{max}$) {
  for (split cells i of level l) {
    if ($\xi < \xi_{join}$ & level of each neighboring cells $\leqslant l+1$ & children cells non-split) {
      Cell i is unrefined with children averaging (see point 3 of Section 3.3) to overwrite the values of cell i;
    }
  }
}

4. Extension to multiphase flow model of Schmidmayer et al. [26]

The AMR method presented in this paper is devoted to applications to multiphase compressible flows and particularly to diffuse interface models. The model retained is the one presented in Schmidmayer et al.[26] for capillary flows. However, the presented AMR method can be easily adapted to treat extra physics as for example phase transition (Massoni et al. [15], Saurel et al. [24], cavitation (Petitpas et al. [19]), detonation in high energetic materials (Petitpas et al. [20]), solid-fluid interaction and compaction of granular media (Favrie and Gavrilyuk [9, 8]) and low Mach number flows (Murrone and Guillard [17]).

We recall here the main properties of the Schmidmayer et al. model [26] and a brief overview of the basic numerical scheme in the context of non AMR methods.

4.1. Multiphase system of equations

The pressure relaxation model with capillary effects of [26] is:

$$\begin{cases} \frac{\partial \alpha_1}{\partial t} + \mathbf{u}\cdot\nabla\alpha_1 & = \mu\left(P_1 - P_2\right), \\ \frac{\partial \alpha_1\rho_1}{\partial t} + \boldsymbol{\nabla}\cdot\left(\alpha_1\rho_1\mathbf{u}\right) & = 0, \\ \frac{\partial \alpha_2\rho_2}{\partial t} + \boldsymbol{\nabla}\cdot\left(\alpha_2\rho_2\mathbf{u}\right) & = 0, \\ \frac{\partial \rho\mathbf{u}}{\partial t} + \boldsymbol{\nabla}\cdot\left(\rho\mathbf{u}\otimes\mathbf{u} + P\overline{\overline{I}} + \overline{\overline{\Omega}}\right) & = \mathbf{0}, \\ \frac{\partial \alpha_1\rho_1 e_1}{\partial t} + \boldsymbol{\nabla}\cdot\left(\alpha_1\rho_1 e_1\mathbf{u}\right) + \alpha_1 P_1\boldsymbol{\nabla}\cdot\mathbf{u} & = -\mu P_I\left(P_1 - P_2\right), \\ \frac{\partial \alpha_2\rho_2 e_2}{\partial t} + \boldsymbol{\nabla}\cdot\left(\alpha_2\rho_2 e_2\mathbf{u}\right) + \alpha_2 P_2\boldsymbol{\nabla}\cdot\mathbf{u} & = \mu P_I\left(P_1 - P_2\right), \\ \frac{\partial c}{\partial t} + \mathbf{u}\cdot\boldsymbol{\nabla}c & = 0, \end{cases} \tag{7}$$

where α_k, ρ_k, e_k and P_k are the volume fraction, the density, the internal energy and the pressure of phase k, and each fluid is governed by its own equation of state (EOS) $e_k = e_k\left(\rho_k, P_k\right)$.

ρ, P and $\mathbf{u}$ are the mixture variables for density, pressure and velocity. Concerning the capillary effects terms, σ is the surface tension coefficient, c is a color function and $\overline{\overline{\Omega}}$ is the capillary tensor given by:

$$\overline{\overline{\Omega}} = -\sigma \left(\|\boldsymbol{\nabla} c\| \overline{\overline{I}} - \frac{\boldsymbol{\nabla} c \otimes \boldsymbol{\nabla} c}{\|\boldsymbol{\nabla} c\|} \right).$$

μ is the pressure relaxation coefficient, $P_I = \frac{Z_2 P_1 + Z_1 P_2}{Z_1 + Z_2}$ (see [22] for details) and $Z_k = \rho_k a_k$ is the acoustic impedance of the phase k with a_k being the speed of sound of the corresponding phase. The mixture pressure is given by:

$$P = \alpha_1 P_1 + \alpha_2 P_2.$$

Due to the condition $P_1 \neq P_2$ in this model, the total energy equation of the mixture is replaced by the internal energy equation for each phase. Nevertheless, the mixture total energy equation of the system can be written in usual form:

$$\frac{\partial \rho E + \varepsilon_\sigma}{\partial t} + \boldsymbol{\nabla} \cdot \left((\rho E + \varepsilon_\sigma + P)\, \mathbf{u} + \overline{\overline{\Omega}} \cdot \mathbf{u} \right) = 0. \tag{8}$$

where $E = e + \frac{1}{2}\|\mathbf{u}\|^2$ and e are the mixture variables for total energy and internal energy. And the capillary energy is equal to $\varepsilon_\sigma = \sigma \|\boldsymbol{\nabla} c\|$.

The equation (8) is redundant when both phasic internal energy equations are solved, but it will appear to be an important ingredient for numerical method to ensure the energy conservation and to preserve a correct treatment of shock waves.

One can note that the surface tension effects are missing in the phasic energy equations since it is only a mixture characteristic.

A special splitting procedure will be done for the numerical resolution of model (7).

4.2. Splitting procedure

Model (7) without the relaxation terms is split in two submodels. The first submodel does not take into account the surface tension terms and the second one contains the only capillary terms. The submodels are presented below only in the x-direction.

Hyperbolic submodel 1

The first submodel is similar to that presented in [25] with additional decoupled equations for the gradient of the color function:

$$\begin{cases} \dfrac{\partial \alpha_1}{\partial t} + u\dfrac{\partial \alpha_1}{\partial x} & = 0, \\ \dfrac{\partial \alpha_1\rho_1}{\partial t} + \dfrac{\partial \alpha_1\rho_1 u}{\partial x} & = 0, \\ \dfrac{\partial \alpha_2\rho_2}{\partial t} + \dfrac{\partial \alpha_2\rho_2 u}{\partial x} & = 0, \\ \dfrac{\partial \rho u}{\partial t} + \dfrac{\partial \rho u^2 + \alpha_1 P_1 + \alpha_2 P_2}{\partial x} & = 0, \\ \dfrac{\partial \rho v}{\partial t} + \dfrac{\partial \rho uv}{\partial x} & = 0, \\ \dfrac{\partial \rho w}{\partial t} + \dfrac{\partial \rho uw}{\partial x} & = 0, \\ \dfrac{\partial \alpha_1\rho_1 e_1}{\partial t} + \dfrac{\partial \alpha_1\rho_1 e_1 u}{\partial x} + \alpha_1 P_1 \dfrac{\partial u}{\partial x} & = 0, \\ \dfrac{\partial \alpha_2\rho_2 e_2}{\partial t} + \dfrac{\partial \alpha_2\rho_2 e_2 u}{\partial x} + \alpha_2 P_2 \dfrac{\partial u}{\partial x} & = 0, \\ \dfrac{\partial w_1}{\partial t} + \dfrac{\partial w_1 u}{\partial x} & = 0, \\ \dfrac{\partial w_2}{\partial t} + u\dfrac{\partial w_2}{\partial x} & = 0, \\ \dfrac{\partial w_3}{\partial t} + u\dfrac{\partial w_3}{\partial x} & = 0, \end{cases} \tag{9}$$

where w_1, w_2 and w_3 are the components in the three directions of the gradient $\mathbf{w}$ of the color function c.

This system describes only the transport and the compression waves. The equation for w_1 is taken in conservative form to let the possibility to consider weak solutions. The other terms in this equation will be treated in the second submodel.

The eigenvalues of the system are:

$$\lambda_{1,2,3,4,5,6,7,8,9} = u,$$

$$\lambda_{10} = u - a_f,$$

$$\lambda_{11} = u + a_f,$$

where a_f is the frozen mixture sound speed:

$$a_f^2 = Y_1 a_1^2 + Y_2 a_2^2.$$

The hyperbolicity of this first submodel is proven in [25].

Weakly hyperbolic submodel 2

The second submodel is:

$$\begin{cases} \frac{\partial \alpha_1}{\partial t} & = 0, \\ \frac{\partial \alpha_1 \rho_1}{\partial t} & = 0, \\ \frac{\partial \alpha_2 \rho_2}{\partial t} & = 0, \\ \frac{\partial \rho u}{\partial t} + \left(\frac{\partial \Omega_{11}}{\partial w_1}\frac{\partial w_1}{\partial x} + \frac{\partial \Omega_{11}}{\partial w_2}\frac{\partial w_2}{\partial x} + \frac{\partial \Omega_{11}}{\partial w_3}\frac{\partial w_3}{\partial x} \right) & = 0, \\ \frac{\partial \rho v}{\partial t} + \left(\frac{\partial \Omega_{12}}{\partial w_1}\frac{\partial w_1}{\partial x} + \frac{\partial \Omega_{12}}{\partial w_2}\frac{\partial w_2}{\partial x} + \frac{\partial \Omega_{12}}{\partial w_3}\frac{\partial w_3}{\partial x} \right) & = 0, \\ \frac{\partial \rho w}{\partial t} + \left(\frac{\partial \Omega_{13}}{\partial w_1}\frac{\partial w_1}{\partial x} + \frac{\partial \Omega_{13}}{\partial w_2}\frac{\partial w_2}{\partial x} + \frac{\partial \Omega_{13}}{\partial w_3}\frac{\partial w_3}{\partial x} \right) & = 0, \\ \frac{\partial \alpha_1 \rho_1 e_1}{\partial t} & = 0, \\ \frac{\partial \alpha_2 \rho_2 e_2}{\partial t} & = 0, \\ \frac{\partial w_1}{\partial t} + w_2 \frac{\partial v}{\partial x} + w_3 \frac{\partial w}{\partial x} & = 0, \\ \frac{\partial w_2}{\partial t} & = 0, \\ \frac{\partial w_3}{\partial t} & = 0. \end{cases} \tag{10}$$

This second system describes the capillary effects. Also, the non-conservative product in the equation for w_1 is well defined because w_2 and w_3 are continuous through the shock.

4.3. Numerical Method without AMR

The numerical method is presented as a 3-step method. Each step is successively performed in order to circumvent specific numerical problems:

- First, the hyperbolic non-equilibrium pressure model (9) is solved using a Godunov-type method [26].
- Second, model (10) is solved. A specific attention is paid to the choice for the flux terms in order to ensure the momentum and energy conservation.
- Third, a relaxation procedure leads to the pressure equilibrium.

