ESAIM: PROCEEDINGS AND SURVEYS, Vol. ?, 2019, 1-10
Editors: Will be set by the publisher

CURVED INTERFACE RECONSTRUCTION FOR 2D COMPRESSIBLE MULTI-MATERIAL FLOWS

Igor Chollet[1], Giulia Lissoni[2], Théo Corot, Philippe Hoch, Thomas Leroy, [3] and Laurent Dumas[4]

Abstract. In this paper, we describe an interface reconstruction method in two dimension. This method is an extension of DPIR [1], which reconstructs continuous interfaces and preserves partial volumes using dynamic programming. First we extend the method to curved interfaces. Then, we present tools to improve its robustness in order to apply it to unstructured grid. Finally, we describe an extension to three materials.

Résumé. Dans cette article nous décrivons une technique de reconstruction d'interfaces en dimension deux. Cette méthode est une extension de DPIR [1]. Elle permet de reconstruire des interfaces continues préservant les volumes de chaque fluide en utilisant un algorithme de programmation dynamique. Dans un premier temps, nous étendons l'algorithme à des interfaces courbes. Puis nous proposons des solutions pour améliorer la robustesse de la méthode dans le but de l'appliquer à des maillages non structurés. Finalement, la méthode est étendue au cas trois matériaux.

1. Introduction

Interface reconstruction (IR) methods are encountered in numerical simulation of multi-material or multi-fluid flows. In a Volume of Fluid (VOF) approach in a case of two materials, denote C the volume fraction of material 1 encountered in volume V:

$$C = \frac{1}{V}\int_V \chi(x,y,z)dxdydz \tag{1}$$

where χ denotes the indicator function of material 1:

$$\chi(x,y,z) = \begin{cases} 1 & \text{if } (x,y,z) \in \text{ material } 1 \\ 0 & \text{if } (x,y,z) \in \text{ material } 2 \end{cases}$$

The objective of IR methods is to define a geometric interface separating material 1 and material 2 with the following properties:

- P1: volume fractions conservation,
- P2: continuity of the interface,
- P3: robustness,

[1] Sorbonne Université , institut des sciences du calcul et des donnes, ISCD, F-75005 Paris, France
[2] Université Côte d'Azur, Inria, CNRS, LJAD, Parc Valrose, 06108 Nice, France
[3] CEA DAM DIF, BP 12, 91297 Arpajon Cedex, France
[4] Laboratoire de Mathématiques de Versailles, UVSQ, CNRS, Université Paris-Saclay, 78035 Versailles, France.

- P4: low or moderate computational cost.

The first IR method for volume tracking that has been introduced in 1982 is due to D.L. Youngs [7]. This method consists in assuming that the interface in each mixed cell (that is such $0 < C < 1$) is made of a segment joining two of its edges. The normal to this segment is colinear to the gradient of the volume fraction ∇C and its position is obtained by assuming an exact conservation of partial volumes. With such construction, it is clear that the conservation property P1 is fulfilled by contrast with the continuity property P2. The two other desired properties, robustness and low cost, are also satisfied with this method.

There exist many variants to Youngs method, for instance an order 2 reconstruction ([4]) or an extension to more than two materials ([6]). Some correction terms to the normal computation have also been proposed in order to reduce undesirable effects and to smooth the interface ([2]). The two references [4] and [5] give various examples of applications of Youngs IR method. Even though Young's method is still largely used up to now because of its simplicity and robustness, it suffers from the non continuity of the interface.

Recently, in [1], a new reconstruction method which ensures continuity of the interface and preserves volume fractions has been introduced. This new interface reconstruction method, called DPIR (*Dynamic Programming Interface Reconstruction*), is introduced in the next section and will be used as a starting point for the presented work. It consists of two main steps. First, the minimization of a suitable energy functional to obtain a continuous linear interface. Second, the addition of a control point in each cell in order to recover the correct volume fractions. This point is usually located on the perpendicular bisector of the interface segments.

In this paper there are three main goals. First, DPIR method is extended to curved interfaces (Sec. 3). It is of interest in particular in the case of curved meshes, where an exact reconstruction of the interface is expected. In order to be a serious candidate to be used in multi-material hydrodynamic simulation using ALE remap methods, DPIR method must be able to deal with distorted meshes. Although the principle of the method remains unchanged (one minimization step, one correction step), several improvements are proposed (Sec. 4), in particular to deal with strongly distorted cells and small volume fraction issues. Finally, this work ends with a generalization of the method to three materials (Sec. 5); interface reconstruction for multi-material simulations is a complicated issue, and a comparison of several existing methods can be found in [3]. The proposed method applies DPIR method to all the materials *without choosing any material ordering* and a suitable average is applied to obtain the final interfaces. The method is tested on two test cases with triple point configurations on cartesian meshes, giving encouraging results for future applications to unstructured meshes.

2. Interface reconstruction with DPIR

DPIR method deals with interface reconstruction as a minimization problem of the sum of volume fraction errors. It relies on the minimization of the functional

$$J(y) = \sum_j |vol_{y,j} - vol_j|^2 \tag{2}$$

where $t \mapsto y(t)$ is the associated interface curve, vol_j denotes the targeted volume fraction in cell j and $vol_{y,j}$ the volume fraction obtained with the curve y.

This method is split into two steps. First, dynamic programming is used to minimize the cost function J. Second, a correction is made on the curve y (obtained in each cell) to recover the targeted volume fraction. DPIR algorithm is decribed below, for more information we invite the reader to consult [1].

