The 9th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2024
3–7 June 2024, Lisboa, Portugal

DIVERGENCE FREE VELOCITY INTERPOLATION FOR SURFACE MARKER TRACKING

E. Aulisa[1], G. Barbi[2], A. Cervone[2], A. Chierici[1], F. Giangolini[2], S. Manservisi[2] and L. Sirotti[2]

[1] Texas Tech University, Department of Mathematics and Statistics, 1108 Memorial Circle, Lubbock, Texas, USA

[2] University of Bologna - DIN, Via dei Colli 16, 40136 Bologna (BO), Italy
e-mail: giacomo.barbi3@unibo.it

Key words: Finite element interpolation, Raviart-Thomas basis function, interface tracking, multiphase flow, divergence-free velocity

Summary. In the context of multiphase flow simulation, the interface tracking has a crucial role in order to properly preserve the mass of a specific phase and compute all the quantities related to the position of the interface such as the surface tension. In this work, we exploit a point-wise divergence-free finite element representation of the velocity field to improve the mass conservation features of a surface tracking technique based on the reconstruction of the interface through a best-fit quadratic interpolation of a set of markers. In fact, the divergence-free condition of the velocity is strictly related to the mass conservation and can achieve better results than classic bi-linear or bi-quadratic finite element interpolations. The Raviart-Thomas interpolation guarantees that the reconstruction of the field is appropriately divergence-free in each point of the computational domain, differently from the finite element Lagrangian interpolation that is only divergence-free in the weak form (i.e. when integrated on a cell of the domain). The interface tracking technique adopted in this work is based on the marker technique, through which the surface equation is found as the best-fit quadric approximation of the marker positions that are advected in time through a Runge-Kutta 4th order algorithm. The approach is tested with a set of kinematic examples that stress the advection algorithm due to deformation of the initial surface configuration, and compared to the classical Lagrangian interpolation techniques.

1 INTRODUCTION

It is well known that mass conservation is a key feature when numerical simulations of certain types of fluids are considered, such as multiphase flows. This feature can be easily understood if we suppose that the density of the considered phases is represented by a constant value. Therefore, the classical constraint of the vanishing velocity divergence is a natural consequence, that must be addressed with the most suitable numerical tools.

In this work, a divergence-free representation of the velocity field is considered by using the Raviart-Thomas finite element family [1]. With this kind of approximation, the velocity field lies in the space $\mathbf{H}(\mathrm{div})$, defined as the space of square-integrable function with the divergence in the L^2 space. In particular, it is possible to represent a pointwise divergence-free field avoiding the concept of the weak divergence connected to the standard finite element discretization [2]. In this

work, we exploit the lowest-order Raviart-Thomas finite element family ($\mathbf{RT}_0$) to approximate a velocity field in the framework of a multiphase flow problem, to compare this representation with a standard approximation by using a Lagrangian finite element family such as Taylor-Hood.

Additionally, our study is focused on addressing the treatment of the interface between the multiphase phases. Although numerous numerical techniques and algorithms have been developed over the years in the literature to address this issue (VOF, Level Set method, etc.) [3], our attention is drawn to the Front Tracking Method [4].

This methodology has gained interest, particularly in interface tracking within multiphase flow simulations. Different approaches have been explored [5, 6], focusing on marker reconstruction. We introduce a novel numerical algorithm for surface advection on two-dimensional domains, emphasizing marker reconstruction by using the best-fit quadratic interpolating equation. The basic idea of this technique is to advect a set of points that represent the interface, from the initial configuration, with the objective of preserving the topology information of the interface during the simulation transient. Various functions designed to manage marker positions and define interfaces between phases are described, and a comparison between two different types of discrete velocity representation is presented.

This paper is structured as follows: in the first section, we outline the routines employed for the interface approximation by using the marker technique. After that, we provide an overview of the velocity field interpolation, with specific attention to the Raviart–Thomas finite elements. Finally, the numerical results related to the interface advection are presented with a comparison of two different types of velocity interpolation.