The unknown vector $\mathbf{U}^{n+1}$ is obtained from the initial condition $\mathbf{U}^n$ by application of the three successive operators according to the sequence:

$$\mathbf{U}^{n+1} = L_{relax} L_{cap} L_{hyper} \left(\mathbf{U}^n \right),$$

where the vector $\mathbf{U}$ contains the unknown quantities defined in the system:

$$\mathbf{U} = \left[\alpha_1, \alpha_1\rho_1, \alpha_2\rho_2, \rho u, \rho v, \rho w, \alpha_1\rho_1 e_1, \alpha_2\rho_2 e_2, c, \rho E + \varepsilon_\sigma \right]^T$$

Each step of the numerical method is fully detailed in Schmidmayer et al. [26]. It is important to note that, for the different capillary terms, the vector $\mathbf{w}$ is computed *via* derivatives of the color function which are computed by using second-order finite difference approximations. Plus, to go through the operators chain, the solution at time $n+1$ is obtained by a pressure relaxation algorithm and corrects the components of $\mathbf{U}^{cap}$:

$$\mathbf{U}^{n+1} = L_{relax} \left(\mathbf{U}^{cap} \right).$$

The details about the pressure relaxation algorithm as well as the correction procedure used to guarantee total energy conservation can be found in Saurel *et al.* [25].

4.4. Extension of the AMR algorithm for multiphase flow

The extension of the AMR algorithm to treat multiphase flow model (7) implies several modification to the method:

1. Model (7) is non-conservative. It is thus necessary to take into account for non conservative terms in the advancement procedure,
2. The global time-step of integration is slightly modified to add a cell gradient procedure G that computes the color function gradients required in the capillary effects formulation. It is done before going to the higher tree level and integration procedure now reads:

$$\begin{array}{lcll} I\left(l_{min}\right) & = & A\left(l_{min}\right) I\left(l_{min}+1\right) G\left(l_{min}\right) R\left(l_{min}\right), & \\ I\left(l\right) & = & A\left(l\right) I\left(l+1\right) A\left(l\right) I\left(l+1\right) G\left(l\right) R\left(l\right), & for \quad l \neq l_{min}, l_{max}, \\ I\left(l_{max}\right) & = & A\left(l_{max}\right) A\left(l_{max}\right) R\left(l_{max}\right) & \end{array} \tag{11}$$

   The same precedent example thus implies now a a sequence generated by (11) for $l_{min} = 0$ and $l_{max} = 2$ gives:

$$\begin{array}{cccc} [R\left(0\right) G\left(0\right) & [R\left(1\right) G\left(1\right) & [R\left(2\right) A\left(2\right) A\left(2\right)] & \\ A\left(1\right) & [R\left(2\right) A\left(2\right) A\left(2\right)] & A\left(1\right)] & A\left(0\right)]. \end{array}$$

3. Model (7) also contains relaxation terms that implies modification in the algorithm.

In the following part, the modification of advancing procedure as well as the cell gradient and relaxation procedures specific to multiphase capillary flows are detailed.

4.4.1. Cell gradient procedure

The cell gradient procedure G is the procedure to compute the different cell gradients which could be needed to treat a specific physic, here the capillary effects. Indeed, when the computation of the capillary effects is done, the fluxes calculation on a boundary uses the color function cell gradient of each neighboring cell of this boundary. And to avoid the cell gradient calculation multiple times for each cell, it is computed in a loop going through each cell, and not when the calculation of the fluxes are involved. Thus, in the case where one of the two neighboring cell of a boundary has a smaller level than this boundary, the cell gradient in this cell has to be computed before doing the fluxes calculation. But, in the recursive integration procedure (Equation (11)), if the cell gradient procedure $G\left(l_{min}\right)$ is not done before going to the integration procedure of the higher level $I\left(l+1\right)$, the needed cell gradient for the fluxes calculation of the advancing procedure of this higher level $A\left(l+1\right)$ would not have been computed. It thus explains why this procedure is necessary. So, the procedure is done at the level l before going to the recursive integration procedure $I\left(l+1\right)$ and it follows the pseudocode:

```
if (l < l_max) {
  for (leaf cells i of level l) {
      Compute the color function gradients G_i(U_i^n);
  }
}
```

One should note that using cell gradients involves a particularity when coupling with AMR method. Indeed, the cell gradients calculation is always done at a particular time in a given cell and thus it does not take into account that neighboring cells with other levels could be at other times due to the recursive integration procedure. Nevertheless, this particularity should not be problematic because it happens in zones where the gradients are small. An example could be the computation of a droplet: the color function gradients in cells are significant at the interface position and when the refinement criteria are well chosen, the zones at and around the interface are fully refined and then this particularity does not appear.

4.4.2. Advancing procedure

For the resolution of model (7), the advancing procedure have to take into account three additional points: one is related to the additional physics (here the capillary effects), another for the relaxation step and a last one for non conservative terms. Source terms are absent in the model, thus the corresponding point is avoided. If there were some, they would have been added in the following pseudocode between the point 2 and 3, and under the same formulation than in the pseudocode of Section 3.3. Consequently, the pseudocode of the $A(l)$ procedure is:

1. — Hyperbolic resolution —

   for (leaf boundaries s of level l) {
   Compute the hyperbolic fluxes tensor $\overline{\overline{F}}_s^\star = \overline{\overline{F}}_s^\star(\mathbf{U}_L^n, \mathbf{U}_R^n)$ and its corresponding contact discontinuity velocity $\mathbf{u}_s^\star$;
   $\widetilde{\mathbf{F}}_L = \widetilde{\mathbf{F}}_L - l_{diff,L} L_s \left(\overline{\overline{F}}_s^\star + \mathbf{H}_{nc,s}(\mathbf{U}_L^n)\, \mathbf{u}_s^\star \right) \cdot \mathbf{n}_s$;
   $\widetilde{\mathbf{F}}_R = \widetilde{\mathbf{F}}_R + l_{diff,R} L_s \left(\overline{\overline{F}}_s^\star + \mathbf{H}_{nc,s}(\mathbf{U}_R^n)\, \mathbf{u}_s^\star \right) \cdot \mathbf{n}_s$;
   }
   for (leaf cells i of level l) {
   $\mathbf{U}_i^1 = \mathbf{U}_i^n + \frac{\Delta t}{V_i} \widetilde{\mathbf{F}}_i$;
   $\widetilde{\mathbf{F}}_i = \mathbf{0}$;
   }

2. — Capillary effects resolution —

   for (leaf cells i of level l) {
   Compute the color function gradients $\mathbf{G}_i = \mathbf{G}_i(\mathbf{U}_i^1)$;
   }
   for (leaf boundaries s of level l) {
   Compute the capillary fluxes tensor $\overline{\overline{F}}_s^{cap}(\mathbf{U}_L^1, \mathbf{U}_R^1, \mathbf{G}_L, \mathbf{G}_R)$;
   $\widetilde{\mathbf{F}}_L = \widetilde{\mathbf{F}}_L - l_{diff,L} L_s \overline{\overline{F}}_s^{cap} \cdot \mathbf{n}_s$;
   $\widetilde{\mathbf{F}}_R = \widetilde{\mathbf{F}}_R + l_{diff,R} L_s \overline{\overline{F}}_s^{cap} \cdot \mathbf{n}_s$;
   }

for (leaf cells i of level l) {
$\mathbf{U}_i^2 = \mathbf{U}_i^1 + \frac{\Delta t}{V_i}\widetilde{\mathbf{F}}_i$;
$\widetilde{\mathbf{F}}_i = \mathbf{0}$;
}

3. — Relaxations resolution —

for (leaf cells i of level l) {
Compute the relaxation procedure to obtain $\mathbf{U}_i^{n+1}$ from $\mathbf{U}_i^2$;
}

4. — Averaging of the children for the parent cells —

for (parent cells i of level l) {
for (child cells j of parent cell i) {
$\widetilde{\mathbf{F}}_i = \widetilde{\mathbf{F}}_i + \mathbf{U}_j^{n+1}$;
}
$\widetilde{\mathbf{F}}_i = \widetilde{\mathbf{F}}_i /$ Number of child cells;
Compute the relaxations procedure to obtain $\mathbf{U}_i^{n+1}$ from $\widetilde{\mathbf{F}}_i$;
$\widetilde{\mathbf{F}}_i = \mathbf{0}$;
}

where the hyperbolic fluxes tensor $\overline{\overline{F}}^{\star} = \left(\mathbf{F}_x^{\star}, \mathbf{F}_y^{\star}, \mathbf{F}_z^{\star}\right)$, the non-conservative vector $\mathbf{H}_{nc}$ and the capillary fluxes tensor $\overline{\overline{F}}^{cap} = \left(\mathbf{F}_x^{cap}, \mathbf{F}_y^{cap}, \mathbf{F}_z^{cap}\right)$ are given in a Cartesian expression by:

$$\mathbf{F}_x^{\star}(\mathbf{U}) = \begin{bmatrix} \alpha_1 u \\ \alpha_1\rho_1 u \\ \alpha_2\rho_2 u \\ \rho u^2 + P \\ \rho uv \\ \rho uw \\ \alpha_1\rho_1 e_1 u \\ \alpha_2\rho_2 e_2 u \\ cu \\ (\rho E + P)\, u \end{bmatrix} \qquad \mathbf{F}_y^{\star}(\mathbf{U}) = \begin{bmatrix} \alpha_1 v \\ \alpha_1\rho_1 v \\ \alpha_2\rho_2 v \\ \rho uv \\ \rho v^2 + P \\ \rho vw \\ \alpha_1\rho_1 e_1 v \\ \alpha_2\rho_2 e_2 v \\ cv \\ (\rho E + P)\, v \end{bmatrix} \qquad \mathbf{F}_z^{\star}(\mathbf{U}) = \begin{bmatrix} \alpha_1 w \\ \alpha_1\rho_1 w \\ \alpha_2\rho_2 w \\ \rho uw \\ \rho vw \\ \rho w^2 + P \\ \alpha_1\rho_1 e_1 w \\ \alpha_2\rho_2 e_2 w \\ cw \\ (\rho E + P)\, w \end{bmatrix}$$

$$\mathbf{F}_x^{cap}(\mathbf{U}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \Omega_{11} \\ \Omega_{12} \\ \Omega_{13} \\ 0 \\ 0 \\ 0 \\ \varepsilon_\sigma u + \Omega_{11} u + \Omega_{12} v + \Omega_{13} w \end{bmatrix} \quad \mathbf{F}_y^{cap}(\mathbf{U}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \Omega_{21} \\ \Omega_{22} \\ \Omega_{23} \\ 0 \\ 0 \\ 0 \\ \varepsilon_\sigma u + \Omega_{21} u + \Omega_{22} v + \Omega_{23} w \end{bmatrix}$$

$$\mathbf{F}_z^{cap}(\mathbf{U}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \Omega_{31} \\ \Omega_{32} \\ \Omega_{33} \\ 0 \\ 0 \\ 0 \\ \varepsilon_\sigma u + \Omega_{31} u + \Omega_{32} v + \Omega_{33} w \end{bmatrix}$$

$$\mathbf{H}_{nc} = \begin{bmatrix} -\alpha_1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \alpha_1 P_1 \\ \alpha_2 P_2 \\ -c \\ 0 \end{bmatrix}$$