2.1. First step: minimization of J with dynamic programming

During the first step, the interface curve $t \mapsto y(t)$ is assumed to be piecewise linear in each cell (see Fig. 1). The minimization problem consists in finding a finite number of points $(M_i)_{0 \leq i \leq N}$, located on the edges of

mixed cells, such that $M_0 = M_N$. More precisely, the possible locations of points M_i are obtained after finding the so-called *internal* and *external curves* that will bound the interface curve (see Fig. 1). The internal and external curves are also polygonal curves with nodes located at the mesh nodes and are obtained by a simple search algorithm among mixed cells. Once these curves are found, a dynamic programming procedure is applied to find the piecewise linear closed curve that minimize the cost function J. The computational cost of this step is $\mathcal{O}(NL^2)$ where L is the discretization number of each edge.

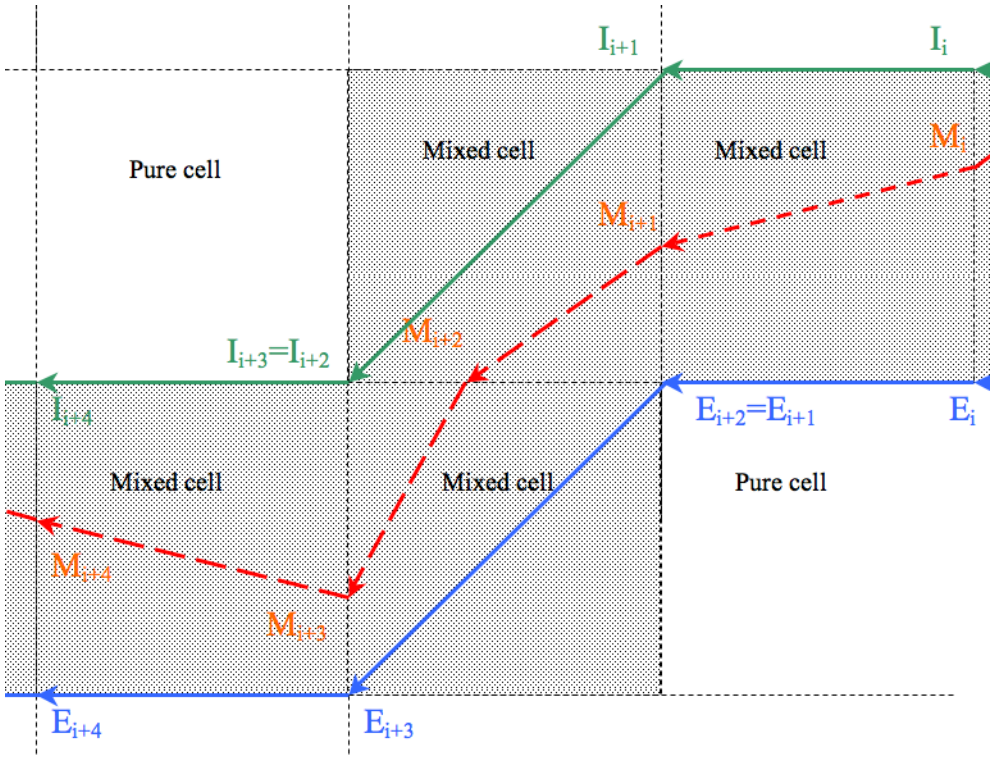

FIGURE 1. The interface curve (dotted line) with the internal (points I_i) and external (points E_i) curves.

Note that a penalty term $p(y) = \sum_{i=0}^{N-1} \lambda ||M_{i+1} - M_i||$ can be added to the cost function J in order to reduce the interface length and avoid wave effects.

Dynamic programming is a very efficient tool to minimize J, it has a low cost (property P4) and is robust (property P3). This step gives a first approximation of the interface which is continuous (property P2). However, the obtained interface does not satisfy the volume conservation (property P1).

2.2. Second step: local correction of volume fractions

The goal of the second step is to correct volume fractions in order to recover Property P1. In the original DPIR algorithm, described in [1], a control point is added in each mixed cell. This point is located on the perpendicular bisector of the interface segment and placed in order to have an exact volume conservation.

The complete method including the previous two steps, called DPIR (Dynamic Programming Interface Reconstruction) can then be summarized by:

DPIR Algorithm: *for a given distribution of volume fractions:*

- **Initialization**: *define the internal and external curves that will bound the interface;*

- **Step 1 (global step)***: minimize the cost functional J (2) with dynamic programming;*
- **Step 2 (local step)***: add a control point in each cell to have an exact conservation of volume fractions.*

In the next sections, we present some extensions of this method. We extend it to curved interfaces in Section 3, describes tools to make it more robust in Section 4 and extend it to three materials in Section 5.

3. DPIR extension to curved interfaces reconstruction

In order to obtain a curved interface, more suited to some cases (circle reconstruction for instance), rational quadratic Bezier curves are introduced in the local correction phase of DPIR.