2 SURFACE MARKER ALGORITHM

2.1 Marker geometry initialization

The first step of the algorithm involves the implementation of an initialization function, to define the set of the markers and their related parameters. This function is specifically designed to define the initial geometry of the marker cloud, representing one of the two phases within the domain. To achieve this, the library is equipped with two possible methods: the first one provides an explicit set of points with related parameters; otherwise providing a generic bivariate quadratic level-set function. Concerning the first method, it is sufficient to provide a complete set of points along with their geometric coordinates, normal vectors, and curvature for each marker. This allows the initialization of interface shapes that cannot be represented with a quadratic function. Alternatively, the second method involves providing the equation of a generic conic equation in the form:

$$f(x, y) = Ax^2 + Bxy + Cy^2 + Dx + Ey + F\,, \tag{1}$$

where the chosen initial geometry of the marker cloud defines the coefficients A, B, C, D, E, and F. This function also allows the possibility of initializing multiple geometries based on different chosen quadratic functions. After that, the markers can be inserted onto it by providing the desired number of points for every cell. Therefore, giving n_p markers for each cell with their corresponding points coordinates $\mathbf{x}_p = (x_p, y_p)$, the unit normal vector $\hat{\mathbf{n}}$ and the curvature value k can be calculated for each marker [7]. Moreover, the arc length ds associated with every marker of the quadratic function is considered. To evaluate it, an osculating circle is built on the conic connecting the arc length of a circular sector to ds and considering $ds = Rd\theta$.

In addition to the marker initialization, another feature has been provided to identify different phases inside the domain. In particular, at every node of the grid, the sign of the conic surface is considered. If a cell is equipped with a negative sign at every node, that region is regarded as the first phase, and an internal marker is placed at the center of the cell. The advection step is also applied to this new marker inside the cell, to manage the color function variable. Otherwise, if the sign of the conic is positive at every grid node, it indicates the second phase. The third option arises when a cell has nodes with different signs of the conic, representing the set of boundary cells, that are indeed cut by the surface quadratic function.

2.2 Color function evaluation

In the context of multiphase simulations, the tracking of the two phases is an essential requirement. To address this need, the library includes a function capable of evaluating the value of the color function C for every cell. This variable serves as a tool for monitoring the phase under consideration and represents the ratio between the area occupied by one of the two phases A_c and the total area A of the respective cell. Indeed, the value of C varies between 0 and 1, with the extremes of this interval representing the different phases. To illustrate, considering a bubble (phase 1) inside a domain (phase 2), the interior phase can be associated with $C = 1$, while the exterior phase is represented by $C = 0$. Therefore, C can be defined as $C = A_c/A$.

The value of A_c is computed by implementing a linear fit of the interface cell markers, providing the best approximation of these points in the cell with a line. In order to evaluate the area of the interior phase, the linear fit is employed using an exact subdomain polynomial integration [8] implemented for finite elements with linear cuts. As a result, the value of C is a piecewise constant for every cell in the domain. On the other hand, the library also provides the evaluation of a pointwise color function C_n, defined on the grid nodes. Note that, since the advection of a phase can produce thin filaments, the evaluation of C_n requires some attention. Initially, every node inside a cut cell is assigned with a value equal to 0.75 if the sign of the quadratic equation is positive. Otherwise, for the grid nodes with a negative conic sign, we fix the value at 0.25. These values on the grid nodes remain unchanged for boundary cells, i.e. the cells containing the markers. However, for the internal cells, described by $C = 1$, a check on the nodes is performed in order to assign $C_n = 1$ on every node.

2.3 Runge-Kutta advection scheme

After the initialization routine, the marker located on the quadratic function for each boundary cell and the central marker are ready to be advected by a velocity field. In particular, in order to test the library the preliminary simulations have been performed considering an analytical velocity field, without implementing and resolving the Navier-Stokes type equations. On the other hand, the numerical scheme adopted for the advection does not change in the presence of a fully solved velocity field, which derives for example from the resolution of the Navier-Stokes system in the context of multiphase flow simulations.