4.4.3. Relaxation procedure

The relaxation procedure is present in two locations of the multiphase AMR algorithm, i.e., at the end of each time step (reference to the relaxations resolution in point 3 of the advancing procedure Section 4.4.2) and in the unrefinement procedure (see next paragraph). This relaxation procedure is fully detailed in [25] and its aim is to take into account the mechanical equilibrium *via* pressure relaxation and its respective impact on the volume fraction and density of each phase. If pressure relaxation is activated, a mechanical equilibrium model is solved using the pressure desequilibrium model (7) (see [26] for details).

When cells are joined (during a refinement) an operation of averaging has to be performed before removing children cells (see point 4 of Section 3.4.3). When dealing with a multiphase model as model 7, a supplementary relaxation procedure has to be also added. The point 4 of Section 3.4.3 is then replaced by the pseudocode:

```
if (l < l_max) {
  for (split cells i of level l) {
      if (ξ < ξ_join & level of each neighboring cells ⩽ l + 1 & children cells non-split) {
        Cell i is unrefined with children averaging and relaxations procedure to
        overwrite the values of cell i;
      }
  }
}
```

5. Numerical results

The ability of the new AMR method to solve different multiphase flow applications is proven in 1D, 2D and 3D configrations. Comparisons with theoretical results and non-AMR method are also shown. The results are obtained thanks to an high-order scheme with a MUSCL-Hancock procedure and using the Harten-Lax-van Leer Contact (HLLC) approximate Riemann solver.

In each presented cases, the equation of state (EOS) for the air obeys to the ideal gas law:

$$P_{air} = (\gamma_{air} - 1)\,\rho_{air} e_{air},$$

with $\gamma_{air} = 1.4$.

The water obeys the stiffened gas EOS:

$$P_{water} = (\gamma_{water} - 1)\,\rho_{water} e_{water} - \gamma_{water} P_{\infty,water},$$

where the stiffened gas EOS parameters are $\gamma_{water} = 4.4$ and $P_{\infty,water} = 6.10^8 Pa$.

5.1. 1D transport test

The goal of the first test is to show the influence of the different criteria of refinement in comparison to a fully refined non-AMR mesh on a 1D transport test case. To avoid any potential complex interaction with a multiphase model, this first test is done for only one phase and then the Euler model is computed. Note that for comparison equivalence, the size of the cells for the non-AMR mesh and for the cells at the highest level (l_{max}) using the AMR method are identical.

Thus, a simple transport of a different density along the axis is done. The fluid is air with a density discontinuity of $\rho_{discontinuity} = 10 kg.m^{-3}$ in a density environment of $\rho_{environment} = 1 kg.m^{-3}$. The velocity of the whole domain is set to $u = 50 m.s^{-1}$, the pressure is uniform and the Neumann boundary condition is used. At the initialization the center of the density discontinuity is set at the coordinate $0.3m$ and has a length of $0.2m$.

The simulation time is $t = 8ms$. Only the density is shown since all the other variables remain uniform.

Concerning the AMR method, 4 refinement levels ($l_{max} = 4$) are involved which means that there are 5 levels in total with the initial one. The mesh is initialized with $N = 10$ cells and then the corresponding number of cells with a the full refinement is $N\text{x}2^{l_{max}} = 160$. The choice of the refinement criteria is one of the most difficult part in an AMR method and it is completely case-dependent. However, the gradient refinement criterion in this first test is obvious and is based on density variation, thus the rest of the different criteria (ϵ, ξ_{split} and ξ_{join}) are tested to observe their impact on the results. The information concerning the AMR data is given in Table 3 for 4 different test cases.

Test case	Initial number of cells	l_{max}	Equivalent mesh	ϵ	ξ_{split}	ξ_{join}	Max number of cells involved
AMR 1	10	4	160	0.1	0.5	0.5	50
AMR 2	10	4	160	0.1	0.5	0.1	61
AMR 3	10	4	160	0.1	0.1	0.1	73
AMR 4	10	4	160	1	0.1	0.1	56

Table 3: AMR data for the 1D transport test case using the Euler model.

Figure 7 shows the results comparison with the different criteria of Table 3. The initialization using AMR method is shown in blue and it is identical for each test as the different criteria only has an influence during the evolution of the discontinuity transport. The non-AMR result is also shown in red but it is partially hidden by the AMR result of the third test (in purple) because this third AMR test gives as good result as the non-AMR one.

On the top image of Figure 7, results using three combinations of ξ_{split} and ξ_{join} are compared:

- First, one can observe the difference concerning the shape of the results for the test AMR 1 (yellow), where the previously cited criteria are taken equal to $\xi_{split} = \xi_{join} = 0.5$, and AMR 2 (black), where they have different values $\xi_{split} = 0.5$ and $\xi_{join} = 0.1$. The result with a lower ξ_{join} is closer to the non-AMR one at the head of this heaviside function but at the rear, the results are similar between the two AMR tests and they have a lower density than the non-AMR result. In the two cases the matching with the non-AMR result is better at the head of the discontinuity. One can conclude that the refinement and unrefinement process gives better results in the upwind direction and that having the same value of the criteria yield to better results concerning the symmetrical aspect. In the following tests, to avoid a maximum this non-symmetrical aspect the values of those two criteria are always taken equal. One can also note that the maximum number of cells involved is higher in the second test case than in the first one (see Table 3) because the joining criteria is smaller in the second test and then the joining occurs less often.

- Second, in the case where the two criteria are taken with a lower value ($\xi_{split} = \xi_{join} = 0.1$), the result (AMR 3 test case in purple) is in better agreement with the one of the non-AMR method, not only at the head of the discontinuity but also at the rear. In fact, the values of these criteria indirectly give the number of refined cells around a detected discontinuity (detected through the gradient criterion limited value ϵ). More the values of ξ_{split} and ξ_{join} are small, more the number of refined cells around the discontinuity is important. Note that the number of fictive time iteration for the diffusion equation (6) of ξ also indirectly control the number of refined cells around the discontinuity. 4 iterations are involved in the presented tests.

The results on the bottom image of Figure 7 shows the importance of the gradient criterion limited value ϵ with the comparison between tests AMR 3 (purple) and 4 (green) where $\epsilon = 0.1$ and $\epsilon = 1$, respectively. Lower is the value of the criterion, closer to the non-AMR method is the result. However, as shown in Table 3 with the maximum number of cells involved, it is important to notice that this value has to be well chosen, not only to be close to the equivalent solution, but also not to be a value where all the mesh is refined, or in that case the AMR principal advantage is lost. Furthermore, this criterion compares the normalized variation of the chosen physical variable, here density, with the value of ϵ (Equation (5)). And because it is normalized with the minimum density of the cell where the calculation is done or of its neighboring cells, plus because the absolute numerical diffusion is the same on the two sides of the discontinuity, the normalized variation is higher on the side of the smaller density and then this side is more refined. Thus, the head of the heaviside discontinuity moves in a refined mesh containing more cells at the highest level than in its rear and it explains the non-symmetrical aspect that is clearly observable for high values of ϵ.

The variation of the initial number of cells and number of levels for a constant equivalent non-AMR mesh is not shown here because it yields to close results, even if the total number of cells involved are different. Then, having the lower initial number of cells with the higher number of refinement levels seems the best option since the results are similar but the computational time should be lower. "Should be" because of the balance between the computational time lost in the recursive integration and refinement procedures with a high number of levels and the gain between the computation of two different initial meshes. Moreover, using high order methods reduce the number of cells involved in the calculation due to the stiffer discontinuities. This last point partially counterbalances the additional computational time involved by these high order methods.

In the following, the impact of the criteria is no more shown but it is important to keep in mind that the choice of those ones is crucial to obtain as good results as expected and to not involved to much cells in the computation. Thus, a set of chosen criteria that is one good balanced possible solution between computational time involved and quality of the results is proposed for each test.