A second order rational quadratic Bezier curve is a parametric curve defined by three control points P_0, P_1 and P_2. Here, P_1 will play the role of the control point introduced in the correction step. A weight $\omega \in [0, +\infty]$ is associated to this point (Fig. 2).

$$\mathbf{M}^\omega(q) = \begin{pmatrix} x(q) \\ y(q) \end{pmatrix} = \frac{P_0(1-q)^2 + 2\omega q(1-q)P_1 + q^2 P_2}{(1-q)^2 + 2\omega q(1-q) + q^2}, \qquad q \in [0,1].$$

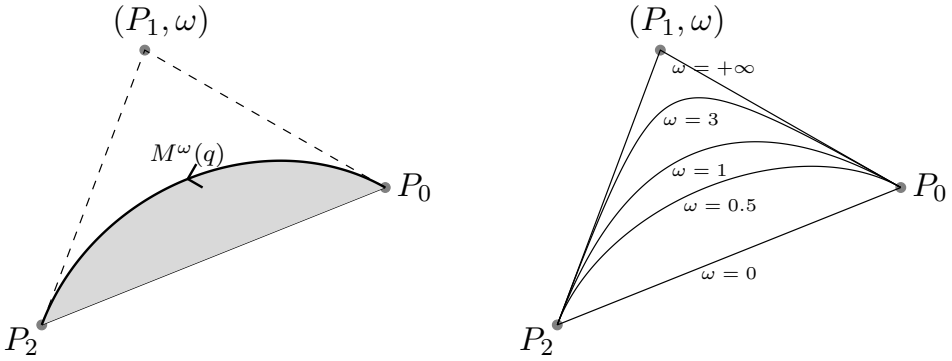

FIGURE 2. A second order rational quadratic Bezier curve (area under Bezier curve and straight segment $[P_0, P_2]$ and shape curve evolution with respect to weight parameter ω) of parameterization (1).

The area $A(\mathbf{M}^\omega(q), P_0, P_1, P_2)$ under a rational quadratic Bezier curve (see Fig. 2 on the left) can be computed as

$$A(\mathbf{M}^\omega(q), P_0, P_1, P_2) = f(\omega) \cdot A(P_0, P_1, P_2),$$

where $A(P_0, P_1, P_2)$ is the area of the triangle $\widehat{P_0P_1P_2}$ and

$$f(\omega) = \begin{cases} 0 & \text{if } \omega = 0, \\ \dfrac{2\omega}{1-\omega^2}\left(\dfrac{1}{1-\omega^2}\arctan\left(\sqrt{\dfrac{1-\omega}{1+\omega}}\right) - \dfrac{\omega}{2}\right) & \text{if } \omega \in (0,1), \\ \dfrac{2}{3} & \text{if } \omega = 1, \\ \dfrac{\omega}{\omega^2-1}\left(\omega - \dfrac{1}{\sqrt{1-\omega^2}}\ln\left(\omega + \sqrt{\omega^2-1}\right)\right) & \text{if } \omega > 1. \end{cases}$$

In this paper, the value of ω will be fixed to 1 if no other specification is made.

Let M_i, $i = 0, \dots N$ be the coordinates of the points obtained after the first step of DPIR. Let P_i be the control point associated with the piece of interface $[M_i, M_{i+1}]$. The position of P_i is defined in order to preserve the

volume fraction defined by the associated rational quadratic Bezier curve (M_i, P_i, M_{i+1}). We use a dichotomy to find the position of P_i. Let us apply this algorithm to the reconstruction of a circle. First we consider a circle on a coarse mesh. We compare the results obtained with Youngs method, the original DPIR and our extension with $\omega = 0.2$. Figure 3 shows that the use of curved interfaces can greatly improve results.

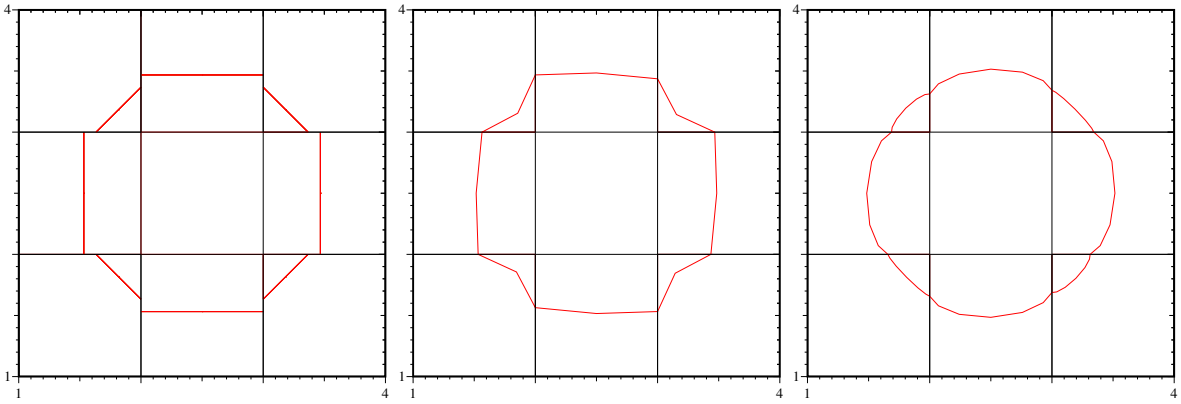

FIGURE 3. Circle reconstruction with Youngs method (left), DPIR (middle) and DPIR using curved interfaces (right).

Then we apply our method to a refined mesh (Fig. 4). Here again, we can see improvements with curved interface.