In the library, the standard Runge-Kutta method has been implemented in order to move the markers in the cell. Although the simulations performed use a 4-th order Runge-Kutta scheme, the library provides the flexibility to perform a generic n-th order scheme. In this section, the 4th-order scheme is described to recall standard results present in the literature about discretized advection schemes. Therefore, considering an initial time t_0 and a fixed time step interval Δt,

the initial value problem is defined as

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}(\mathbf{x}, t)\,, \qquad \text{with} \quad \mathbf{x}(t_0) = \mathbf{x}_0\,, \tag{2}$$

where $\mathbf{v}$ is the velocity field function of the space $\mathbf{x}$ and the time t, and $\mathbf{x}_0$ represents the initial position. To evaluate the position, i.e., the spatial coordinates $\mathbf{x}_i$ for the i-th marker at the time t, we need to solve the following relation

$$\mathbf{x}_i(t) = \mathbf{x}_i(t_{n-1} + \Delta t) = \mathbf{x}_i(t_{n-1}) + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)\Delta t\,, \tag{3}$$

where the coefficients $\mathbf{k}_i$ are defined as

$$\mathbf{k}_1 = \mathbf{v}\left(\mathbf{x}(t_{n-1}), t_{n-1}\right)\,, \qquad \mathbf{k}_2 = \mathbf{v}\left(\mathbf{x}(t_{n-1}) + \mathbf{k}_1\frac{\Delta t}{2}, t_{n-1} + \frac{\Delta t}{2}\right)\,,$$
$$\mathbf{k}_3 = \mathbf{v}\left(\mathbf{x}(t_{n-1}) + \mathbf{k}_2\frac{\Delta t}{2}, t_{n-1} + \frac{\Delta t}{2}\right)\,, \qquad \mathbf{k}_4 = \mathbf{v}\left(\mathbf{x}(t_{n-1}) + \mathbf{k}_3\Delta t, t_{n-1} + \Delta t\right)\,.$$

This numerical scheme is performed at every time step and in every cell to represent the surface motion by moving the markers. This step serves as the initial configuration for the next section, where, starting from the advection result, the rebuilding of the marker position is performed when certain conditions are not satisfied.

2.4 Best-fit quadratic function

After the marker advection for every cell, a crucial aspect of the algorithm is the computation of a conic equation that represents the best-fit approximation of the marker positions. This is achieved by exploiting the information about the marker of a single cell at a specific time step, i.e. position, the normal vector, along with the markers from neighboring cells. The resulting function is then selected from an ellipse, a hyperbola, or a parabola. In particular, the least-square minimization approach is employed to determine the coefficients corresponding to these types of conics, and a criterion is implemented to choose the best one among the three. To briefly summarize, the algorithm for seeking the best-fit bivariate quadratic equation relies solely on the relative positions of the markers.

The first step of the minimization process involves evaluating the barycenter in the cell, considering a weighted set of markers instead of the classical geometrical barycenter. This step is performed using the arc length variable s of each marker, allowing the detection and penalization of regions with a high density of markers (i.e. marker cluster resulting from the advection routine). In particular, the weighted markers are evaluated with a Gaussian distribution curve that depends on the distance between the specific marker and the barycenter, considering also the marker of the neighboring cells. Therefore, the square distance between the barycenter $\mathbf{x}_c$ and the i-th with coordinate $\mathbf{x}_i$ is defined as

$$d_i^2 = (x_i - x_c)^2 + (y_i - y_c)^2\,. \tag{4}$$

The variance related to the Gaussian curve is computed as

$$\sigma^2 = \frac{\sum_{i=1}^{n} d_i^2}{n f_\sigma}\,, \tag{5}$$

where n represents the total number of considered markers, including the markers of the neighboring cells. Additionally, the parameter f_σ can be chosen in order to penalize markers that are distant from the barycenter differently. Therefore, the weight for the i-th marker can be defined using the variance as