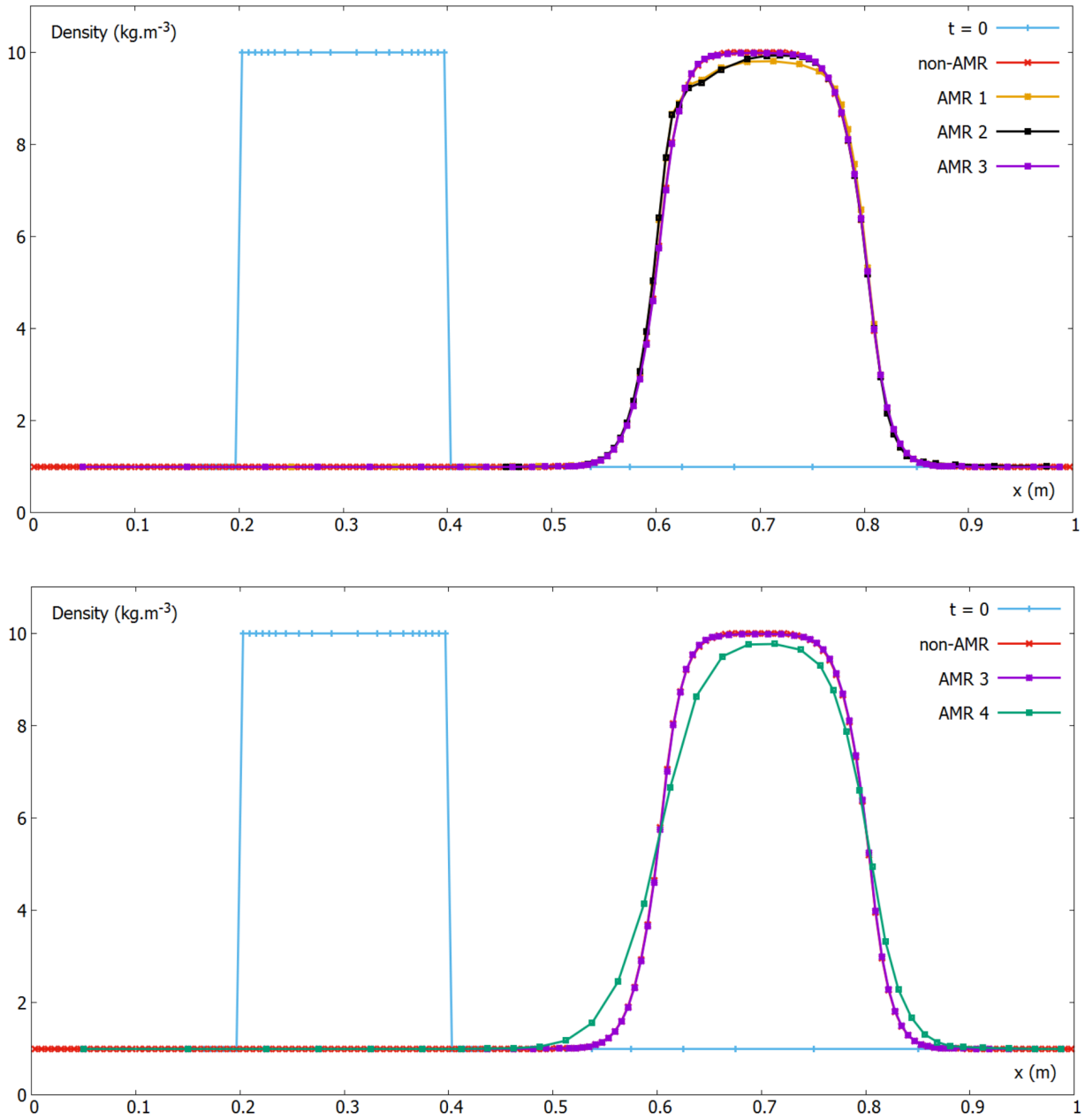

Figure 7: Density ρ along the x-direction for the 1D transport test case using the Euler model. Initialization setup (blue) is shown using the AMR method and results are given for different tests: non-AMR (red) and four AMR (yellow, black, purple and green) with different AMR criteria.

5.2. Liquid/gas shock tube

The shock tube test for multiphase flow is the flow creates by the initial contact of a high and a low pressurization chamber. The high pressure chamber is filled with water at the pressure $P = 1.10^9 Pa$ and with a density of $\rho = 1,000 kg.m^{-3}$. In the low pressure chamber, there is air with a pressure of $P = 1.10^5 Pa$ and a density of $\rho = 50 kg.m^{-3}$. In both chambers the fluid velocity is $u = 0 m.s^{-1}$. The length of the tube is $1m$ and the initial discontinuity is located at $0.7m$ with high pressure chamber on the left and low pressure one on the right. The simulation time is $t_{final} = 241 \mu s$.

As previously explained, there is no predefined obvious refinement criteria to use, it is case-dependent. Specially, which thermodynamic variables variations to look at to compute the gradient refinement criterion. In the shock tube test, because of the physics involved, the mixture density and mixture pressure variations are chosen. The other information concerning the AMR data is given in Table 4 where the equivalent mesh indicates the number of cells of a fully refined mesh and thus indicates the number of cells of the non-AMR method it is compared with.

Initial number of cells	l_{max}	Equivalent mesh	ϵ	ξ_{split}	ξ_{join}	Max number of cells involved
10	8	2,560	0.1	0.1	0.1	210

Table 4: AMR data for the 1D shock tube test case.

Figure 9 shows the exact solution (black line) and the simulation solutions for the non-AMR mesh (top image, red line) and for the AMR mesh (bottom image, blue crosses and line). A shock is propagated from the left to the right, following by the contact discontinuity and the expansion waves are propagated in the opposite direction. In that case, the non-AMR mesh has $2,560$ cells while the AMR mesh starts with 10 cells and has $l_{max} = 8$. The maximum number of cells involved using the AMR method is indicated in Table 4 and it is of 210 cells which is significantly different from the non-AMR number of cells. This high number of points for the non-AMR method explains why its result is only shown with a line while for the AMR method, each point corresponds to each cell where calculations occur.

Globally the simulation results between the two methods are very close but there are still few things to notice:

- First, with the AMR method, the shock wave and the contact discontinuity are always as well reproduced as the non-AMR one. In fact, the gradient criteria are easily respected at these locations and then the mesh is fully refined.

- Second, concerning the expansion waves. Even if the result is still satisfactory, one can observe that the number of cells at the head of the expansion waves (left part of the expansion waves in the images) decreases. This point is clearly observable in Figure 8 where the levels of refinement distribution is shown, in purple is the initial distribution and in blue is the distribution for the end of the simulation corresponding to the results shown in Figure 9. Initially, the mesh is only fully refined around the

discontinuity (located at $0.7m$) and at the end of the simulation, the mesh is fully refined around the shock, the contact discontinuity and the tail of the expansion waves (right part of the expansion waves in the images). In fact, due to the smooth variation of the thermodynamics variables, it is difficult to find a good criterion to refined the expansion waves without refining most of the domain. Here is only fully refined the tail of the expansion waves because of the normalized variations which is sufficient at this location but not for the rest of the expansion waves.

A supplementary table (Table 5) is presented to point out the ability of the AMR method to bring a real gain in comparison to the non-AMR method concerning the computational time (a CPU time ratio of 26 is reached) and the memory involved, even in 1D simulation. Note that the recorded memory involved is only the highest memory used during the simulation for the AMR method, this one evolves during the simulation in function of the mesh distribution.

			Based on AMR	
Mesh	Computational time	Memory (highest involved for AMR)	Computational time factor	Memory factor
non-AMR	26s	6.0Mo	26.0	2.61
AMR	1s	2.3Mo	1	1

Table 5: Performances for the different tests for the 1D shock tube test case.

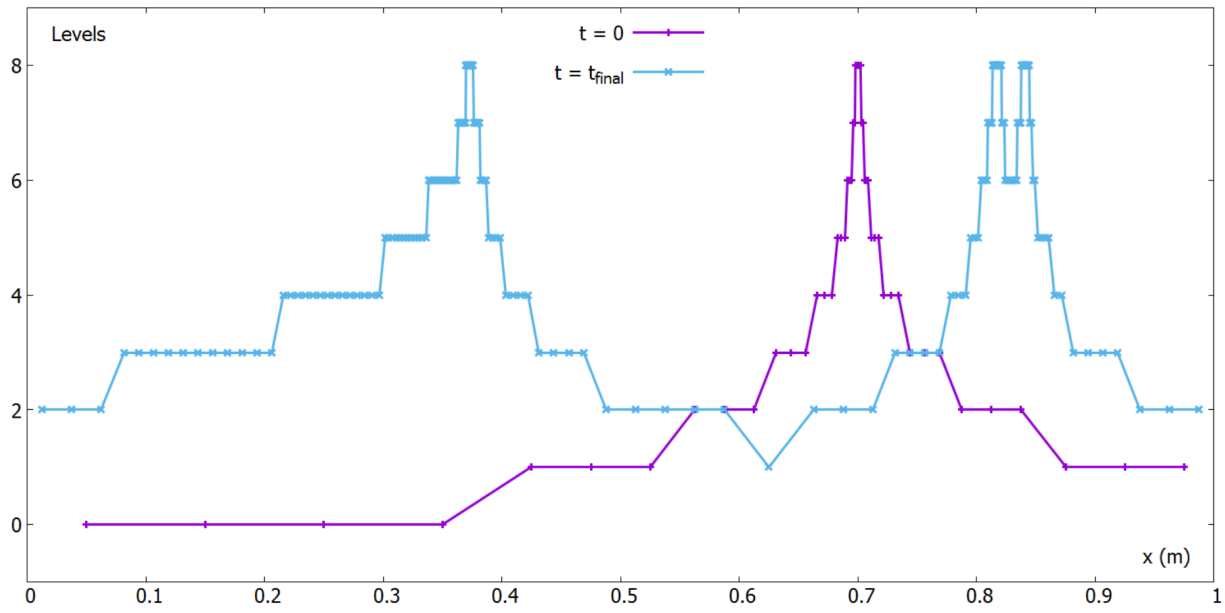

Figure 8: Levels of refinement distribution along the x-direction for the 1D shock tube test case using the AMR method. Initialization (purple crosses) and final result (blue diagonal crosses) are shown. Each cross correspond to a cell where the physical calculations occur.