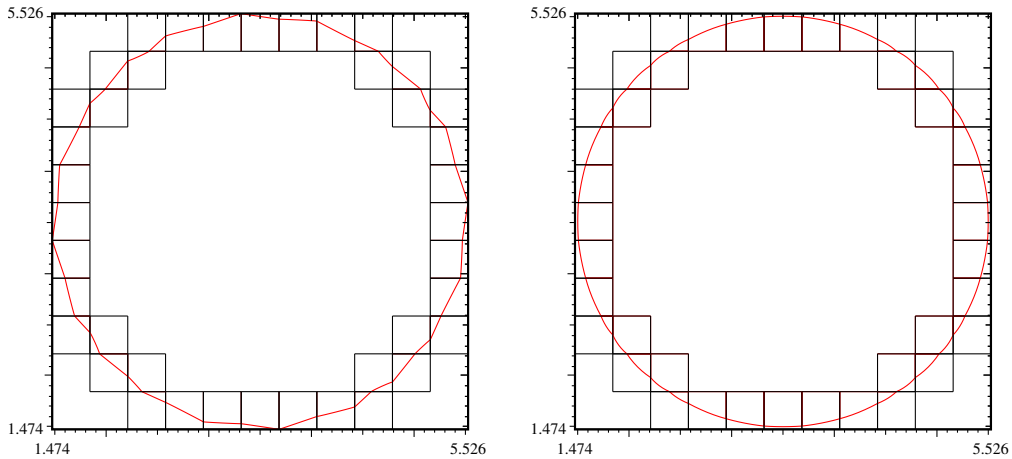

FIGURE 4. Reconstruction of a circle with DPIR without rational quadratic Bezier curves (left) and with rational quadratic Bezier curves (right).

Even if these good results are encouraging, our aim is to obtain a method that works on general meshes. Since we found some lack of robustness, we decided to investigate more deeply this problem in the next section.

4. Robustness improvement of DPIR

To illustrate this robustness problem let us consider the same test as above on an unstructured mesh (Fig. 5). In some cases the points M_i and M_{i+1} obtained after the first step of DPIR can be to close to the same node leading to a spike when the algorithm tries to balance the volume fraction. It has been observed, in some cases, that the algorithm cannot balance the volume since the interface ends up outside the cell.

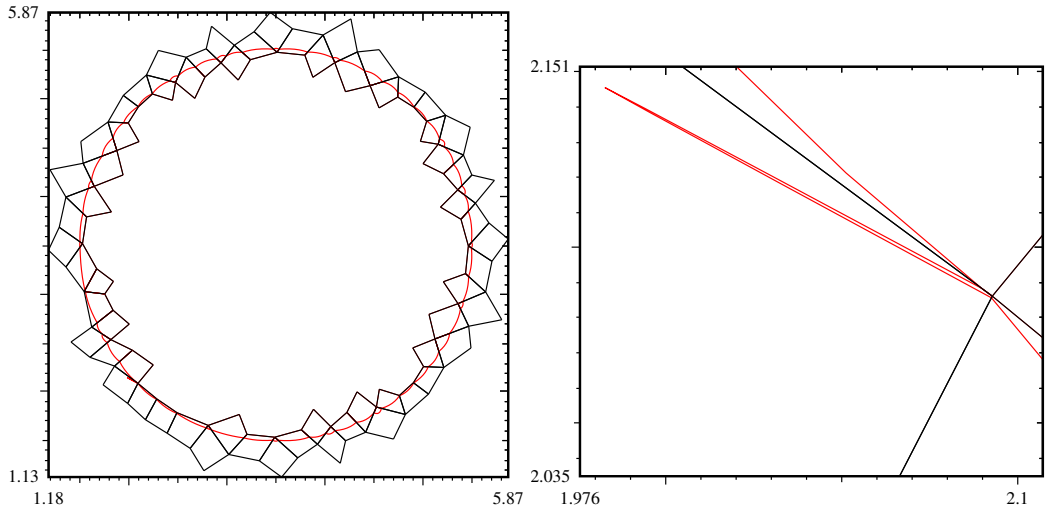

Figure 5. Reconstruction of a circle : some problems with the initial DPIR algorithm.

In order to tackle this problem and improve the robustness of the method, we describe three improvements in the following.

4.1. A new search direction for the control point

A first way to reinforce DPIR robustness is to move the control point in the direction to the cell center (Fig. 6) instead of the perpendicular bisector. This ensures to move in a direction where there is more space available to apply the correction. Consequently it makes the correction step more robust.

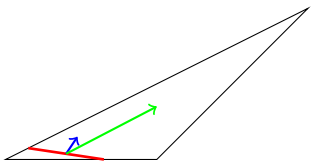

Figure 6. Search direction of the control point: perpendicular bisector vs center of the cell.

4.2. A new discretization of the cell edges

A second idea is to change, in the first step of DPIR, the discretization of the segments crossed by the interface: instead of considering a uniform discretization, we choose Chebyshev points. It gives a discretization which is finer around the corners than in the middle of the segments. Consequently, it permits to avoid points M_i M_{i+1} that are too close to each other and it reduces the number of points on each edge without altering the reconstruction quality.

4.3. **A new penalty term**

It may happen for some cells that the cost function term $vol(M_i, M_{i+1})$ and the penalty term $\|M_{i+1} - M_i\|$ have very different scales. In order to fix this problem, a second penalization term is added to the cost functional:

$$\widetilde{p} = \sum_{i=0}^{N-1} \widetilde{p}_{i,i+1} = \sum_{i=0}^{N-1} \frac{|\mathrm{vol}_{\mathrm{target}} - \mathrm{vol}(M_i, M_{i+1})|}{\mathrm{vol}(M_i, M_{i+1})}.$$

It leads to the following minimization problem:

$$\min_{M_0,\dots M_N} \sum_{i=0}^{N-1} \left(|\mathrm{vol}(M_i, M_{i+1}) - \mathrm{vol}_{\mathrm{target}}|^2 + \lambda \|M_{i+1} - M_i\| + \widetilde{p}_{i,i+1} \right).$$

If the volume fraction between M_i and M_{i+1} is too small with respect to the correction, the second penalization $\widetilde{p}$ becomes big and those points are not chosen by the minimization.