$$w_i = s\, e^{-\frac{d_i^2}{2\sigma^2}}\,, \tag{6}$$

by which, a new barycenter point $\mathbf{x}_g$ can be determined as

$$\mathbf{x}_g = \sum_{i=1}^{n} \mathbf{x}_i w_i\,. \tag{7}$$

Let $\delta_i = \sqrt{(x_i - x_g)^2 + (y_i - y_g)^2}$, and $\delta_{max} = \max(\delta_i)$, $i = 1, \cdots, n$ and consider the variables $\chi_i = (x_i - x_g)/\delta_{max}$ and $\psi_i = (y_i - y_g)/\delta_{max}$. With these new variables, we can build a new quadratic cost function to be minimized, i.e. $\sum_{i=1}^{n} w_i(A\chi_i^2 + B\chi_i\psi_i + C\psi_i^2 + D\chi_i + E\psi_i + F)^2$. Hence, we want to find the coefficients $X = [A, B, C, \dots]$ related to the new entries of the cost function matrix M. Therefore, for every marker we can build a corresponding row $M(i,)$ that fills the matrix M which is then decomposed with the Jacobi Singular Value Decomposition (SVD) technique. In particular, for a quadratic interpolation, the six matrix entries are expressed as

$$M(i,) = \left[\chi_i^2, \quad \chi_i\psi_i, \quad \psi_i^2, \quad \chi_i, \quad \psi_i, \quad 1\right]\,. \tag{8}$$

Since the matrix M is built, the Jacobi decomposition $M = USV^*$ is performed by exploiting the linear algebra library Eigen [9]. In particular, considering the right singular vector V matrix, the conic coefficients X that represent the best-fit approximation of the marker position are taken from the last column of V, which represents the minimum eigenvector of the matrix [10].

On the other hand, our analysis has shown that the previous steps are not sufficient to find the best conic coefficients in every cell. Neglecting information about the normal vector can lead to quadrics that are in good agreement with the marker positions, but are not representative of the real interface position. For example, considering a parabola, a method to mitigate this aspect is to align the conic with the principal direction of the marker cloud. Specifically, we can define the $n \times 2$ matrix X, for which the i-th row can be filled with

$$X(i,) = [x_i - x_g, y_i - y_g]\,. \tag{9}$$

Therefore, if we consider the square matrix B, equal to $B = X^T X$ and with dimension 2×2, the eigenvector corresponding to the minimum eigenvalue of B represents the principal direction.

After that, we consider for every marker its outer normal, i.e. $\hat{\mathbf{n}}_i^{old}$ and also $\hat{\mathbf{n}}_{i,quad}$ representing the outer normal vector of the three evaluated conic (ellipse, hyperbola, parabola). Therefore, an additional cost function is built such that, for each conic, computes the value of the distance between the markers and the best-fit quadratic function with a penalizing term. This one is evaluated considering the dot product between the previously cited normal vectors $\hat{\mathbf{n}}_i^{old} \cdot \hat{\mathbf{n}}_{i,quad}$. Finally, the best-fit conic corresponds to the one that is able to preserve the direction of the normal vectors most efficiently, i.e. the conic equipped with the lower value of the cost function previously described.

2.5 Rebuilding marker position

Once the best-fit quadratic function based on the advected marker positions is found, we have the option to rearrange the markers inside the cell. This feature gains interest since it is not possible to control the marker distribution directly after advection. Indeed, the new marker position depends solely on the imposed velocity field. Consequently, even if we start with a homogeneous distribution of the marker inside the cell, this homogeneity may be lost after advection, leading to regions with very high or low marker density. Thus, the ability to regenerate markers with a better distribution based on the best-fit conic can be a useful computational tool.

Furthermore, although the distribution can be managed with the rebuilding operation, we are also interested in maintaining a controlled number of markers inside every cell. Hence, the library is equipped with two parameters n_{min} and n_{max}, representing the minimum and maximum number of possible markers inside the cell. Therefore, if the advection results in a number of markers n for a cell that satisfies $n_{min} \leq n \leq n_{max}$, we do not apply the rebuilding algorithm. Otherwise, we regenerate a fixed number of markers n_0 inside the cell starting from the best-fit quadratic equation.