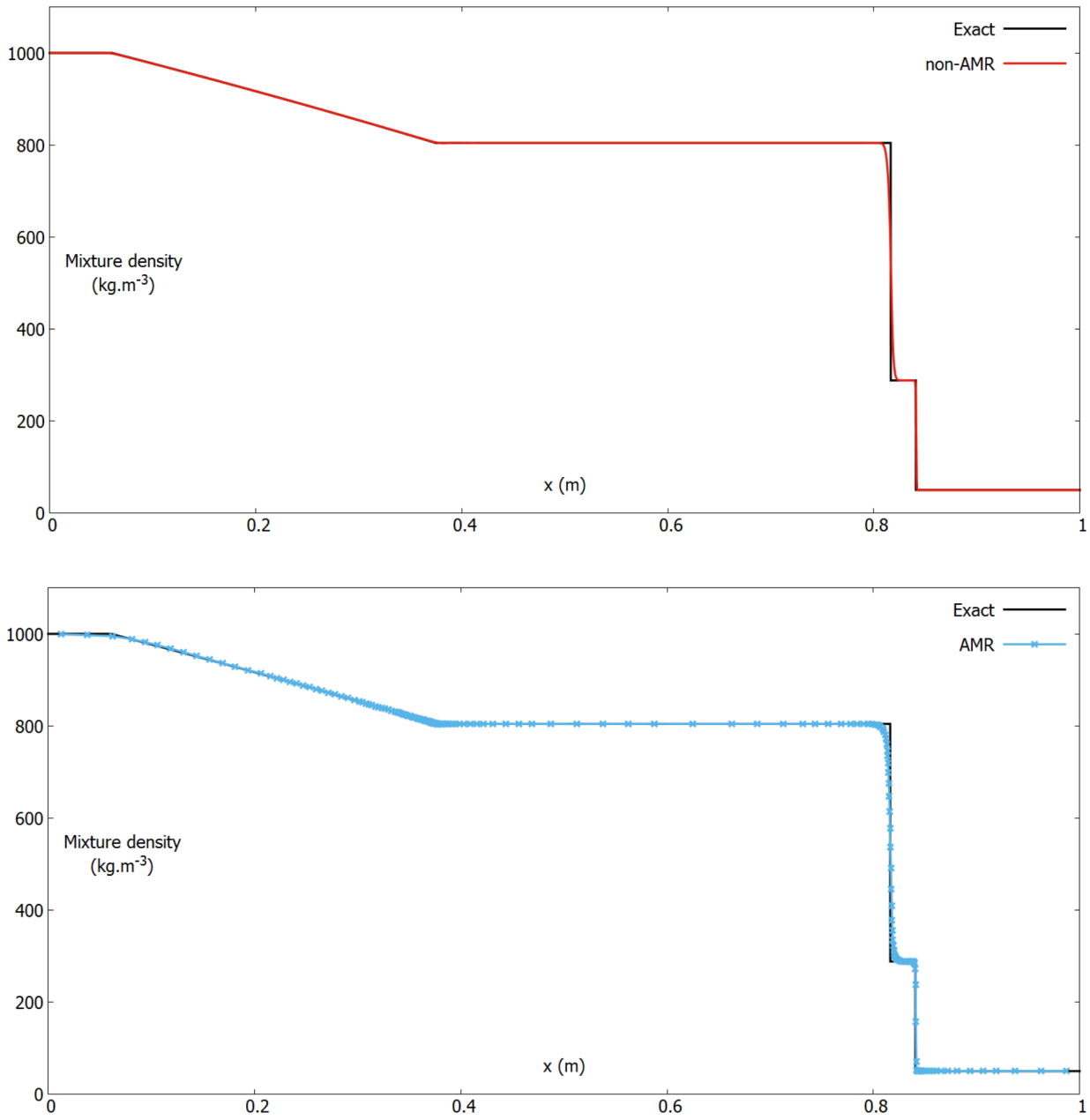

Figure 9: Density ρ along the x-direction for the 1D shock tube test case. Exact solution (black line) is shown and results are given for different tests: non-AMR (top image, red line) and AMR (bottom image, blue crosses and line).

5.3. 2D transport test

A 2D transport test is done here by solving the advection problem of a square of water ($\rho_{water} = 1,000 kg.m^{-3}$) in air ($\rho_{air} = 1 kg.m^{-3}$). The dimension of the domain is $1m$ x $1m$ and at the initialization the center of the square of water is set at the coordinates $0.3m$ on x and y, and its side length is $0.2m$. The velocity of the whole domain is set to $u = 50m.s^{-1}$ in the x-direction and $v = 50m.s^{-1}$ in the y-direction, the pressure P is uniform and the Neumann boundary condition is used at each boundary. The simulation time is $t = 8ms$. The left image in Figure 10 shows the initial state with the AMR mesh (in blue the water and white the air).

Concerning the AMR data, the gradient refinement criterion ϵ can be based on two variables variations: the mixture density and the volume fraction. But, only one is useful since they have a similar shape, then arbitrarily the volume fraction is chosen. The initial mesh is of $N^2 = 10$x10 cells and the maximum number of refinement level is $l_{max} = 4$, then the equivalent fully refined mesh for this test case is of $\left(N^{l_{max}}\right)^2 = 25,600$ cells. l_{max} is not taken as high as in the 1D case ($l_{max} = 8$) due to the induced computational time, specially for the comparison with the non-AMR method (it would have been $6,553,600$ cells). As previously said, the values of the different criteria are case-dependent and due to the 2D accentuate diffusion, the criteria are chosen lower than the ones of the 1D tests, *i.e.*, $\epsilon = 0.08$, $\xi_{split} = 0.05$ and $\xi_{join} = 0.05$. All the information for the AMR data are summarize in Table 6.

Initial number of cells	l_{max}	Equivalent mesh	ϵ	ξ_{split}	ξ_{join}	Max number of cells involved
10x10	4	25,600	0.08	0.05	0.05	3,040

Table 6: AMR data for the 2D transport test case.

On the right of Figure 10 is shown the 2D result of the simulation with the AMR mesh. One can observe that the square of water is well transported: the centre of gravity is at the correct position and the shape is well preserved. Figure 11 shows the mixture density ρ along the diagonal direction ($x = y$) for the initialization state (purple crosses) using the AMR method and for the simulation results using the non-AMR method (red diagonal crosses) and the AMR one (blue stars). It then shows that the AMR method gives almost identical results than the non-AMR one for the 2D transport test case. Note that only the mixture density is shown since all the other variables remain uniform and because the volume fraction has a similar shape than the mixture density.

Even if the number of levels are lower that in the 1D shock tube test case, the performances concerning the computational time and the memory involved are always good for the AMR method in comparison to the non-AMR one (Figure 7). A gain of almost 14 times is obtain for the computational time, which is a bit lower than the previous test case, but the memory factor is higher and it is about 5. The conclusion is that, even if the presented AMR method is not optimized concerning the memory involved, a good factor compared to non-AMR method is obtained. Moreover, if the number of refinement levels is increased, the

computational time and the memory involved drastically increase for the non-AMR method while it is not the case for the AMR one. Indeed, the ratio of computational cells between the AMR and non-AMR methods decreases. Specially adding the fact that the interface is sharper than in the presented case and then a smaller domain of cells at the highest level is computed. In conclusion, even higher performances are expected for the AMR method.

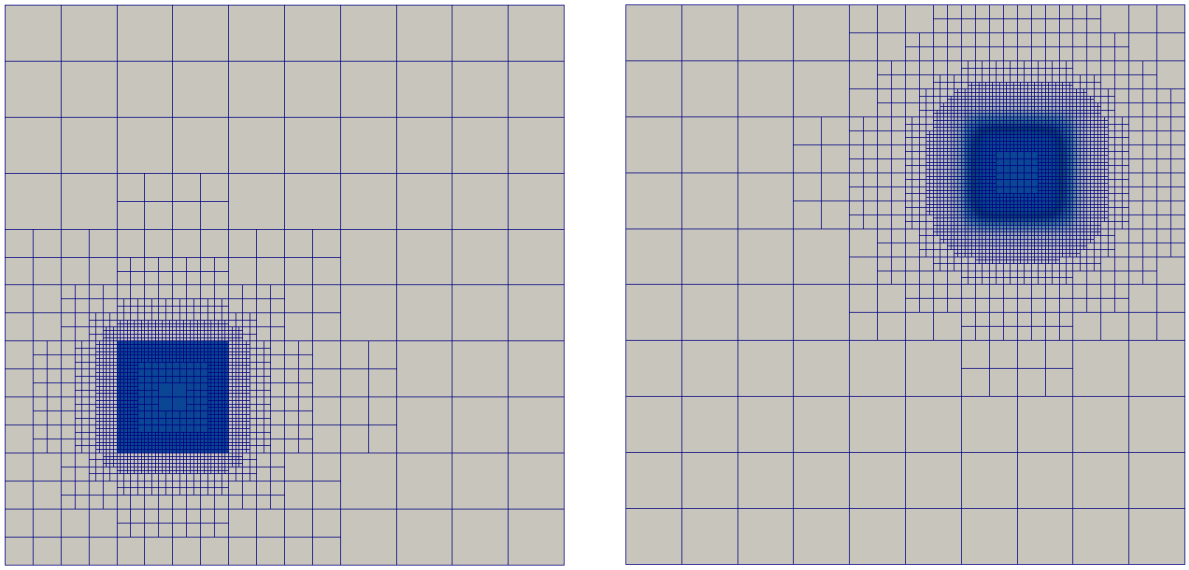

Figure 10: Volume fraction of water α_{water} for the 2D transport test case. In blue the water and white the air. On the left, initialization state, and on the right, the correctly transported result.

			Based on AMR	
Mesh	Computational time	Memory (highest involved for AMR)	Computational time factor	Memory factor
non-AMR (160x160 cells)	36m 49s	57.3Mo	13.636	5.209
AMR	2m 42s	11.0Mo	1	1

Table 7: Performances for the different meshes for the 2D transport test case.

5.4. 3D capillary effects test: Laplace jump

In the following 3D simulation test, a droplet of water is placed in an air environment. When the droplet is at the steady state, the Laplace pressure jump must be recovered thanks to the capillary effects. The expression of the theoretical pressure jump for a sphere is:

$$[P] = \frac{2\sigma}{R},$$

where $[P]$ expresses the pressure jump between inside the droplet and the air, here $P_{water} - P_{air}$.