4.4. **Numerical results**

First let us apply these corrections to the reconstruction of the circle on an unstructured grid. Fig. 7 shows that the problem visible on Fig. 5 has been solved.

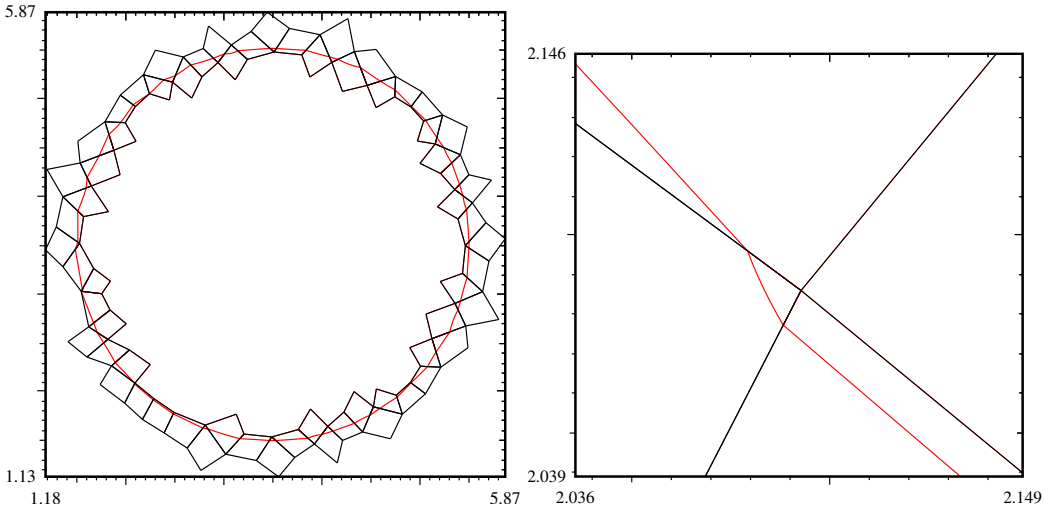

FIGURE 7. Reconstruction of a circle.

We also applied DPIR method for curved interfaces with the three proposed improvements to the reconstruction of a J (Fig. 8) on unperturbed and perturbed cartesian meshes. This test is here to show that we do not lose any precision with these improvements since DPIR already gives good results (see [1]) on this test.

Then, we apply it to the reconstruction of a square on a cartesian mesh. We compare results obtained with classical DPIR to results obtained with DPIR for curved interfaces with $\omega = 1$. First, let us remark that our method is just an extension of classical DPIR: in fact, to recover the original method we just have to set $\omega = \infty$. Results plotted Fig. 9 show that the original DPIR method can still be a good choice since it allows the reconstruction of corners contrary to $\omega < \infty$.

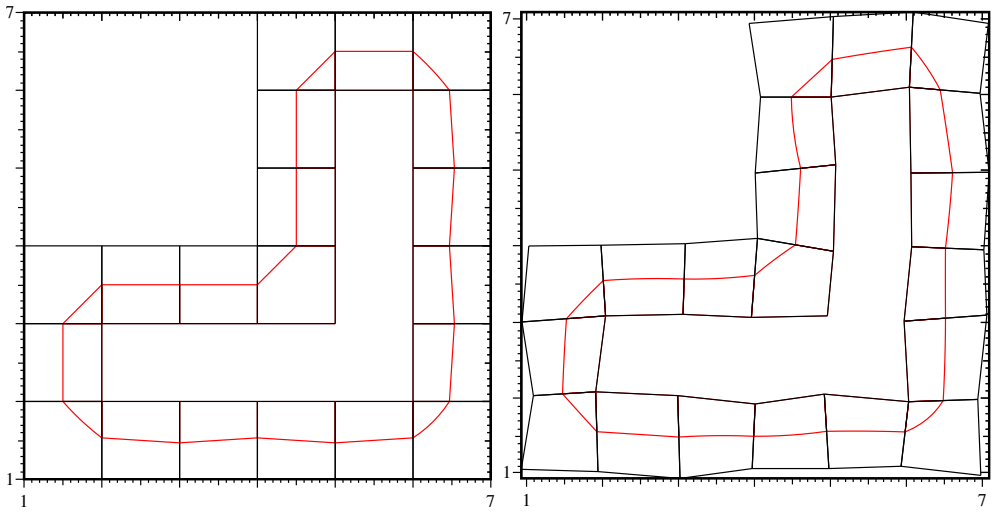

FIGURE 8. Reconstruction of a J on a cartesian mesh (left) and on a distorted cartesian mesh (right).