However, the resulting conic does not allow for a direct remeshing of the markers inside the cell. In certain situations, the conic obtained may have more than two intersections with the cell edges, such as four intersections. To overcome this problem, we perform an adaptive refinement of the cell until we obtain sub-cells with only two intersections with the quadratic equation.

After the adaptive refinement, each cell has only a double intersection, represented by $\mathbf{x}_0 = (x_0, y_0)$, $\mathbf{x}_1 = (x_1, y_1)$, with the edges. The goal is to find the coordinates of the new set of markers located on the conic. We start by finding the center coordinates of the osculating circle to the conic, as

$$\mathbf{x}_c = (x_c, y_c) = \mathbf{x}_m - \frac{\hat{\mathbf{n}}}{k}\,. \tag{10}$$

In this case $\mathbf{x}_m$ represents the mid-point between the intersections $\mathbf{x}_0$ and $\mathbf{x}_1$, $\hat{\mathbf{n}}$ is the outward normal vector, and k is the quadratic function curvature evaluated at $\mathbf{x}_m$. After that, we evaluate the angles between the center of the osculating circle $\mathbf{x}_c$ and the intersections $\mathbf{x}_1$ and $\mathbf{x}_0$ as

$$\theta_0 = \arctan\left(\frac{y_0 - y_c}{x_0 - x_c}\right)\,, \qquad \theta_1 = \arctan\left(\frac{y_1 - y_c}{x_1 - x_c}\right)\,. \tag{11}$$

In addition, we also define other two variables representing the differences between these angles as

$$\Delta\theta_0 = \begin{cases} \theta_1 - \theta_0 & \text{if} \quad \theta_1 > \theta_0 \\ 2\pi + \theta_1 - \theta_0 & \text{if} \quad \theta_1 < \theta_0\,, \end{cases} \tag{12}$$

$$\Delta\theta_1 = \begin{cases} \theta_0 - \theta_1 & \text{if} \quad \theta_0 > \theta_1 \\ 2\pi + \theta_0 - \theta_1 & \text{if} \quad \theta_0 < \theta_1\,. \end{cases} \tag{13}$$

Therefore, we introduce the angle θ_s and the angle variation $\Delta\theta$ with the following conditions

$$\begin{aligned} &\text{if } \Delta\theta_0 \geq \Delta\theta_1 \quad \rightarrow \quad \Delta\theta = \Delta\theta_1 \qquad \theta_s = \theta_1\,, \\ &\text{if } \Delta\theta_0 < \Delta\theta_1 \quad \rightarrow \quad \Delta\theta = \Delta\theta_0 \qquad \theta_s = \theta_0\,. \end{aligned}$$

We define now $\mathbf{v} = \langle v_x, v_y \rangle$ as

$$v_x = R_0 \cos\left(\theta_s + \frac{\Delta\theta}{2}\right), \qquad v_y = R_0 \sin\left(\theta_s + \frac{\Delta\theta}{2}\right), \tag{14}$$

where $R_0 = \sqrt{(x_0 - x_c)^2 + (y_0 - y_c)^2}$ represents the radius of the osculating circle. Then, we can compute the intersection between the conic and the line passing through the circle center $\mathbf{x}_c$ and with the direction given by the vector $\mathbf{v}$. Finally, the new marker positions are found by considering

$$\mathbf{x}_n = \mathbf{x}_c + t_n \mathbf{v}\,, \tag{15}$$

where $t_n = \min(t_1, t_2)$, and t_1 and t_2 are the solutions of $a^* t^2 + b^* t + c^* = 0$. In this case, the normalized coefficients, defined with the $*$, are defined as follows

$$a^* = \frac{a}{\sqrt{a^2 + b^2 + c^2}}\,, \quad b^* = \frac{b}{\sqrt{a^2 + b^2 + c^2}}\,, \quad c^* = \frac{c}{\sqrt{a^2 + b^2 + c^2}}\,, \tag{16}$$

where the a, b, c coefficients are defined considering the quadratic function coefficients, the point $\mathbf{x}_c$ and the vector $\mathbf{v}$ as

$$a = Av_x^2 + Bv_xv_y + Cv_y^2\,, \qquad b = 2Av_xx_c + B(v_yx_c + v_xy_c) + 2Cv_yy_c + Dv_x + Ev_y\,,$$
$$c = Ax_c^2 + Bx_cy_c + Cy_c^2 + Dx_c + Ey_c + F\,.$$