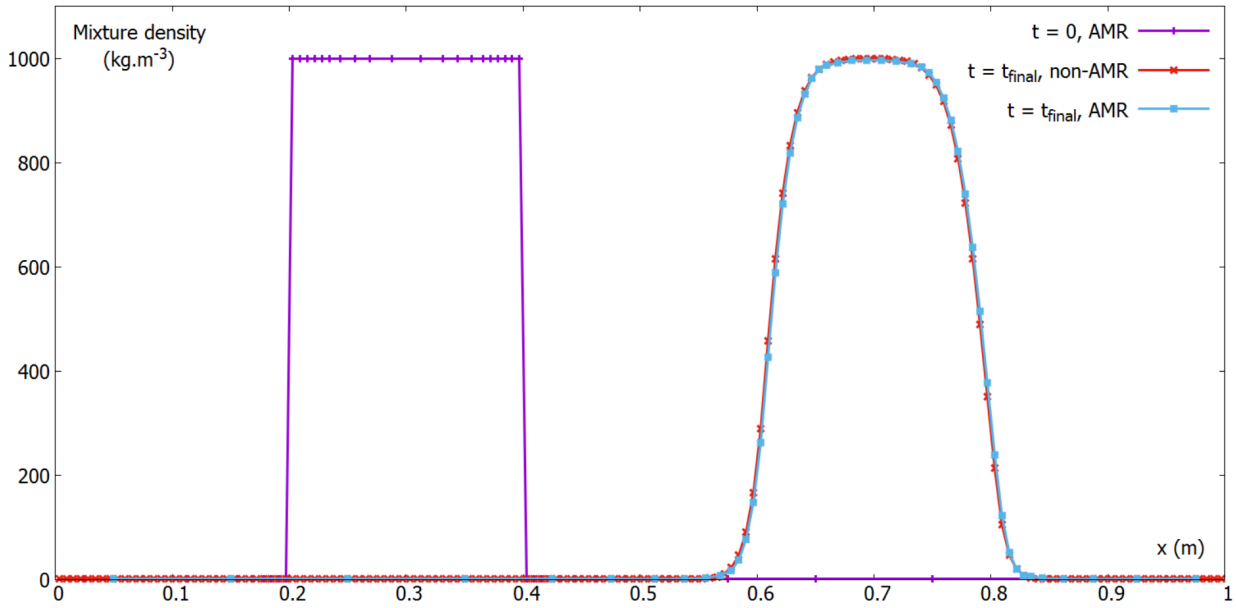

Figure 11: Mixture density ρ along the diagonal direction ($x = y$) for the 2D transport test case. Initialization (purple crosses) and simulation results for the thin mesh (red diagonal crosses) and the AMR mesh (blue stars) are shown.

For the simulation, initially the droplet of water is on a 3D domain filled with air. To accentuate the impact of the surface tension force, the stiffened gas EOS for the water with parameters $\gamma_{water} = 2.1$ and $P_{\infty,water} = 1.10^6 Pa$ is used, the size of the droplet is emphasized (radius of $0.15m$) as the surface tension coefficient $\sigma = 800N.m^{-1}$. At the initialization the pressure is the same in each fluid and is equal to $P = 1.10^5 Pa$, the velocity is null ($\mathbf{u} = \mathbf{0}m.s^{-1}$), the densities are $\rho_{air} = 1kg.m^{-3}$ in the air and $\rho_{water} = 1,000kg.m^{-3}$ in the water, and the outgoing pressure waves boundary condition is used. The volume of the whole domain is $0.75m$ x $0.75m$ x $0.75m$ and at the initialization the droplet is set at the center (see Figure 12). The simulation time to obtain a converged result is $t = 0.59s$.

The information concerning the AMR data is given in Table 8. To correctly treat the capillary effects, only the thickness and the curvature of the interface are important. Thus, the gradient refinement criterion is only based on volume fraction variation. Moremover, as explained for the previous test case, the diffusion is accentuated when dealing with higher dimensions and it directly impact the smoothing of the ξ variable. Thus, the ξ criteria are taking lower than the 2D case, *i.e.*, $\xi_{split} = 0.02$ and $\xi_{join} = 0.02$.

For this test, only the AMR method was computed due to the too long computation of a non-AMR method in 3D. As shown in Table 8, the number of cells that involves a non-AMR calculation is of $4,096,000$ cells while the AMR one only involves a maximum of $267,224$ cells, and this last one already took more than 16 days (on one CPU: Intel® Xeon® CPU E7-4850 v2). Moreover, even if it is not shown here, waves induced by the computation of the capillary effects are coming from the interface locations at the beginning of the simulation. Those ones slowly decrease and disappear while reaching the steady state. In the case of

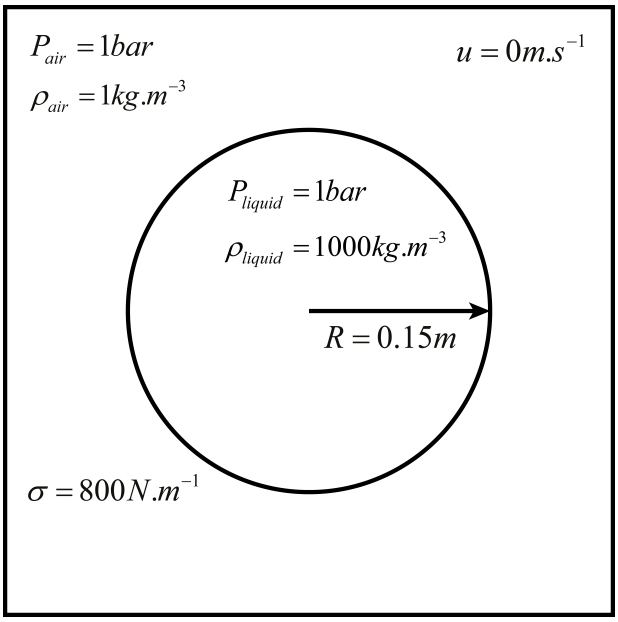

Figure 12: 2D sketch of the initial conditions for the 3D simulation of a liquid droplet placed in air.

the AMR method, the waves are smooth in- and out-side the droplet because of the coarser mesh and thus less oscillations occur than with the fully non-AMR mesh. It results that the AMR method accelerate the convergence of the simulation to a steady state in comparison to non-AMR method.

Initial number of cells	l_{max}	Equivalent mesh	ϵ	ξ_{split}	ξ_{join}	Max number of cells involved	Computational time
20x20x20	3	4,096,000	0.1	0.02	0.02	267,224	16d 21h 53m

Table 8: AMR data for the 3D capillary effects test case.

The theoretical pressure jump of Laplace for this test case is $[P] = 10720Pa$ and it is well recovered in the 3D simulation when using the AMR method as shown in Figures 13 and 14.

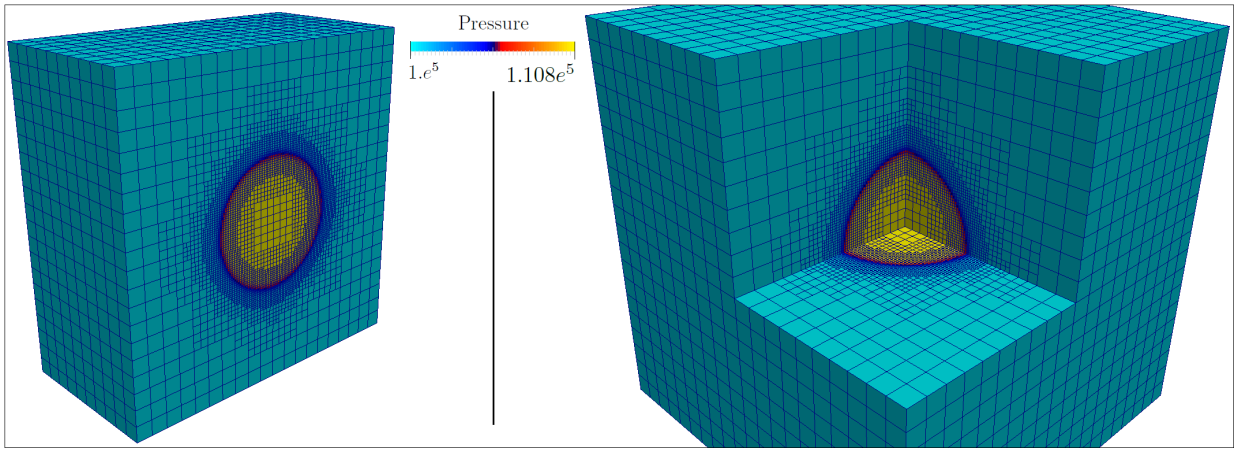

Figure 13: Two different cut views of the the 3D capillary effects test case. Pressure P is shown for the converged state.

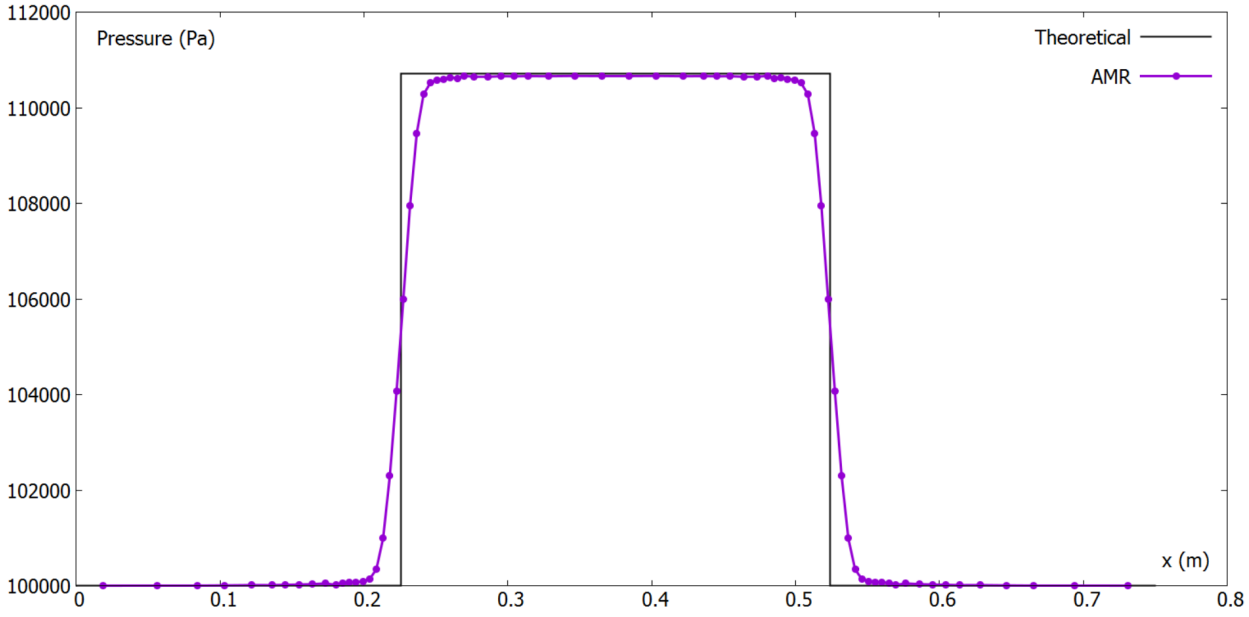

Figure 14: Pressure P at the converged state for the 3D capillary effects test case. Theoretical (black line) and simulation results (purple points) are shown. The converged time is $t = 0.59s$.