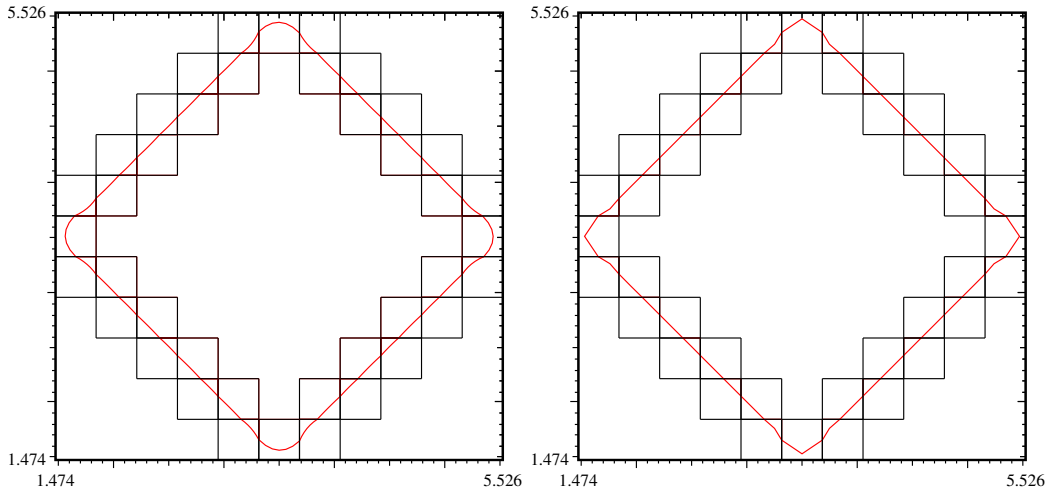

FIGURE 9. Reconstruction of a square on a cartesian mesh with $\omega = 1$ (left) and with classical DPIR, $\omega = \infty$ (right).

Finally, we reconstruct the square on an unstructured triangular mesh using the three improvements and $\omega = \infty$ (see Fig. 10). Let us precise that, on this mesh, DPIR without the robustness improvements is not able to compute an interface. Indeed, the volume correction step leads to interfaces that go out of the cells.

The tools presented in this section to improve the robustness have been shown to be very efficient. These tools permit to use the method on unstructured grids. Indeed, even if the use of DPIR on these kind of meshes is straightforward, without these tools the method tends to be unable to reconstruct the interface.

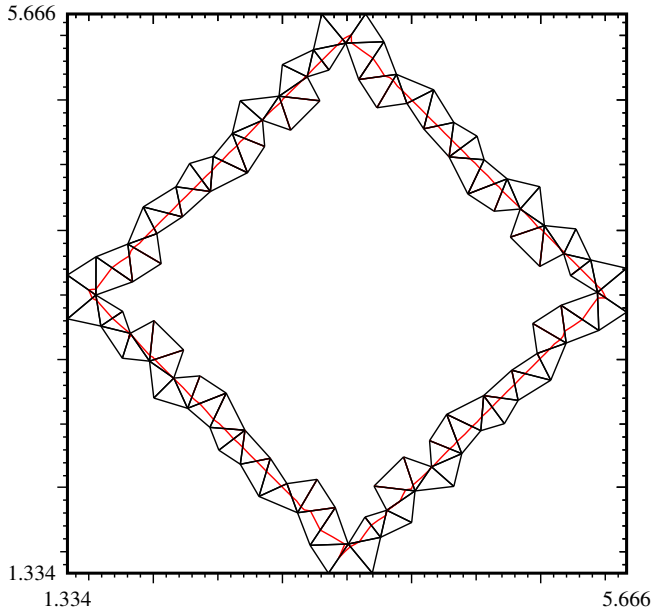

FIGURE 10. Reconstruction of a square on an unstructured mesh with DPIR ($\omega = \infty$) using the robustness improvements.

5. DPIR EXTENSION TO TRIPLE POINT RECONSTRUCTION

In this section, we describe an extension of DPIR to three or more materials. One of the main issue of this extension is the ability of the method to take into account triple points. First we describe different cases that can occur when considering more than two materials. Then we explain how to extend the algorithm to treat triple points. We finally apply it to three and four materials cases.

5.1. **Classification of triple cells**

A cell with exactly two strictly positive volume fractions is called a *double cell*. The treatment of double cells will be the same as above. However the algorithm described above can not handle more than one interface point per edge. When an edge is crossed twice by the interface our method is not able to treat it: DPIR can not handle the so-called *filaments*.

A *triple cell* contains exactly three materials. Two main cases can happen with this kind of cells: it contains a triple point or not. If the cell contains a triple point, we can distinguish the two cases illustrated in Fig. 11a-11b. In the case of Fig. 11b, the red material presents a *filament* (i.e. the interface crosses twice the edge of the cell); since we are not able to deal with this problem even with two material, we will only give some perspectives on this matter at the end. If the cell does not contain a triple point, then materials are aligned in the cell (see Fig. 11c).

We extend DPIR to three materials with a one-against-all approach. First we reconstruct separately the three interfaces, which are then merged to recover the final interface.

5.2. **The new algorithm for interface reconstruction**

The method described below is only suited for the first case of Fig. 11a. Indeed, as explained above, we do not treat *filaments*. The algorithm is still divided in two steps : a first step leading to an interface prediction by running three Dynamic Programming algorithms, and a second step for a local correction of all mixed cells.

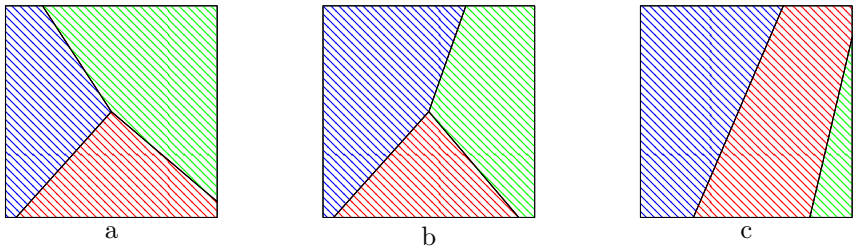

FIGURE 11. Types of triple cells.