Moreover, we are also able to update for the new marker the arc-length function ds as $ds = \Delta\theta(R_0 + R_n)/2$, where R_n is the distance between $\mathbf{x}_n$ and $\mathbf{x}_c$.

3 VELOCITY INTERPOLATION

This section introduces and motivates the initial integration of Raviart-Thomas basis functions into the surface marker reconstruction library. The main objective is to compare the performance of two different velocity field representations, considering interface advection tests. As the Navier-Stokes equations are not solved here, the velocity field is specified using an analytical function, allowing the comparison of how the analytical velocity field approximation can influence the simulation.

Numerical approximations of the velocity field are naturally stored through finite element discretization, such as the classical 9-point biquadratic quadrilateral element. Similarly, the analytical velocity is represented according to the specific data structure from mesh discretization, with exact values only available at the standard degrees of freedom of the mesh elements. The marker reconstruction library is designed to handle both kinematic and dynamic two-phase flow simulations, accommodating scenarios where the velocity field is either imposed or fully resolved.

For computing velocity at points beyond the nodes of the elements (e.g., marker coordinates), interpolation is required. This is crucial for marker advection, where the Runge-Kutta method needs varying velocity values at different positions. This work compares two finite element interpolation techniques for determining the velocity field at marker locations: standard Lagrangian interpolation and Raviart-Thomas interpolation. The velocity field $\mathbf{u}_{mrk}$ of the i-th marker located at coordinates $\mathbf{x}_p$ is compared using the following formulations

$$\mathbf{u}_{mrk} = \sum_{i=1}^{n_{dof}} \mathbf{u}_i \varphi_i(\mathbf{x}_p) \qquad \mathbf{u}_{mrk} = \sum_{i=1}^{n_{faces}} \mathbf{b}_i(\mathbf{x}_p) p_i\,, \tag{17}$$

where n_{dof} represents the biquadratic nodes for a quadrilateral element, and n_{faces} is the number of faces of the same element. Here, φ_i denotes the biquadratic Lagrangian basis functions, while $\mathbf{b}_i$ are the Raviart-Thomas basis functions acting on the flux face values p_i. Since the computational domain is a standard two-dimensional Cartesian mesh, issues related to the convergence of Raviart-Thomas finite elements do not arise. Notably, Raviart-Thomas finite elements preserve only the normal component of the velocity through the element faces, while the tangential component may not be preserved.

The reason for using Raviart-Thomas basis functions for velocity interpolation is driven by two key factors: first, mass conservation in multiphase flow, where constant density in both phases necessitates satisfying the mass conservation equation. A divergence-free velocity field is crucial, and Raviart-Thomas discretization effectively addresses this requirement. Second, the divergence-free property of the velocity fields used in advection tests, which are sinusoidal functions derived from stream functions, aligns with the $\mathbf{H}(\text{div})$ basis functions that should provide an exact representation of the velocity field at every physical point. Specifically, for the lowest-order Raviart-Thomas element $\mathbf{RT}_0$, the divergence is approximated with a constant value, eventually zero, helping to avoid errors associated with standard Lagrangian interpolation.

In summary, the Runge-Kutta advection function has been adapted to support both interpolation methods. This implementation integrates two finite element libraries written in C++: the FEMuSTTU library [11] for surface marker reconstruction, and the ProXPDE library [12] for Raviart-Thomas finite element interpolation.