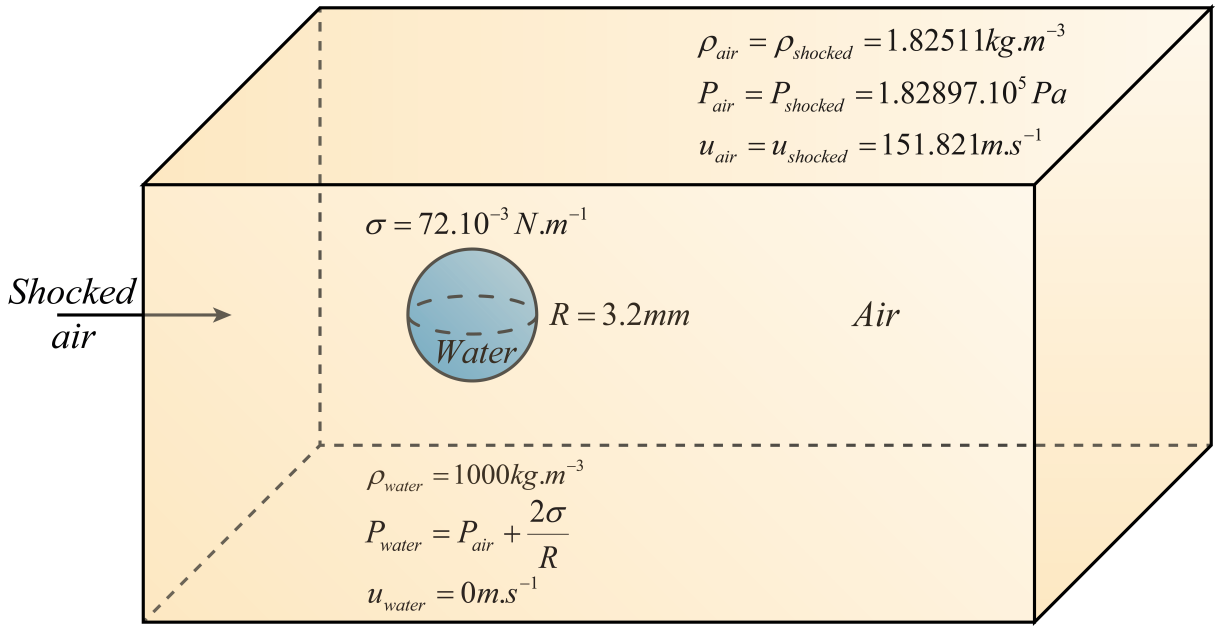

Figure 15: Sketch of the initialization for the droplet atomization test.

The ability of the method on complex flows is presented in this section. Atomization of a 3D water droplet induced by a high speed flow is considered. A spherical water droplet is placed in an air environment already at a shocked state. The initial diameter of the droplet is $D = 6.4mm$ and the shocked state is the corresponding one behind a shock wave of Mach number 1.3 in atmospheric air (see Figure 15 for initialization sketch). The initial densities, pressures and velocities in the x-direction are:

- $\rho_{air} = \rho_{shocked} = 1.82511 kg.m^{-3}$, $P_{air} = P_{shocked} = 1.82897.10^5 Pa$ and $u_{air} = u_{shocked} = 151.821 m.s^{-1}$,
- $\rho_{water} = 1,000 kg.m^{-3}$, $P_{water} = P_{air} + 2\sigma/R$, and $u_{water} = 0 m.s^{-1}$.

The 3D computations are performed on a quarter of the whole domain with two symmetry boundary conditions. Shocked air is entering at the left boundary and the Newmann boundary conditions is used elsewhere. The initial AMR mesh contains 250x50x50 computational cells in a physical domain of 250mm x 50mm x 50mm. The maximum number of refinement levels is $l_{max} = 4$, corresponding to an equivalent non-AMR mesh of $2,560,000,000$ cells. The complete information concerning the AMR data is given in Table 9. The gradient refinement criterion is based on volume fraction, mixture pressure and mixture velocity variations.

An example of the mesh distribution is given in Figure 16 at time $t = 1,000 \mu s$. In the top image, the wireframe is represented for the performed domain with the mixture density gradients colors (dark red for strong gradients and dark blue for null ones). The bottom image represents three quarter of the complete domain with apparent surfaces. One of the

Initial number of cells	l_{max}	Equivalent mesh	ϵ	ξ_{split}	ξ_{join}
250x50x50	4	2,560,000,000	0.02	0.02	0.02

Table 9: AMR data for the droplet atomization test.

two observable surfaces is shown with the mesh and the other without. The refinement at the highest level occurs around the droplet and at its rear due to the vortices that are involved.

Time evolution of the droplet shape is shown in Figure 17. In this figure, water droplet density contour are represented ($\alpha_{water} = 0.5$). The wave propagating on the sides of the droplet due to its compression and stretching is visible. This wave is slowly making filaments all around the droplet. Holes in these filaments are appearing at $1,000\mu s$ and are responsible for finer droplets formation. Fine droplets are visible on Figure 18 at time $1,000\mu s$ with a volume fraction contour of $\alpha_{water} = 0.001$ and colored by the mixture velocity norm. One can observe the speed of the smaller waves which occur on the surface of the droplet, and the velocity of the filaments and of the smaller droplets that are thrown away from the initial droplet (dark red shows strong mixture velocity norms and dark blue shows small ones).

6. Conclusion

A new adaptive mesh refinement (AMR) method using cells boundaries trees has been presented with an extension for multiphase flows. The addition of this second tree boundaries structure presents the advantages to reduce the number of operations during the time step integration (cell neighboring searching is improved) and to simplify the general algorithm in comparison to a fully threaded tree method. The drawback is that the memory involved is not optimized. It is reasonably increased since the number of additional information stored for each cell boundary is relatively small in comparison to what is needed in a cell for the AMR structure as well as for the physical quantities.

The application of the new AMR method on different tests - transport, shock tube, capillary flow and water droplet atomization in 1D, 2D and 3D - was performed with quantitative comparisons regarding exact solutions or non-AMR method results in order to analyze the benefit of this new method. Computational time efficiency and reasonably memory cost have been shown.

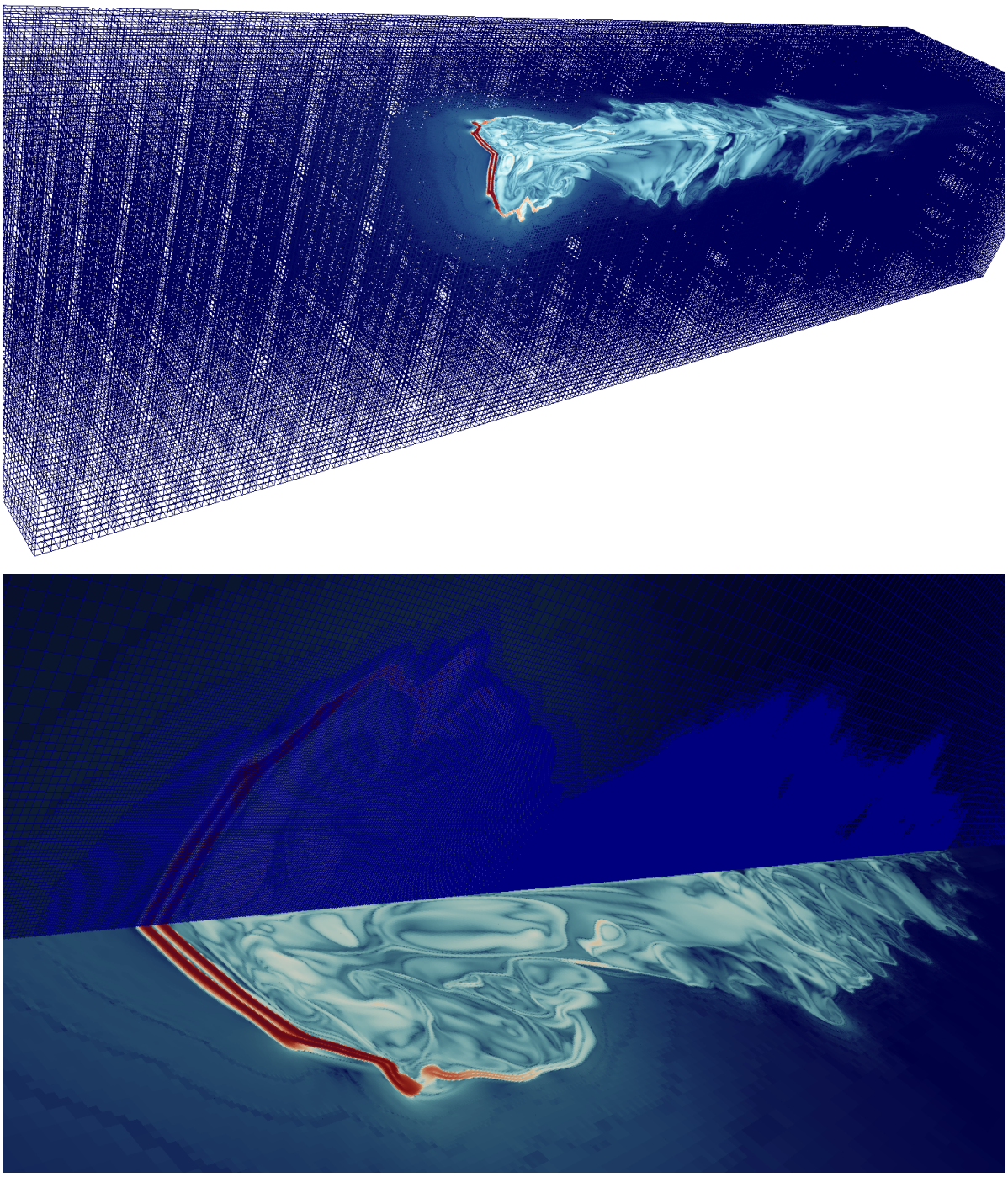

Figure 16: Two images with the mesh and the mixture density gradients apparent (strong gradients in dark red and nulls in dark blue) for the droplet atomization test at time $t = 1,000\mu s$. Refinement of the mesh is observable around the droplet and at its rear (vortices).