5.2.1. *First step: one-against-all approach*

A) Consider each material against all the others. Run the dynamic programming step of DPIR on all those materials. The result of this step is given by the three interfaces illustrated in Fig. 12.
B) Every double cell is composed of edges with exactly 2 or 0 interface points (from outer and inner materials). Average those two points and fix the new resulting point as the final interface point (see Fig. 13).
C) Every triple cell containing a triple point has three edges crossed by two interfaces. Glue the interfaces as previously and compute the temporary triple point as the barycenter of the triangle induced by those three new points (see Fig. 13).

5.2.2. *Second step: mixed cells correction*

Each mixed cell is divided in different sub-cells whose number depends on its type (two for double cells and three for triple cells). We use a different correction algorithm depending on the type of mixed cell. If it is a double cell, we simply use the standard correction step of DPIR in order to correct the partial volumes.

In the case of a triple cell containing a triple point, the objective is to obtain an interface regardless of the order of materials in the correction step. We describe here an algorithm that realizes such a correction.

- Compute signs of correction: decide if a material needs to increase or decrease its volume, by evaluating the sign of the difference between the current volume in the sub-cell and the targeted volume.
- Correct the material that has a correction sign different from the two others by moving the triple point (considered here as a control point) in the leading direction (that is the mean between the two correction directions of sub-cells that have the same correction sign).
- Correct the partial interface between the two others materials using the standard DPIR correction step (adding a control point on the sub-segment). The result of this step is illustrated in Fig. 14.

Let us point out that as soon as you can deal with the filament issue, the case of a cell with three materials aligned (illustrated in Fig. 11c) can be treated in the same way as a double cell.

5.2.3. *Complexity*

The complexity of the algorithm is equal to M times the complexity of DPIR, where M is the number of materials. As all one-against-all DPIR executions are independent, they can be executed in parallel. The correction step only involves local corrections, that are independent. There are two different options :

- from the edge point of view: average contributions of computed temporary interface on each edge (that can be done in parallel) and correct the volume into each cell (possibly in parallel);
- from the cell point of view: average all interface contributions on each edge of the cell crossed by an interface and correct the volume into each cell (possibly in parallel).

Each local correction involves a (small) dichotomy search optimization which compexity is $\mathcal{O}(log_2(|C|/\epsilon))$, with ϵ the requested precision and $|C|$ the diameter of the largest mixed cell of the mesh. If we denote by T the number of mixed cells, the complexity of the correction step is then $\mathcal{O}(Tlog_2(|C|/\epsilon))$ (triple cells only requires 2 dichotomies).

5.3. **A example of DPIR reconstruction on a triple point configuration**

The new DPIR algorithm is applied to the interface reconstruction of a classical triple point test case where materials 1 and 2 are located inside a half sphere and material 3 outside. The different steps of the algorithm on this test are displayed on Fig. 12, 13 and 14.

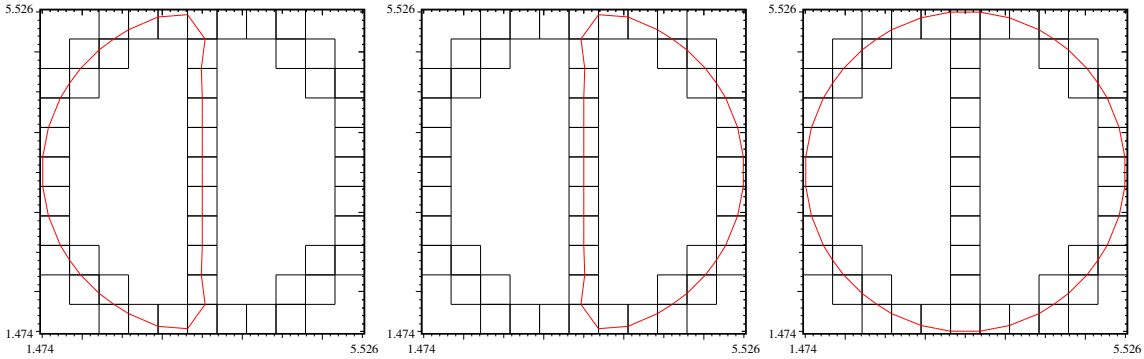

FIGURE 12. A triple point configuration: three interfaces obtained after the Dynamic programming execution, introduced in Sec. 5.2.1, point A).

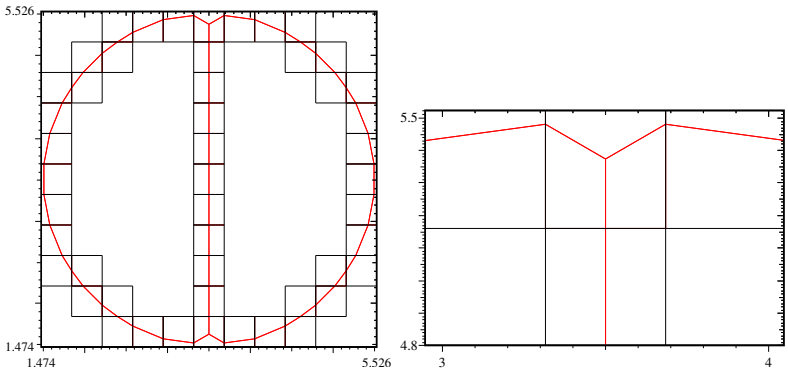

FIGURE 13. A triple point configuration: result after the first step, see Sec. 5.2.1, points B) and C).