4 NUMERICAL RESULTS

4.1 Interface advection test

In this section, numerical results are presented in order to test the algorithm outlined above. In particular, standard tests from the literature are considered benchmark results to evaluate the interface tracking. Flow fields with uniform translations and rotations are used to move the markers in the cells. The numerical tests discussed in [5, 6] are considered examples for the results section. The basic advection motions are intended to shift smooth fluid bodies within a domain while ensuring the preservation of surface shape and volume. The test presented is a two-dimensional problem, with a velocity field characterized by a not uniform vorticity in the domain. As a result, substantial distortions occur at the fluid body interface, complicating its maintenance. To restore the initial configuration at the end of the period, time-dependent analytical velocity functions with a sinusoidal profile are usually used [13].

To facilitate quantitative comparisons of the results, the L_1 error norms presented in [5] have been computed. We can define the *relative mass error* $E_m(t_1)$ to compare the total volume of a phase, in particular the reference volume, at the initial time t_0 and the following time t_1

$$E_m(t_1) = \frac{|\sum_{i=1}^{N_{el}} A_i C_i(t_1) - A_i C_i(t_0)|}{\sum_{i=1}^{N_{el}} A_i C_i(t_0)}\,. \tag{18}$$

In the above definition, $C_i(t)$ represents the color function value at the cell i at time t, A_i represents the area of the cell i meanwhile N_{el} is the total number of the cells. We also introduce

another error, labelled as *geometrical error* $E_g(t_1)$, that can be defined as

$$E_g(t_1) = \sum_{i=1}^{N_{el}} A_i |C_i(t_1) - C_i(t_0)| \,. \tag{19}$$

As previously stated, the goal is to determine whether the final shape matches the initial configuration. To this end, we introduced and calculated an additional type of error, using a circular geometry as the initial shape. Thus, given the radius R and the center (x_c, y_c), the distance between a marker m with position (x_m, y_m) and the center can be evaluated and compared with the radius R. Therefore, we define E_{al} as

$$E_{al} = \sum_{m=1}^{N_m} \left| \sqrt{(x_m - x_c)^2 + (y_m - y_c)^2} - R \right| s_m \,, \tag{20}$$

where, for each marker, the arc length is denoted by s_m.

The method for evaluating the Raviart-Thomas basis functions for the velocity field interpolation involves a single bubble advection test. Therefore, the analytical velocity field used in this simulation is provided by

$$\begin{aligned} u &= 2\sin^2(\pi x)\sin(\pi y)\cos(\pi y)\cos\left(\frac{\pi t}{T}\right) \,, \\ v &= -2\sin(\pi x)\cos(\pi x)\sin^2(\pi y)\cos\left(\frac{\pi t}{T}\right) \,. \end{aligned}$$

The domain under consideration is defined as $\Omega = [-L/2, L/2] \times [-H/2, H/2]$, with $H = L = 1$. The initial circular geometry is centered at $(0, 0.25)$ with a radius of $R = 0.15$. The period T for this test is set to $4\,s$, meaning that the bubble reaches its maximum stretch at $t = 2\,s$. The objective is to compare the performance of a standard Lagrangian interpolation with the Raviart-Thomas interpolation for the velocity field within the surface marker reconstruction algorithm. The simulation evaluates how effectively each interpolation method preserves the accuracy and conservation properties of marker positions during advection.

Table 1: Values of the E_m, E_g and E_{al} errors for different grids, for the Q_2 velocity interpolation.

N_{el}	E_m^Q	E_g^Q	E_{al}^Q
32×32	$1.64 \cdot 10^{-2}$	$1.00 \cdot 10^{-3}$	$9.66 \cdot 10^{-4}$
64×64	$9.88 \cdot 10^{-4}$	$6.69 \cdot 10^{-5}$	$8.52 \cdot 10^{-5}$
128×128	$2.35 \cdot 10^{-4}$	$1.65 \cdot 10^{-5}$	$1.36 \cdot 10^{-5}$

Table 1 presents the error values for the Lagrangian-type interpolation, indicated by the superscript Q, as a function of the number of mesh elements N_{el}, ranging from a 32×32 grid to a 128×128 grid. As expected, each calculated error decreases by increasing the mesh refinement.