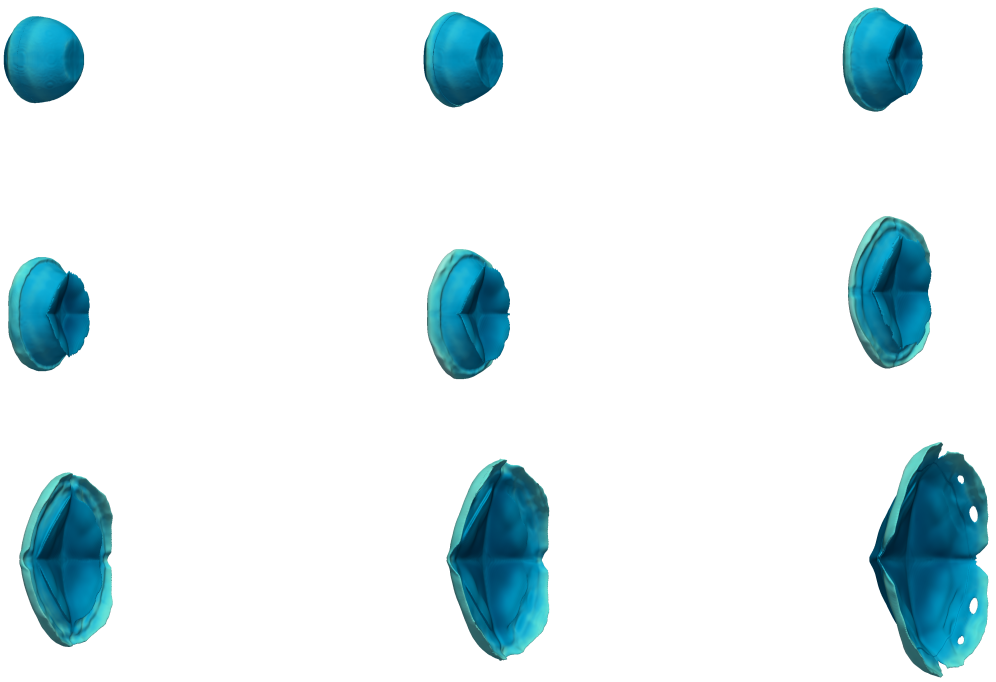

Figure 17: Side views of the droplet contour for the atomization test. Results are shown using $\alpha_{water} = 0.5$ for the contours and at times $200\mu s$, $300\mu s$, $400\mu s$, $500\mu s$, $600\mu s$, $700\mu s$, $800\mu s$, $900\mu s$ and $1,000\mu s$ from the left to the right and from the top to the bottom.

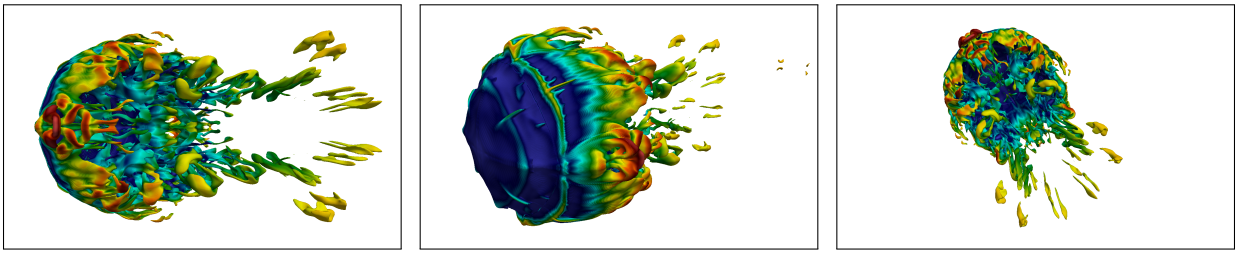

Figure 18: Different views of droplet contour for the atomization test. Results are shown using $\alpha_{water} = 0.001$ for the contours and at time $1,000\mu s$. Colors represents mixture velocity norm (dark red shows strong mixture velocity norms and dark blue shows small ones).

References

[1] M. Aftosmis, J. Melton, and M.J. Berger. Adaptation and surface modeling for cartesian mesh methods. In *AIAA Paper, 12th Computational Fluid Dynamics Conference*, page 1725, 1995.

[2] M. Anderson, E.W. Hirschmann, S.L. Liebling, and D. Neilsen. Relativistic mhd with adaptive mesh refinement. *Classical and Quantum Gravity*, 23(22):6503, 2006.

[3] M.J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of computational Physics*, 82(1):64–84, 1989.

[4] M.J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of computational Physics*, 53(3):484–512, 1984.

[5] X. Chen and V. Yang. Thickness-based adaptive mesh refinement methods for multi-phase flow simulations with thin regions. *Journal of Computational Physics*, 269:22–39, 2014.

[6] W.J. Coirier. An adaptively-refined, cartesian, cell-based scheme for the euler and navier-stokes equations. *Ph.D. thesis, University of Michigan*, 1994.

[7] M. Dumbser, O. Zanotti, A. Hidalgo, and D.S. Balsara. Ader-weno finite volume schemes with space–time adaptive mesh refinement. *Journal of Computational Physics*, 248:257–286, 2013.

[8] N. Favrie and S.L. Gavrilyuk. Diffuse interface model for compressible fluid–compressible elastic–plastic solid interaction. *Journal of Computational Physics*, 231(7):2695–2723, 2012.

[9] N. Favrie and S.L. Gavrilyuk. Dynamic compaction of granular materials. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 469(2160):20130214, 2013.

[10] E. Han, M. Hantke, and S. Müller. Efficient and robust relaxation procedures for multi-component mixtures including phase transition. *Journal of Computational Physics*, 338:217–239, 2017.

[11] S. Hank, N. Favrie, and J. Massoni. Modeling hyperelasticity in non-equilibrium multiphase flows. *Journal of Computational Physics*, 330:65–91, 2017.

[12] A. Hosangadi, V. Ahuja, and S. Arunajatesan. Simulations of cavitating flows using hybrid unstructured meshes. *ASME J. Fluids Eng*, 123:331–340, 2001.

[13] A.M. Khokhlov. Fully threaded tree algorithms for adaptive refinement fluid dynamics simulations. *Journal of Computational Physics*, 143(2):519–543, 1998.

[14] O. Le Métayer, J. Massoni, and R. Saurel. Dynamic relaxation processes in compressible multiphase flows. application to evaporation phenomena. In *Esaim: Proceedings*, volume 40, pages 103–123. EDP Sciences, 2013.

[15] J. Massoni, R. Saurel, B. Nkonga, and R. Abgrall. Proposition de methodes et modeles Euleriens pour les problemes a interfaces entre fluides compressibles en presence de transfert de chaleur. *Int. J. Heat and Mass Transfer*, 45:1287–1307, 2002.

[16] J. Melton, M.J. Berger, M. Aftosmis, and M. Wong. 3d applications of a cartesian grid euler method. In *AIAA Paper, 33rd Aerospace Sciences Meeting and Exhibit*, page 853, 1995.

[17] A. Murrone and H. Guillard. Behavior of upwind scheme in the low mach number limit: Iii. preconditioned dissipation for a five equation two phase model. *Computers & Fluids*, 37(10):1209–1224, 2008.

[18] G.S.H. Pau, J.B. Bell, A.S. Almgren, K.M. Fagnan, and M.J. Lijewski. An adaptive mesh refinement algorithm for compressible two-phase flow in porous media. *Computational Geosciences*, 16(3):577–592, 2012.

[19] F. Petitpas, J. Massoni, R. Saurel, E. Lapebie, and L. Munier. Diffuse interface models for high speed cavitating underwater systems. *Int. J. of Multiphase Flows*, 35(8):747–759, 2009.

[20] F. Petitpas, R. Saurel, E. Franquet, and A. Chinnayya. Modelling detonation waves in condensed energetic materials: Multiphase CJ conditions and multidimensional computations. *Shock waves*, 19(5):377–401, 2009.

[21] S. Popinet and G. Rickard. A tree-based solver for adaptive ocean modelling. *Ocean Modelling*, 16(3):224–249, 2007.

[22] R. Saurel, S.L. Gavrilyuk, and F. Renaud. A multiphase model with internal degrees of freedom: Application to shock-bubble interaction. *Journal of Fluid Mechanics*, 495:283–321, 2003.

[23] R. Saurel and F. Petitpas. Introduction to diffuse interfaces and transformation fronts modelling in compressible media. In *ESAIM: Proceedings*, volume 40, pages 124–143. EDP Sciences, 2013.

[24] R. Saurel, F. Petitpas, and R. Abgrall. Modelling phase transition in metastable liquids: application to cavitating and flashing flows. *Journal of Fluid Mechanics*, 607:313–350, 2008.

[25] R. Saurel, F. Petitpas, and R.A. Berry. Simple and efficient relaxation methods for interfaces separating compressible fluids, cavitating flows and shocks in multiphase mixtures. *Journal of Computational Physics*, 228(5):1678–1712, 2009.

[26] K. Schmidmayer, F. Petitpas, E. Daniel, N. Favrie, and S.L. Gavrilyuk. A model and numerical method for compressible flows with capillary effects. *Journal of Computational Physics*, 334:468–496, 2017.

[27] D.P. Young, R.G. Melvin, M.B. Bieterman, F.T. Johnson, S.S. Samant, and J.E. Bussoletti. A locally refined rectangular grid finite element method: application to computational fluid dynamics and computational physics. *Journal of Computational Physics*, 92(1):1–66, 1991.