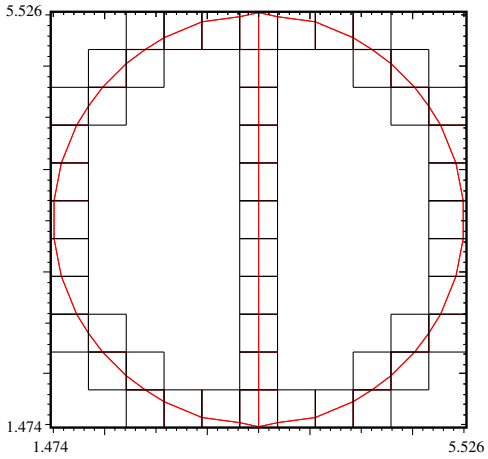

FIGURE 14. A triple point configuration: final result after the second step, see Sec. 5.2.2.

This new algorithm has also been successfully applied to a case with four materials (see Fig. 15).

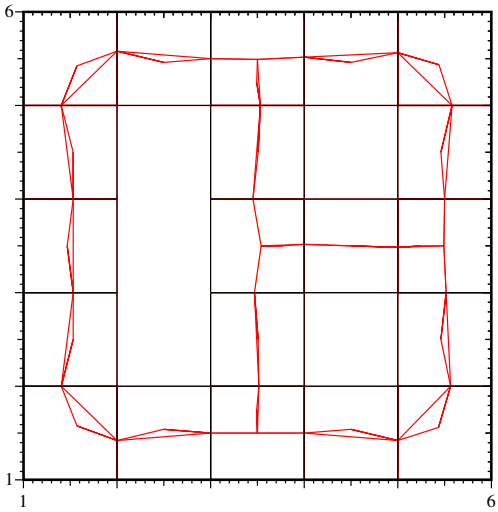

FIGURE 15. Interface reconstruction in a case with 4 materials.

5.4. **Perspectives on the filament issue**

A filament is a portion of the interface of a given material that crosses twice the same edge (Fig. 16). When it appears, one has to detect it before applying the algorithm. Indeed, DPIR can not handle this problem because it looks for one unique interface point per edge.

To deal with this problem, we propose to first detect cells that contain filaments. The edges that should contain two interfaces points are then tagged and split into two sub-edges. Then, methods described above can be applied since we are now looking for one unique interface point per edge (or sub-edge).

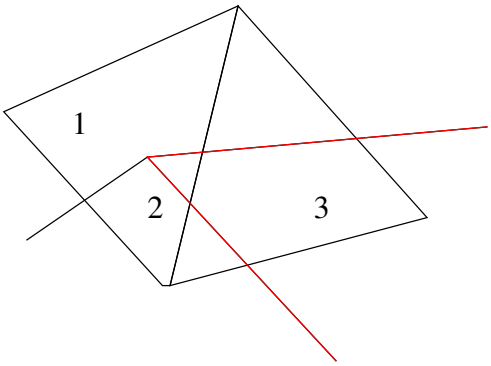

FIGURE 16. Example of filament. The numbers 1,2,3 indicate the three different materials.

6. Conclusion and perspectives

A new and extended version of DPIR algorithm has been developed. It preserves its major advantages, namely the continuity of the interface and the exact conservation of volumes. First, we have extended it to unstructured grids and curved interfaces using rational quadratic Bezier curves. We also described solutions to make it more robust at almost no additional cost.

In a second part, we extended DPIR to three or more materials. In this paper, we chose to use a one-against-all approach even if other methods could be considered in the future.

We are currently working on coupling this method to an ALE scheme in order to use the reconstructed interface with DPIR in anti-diffusive methods and curvature computation to deal with surface tension.

7. Acknowledgment

This work has received financial support from the Commissariat à l'Énergie Atomique. The author I. Chollet also thanks the institut des sciences du calcul et des données of Sorbonne Université.

References

[1] L. Dumas, J.M. Ghidaglia, P. Jaisson, and R. Motte. A new volume-preserving and continuous interface reconstruction method for 2d multimaterial flow. *Int. Journ. for Num. Met. in Fluids*, 1(1):1–2, 2017.

[2] R. Garimella, V. Dyadechko, B.K. Swartz, and M.J. Shashkov. Interface reconstruction in multi-fluid, multi-phase flow simulations. *Proc. of International Meshing Roundtable*, 2005.

[3] M. Kucharik, R.V.V Garimella, S.P. Schofield, and M. Shashkov. A comparative study of interface reconstruction methods for multi-material ale simulations. *Journal of Computational Physics*, 229:pages 2432–2452, 2010.

[4] W.J. Rider and D.B. Kothey. Reconstructing volume tracking. *Journal of Computational Physics*, 141:112–152, 1998.

[5] M. Rudman. Volume tracking methods for interfacial flow calculations. *IJNMF*, 24:671–691, 1997.

[6] S.P. Schofield, R. Garimella, M. Francois, and R. Loubère. A second-order accurate material-order-independent interface reconstruction technique for multi-material flow simulations. *Journal of Computational Physics*, 228:731–745, 2009.

[7] D.L. Youngs. Time dependent multi-material flow with large fluid distorsio. *Numerical Methods for Fluid dynamics*, pages 273–285, 1982.