The error analysis was also conducted for the $\mathbf{RT}_0$ velocity interpolation, with the results presented in Table 2. For these simulations, similar conclusions can be drawn regarding the evolution of errors: every computed error decreases with the grid refinement.

Table 2: Values of the E_m, E_g and E_{al} errors for different grids, for $\mathbf{RT}_0$ velocity interpolation.

N_{el}	E_m^{RT}	E_g^{RT}	E_{al}^{RT}
32×32	$6.43 \cdot 10^{-2}$	$3.94 \cdot 10^{-3}$	$2.24 \cdot 10^{-3}$
64×64	$1.57 \cdot 10^{-2}$	$1.07 \cdot 10^{-3}$	$5.79 \cdot 10^{-4}$
128×128	$3.73 \cdot 10^{-3}$	$2.59 \cdot 10^{-4}$	$1.42 \cdot 10^{-4}$

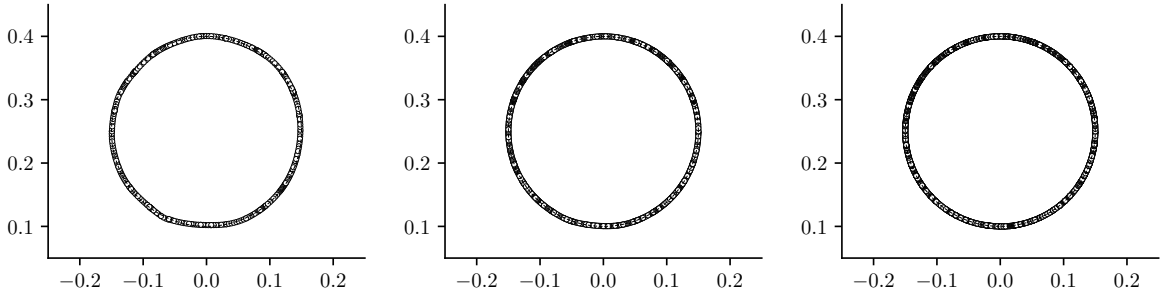

Figure 1: Single bubble test with Q_2 velocity interpolation: comparison of the final interface position at $t = T$ (circular marker) with the initial circular geometry (dashed line) for the meshes with 32×32 (left), 64×64 (center) and 128×128 (right) cells.

When comparing the two techniques, the Q_2 Lagrangian interpolation demonstrates better error results. For each of the computed errors, the magnitude is generally about an order of magnitude lower than that of the corresponding $\mathbf{RT}_0$ interpolation on the same grid. However, it is important to note that these error values are based on the entire simulation, considering both the final values of the color function and the marker positions. While the velocity interpolations for markers may be quite similar within a cell, even a small variation in the velocity field can cause a noticeable difference in the final marker position using the Runge-Kutta advection scheme. Additionally, it is on this final position that every routine of the reconstruction library acts, and thus the resulting best-fit approximating quadratic function can be different.

Figure 1 shows the final positions of the markers for various grid refinements using the standard Lagrangian Q_2 interpolation. Notably, discrepancies from the analytical circular geometry appear only for the coarser grid (32×32). As the grid is refined, the markers' final configuration aligns perfectly with the initial circumference by the end of the period.

Similar observations apply to the $\mathbf{RT}_0$ interpolation, as depicted in Figure 2. For the 32×32 grid, the final marker positions do not perfectly match the initial circular geometry.

In conclusion, the coupling by using Raviart-Thomas interpolated velocity has shown promising preliminary results, despite a slight deterioration in the computed error norm.

5 CONCLUSIONS

This work has focused on the challenges posed by mass conservation in incompressible flow simulations, employing a finite element approach, with a particular focus on the use of Raviart-Thomas basis functions. The main goal was to achieve a divergence-free velocity field, crucial for accurate numerical solutions in engineering applications, ranging from multiphase flows to porous-media flows.

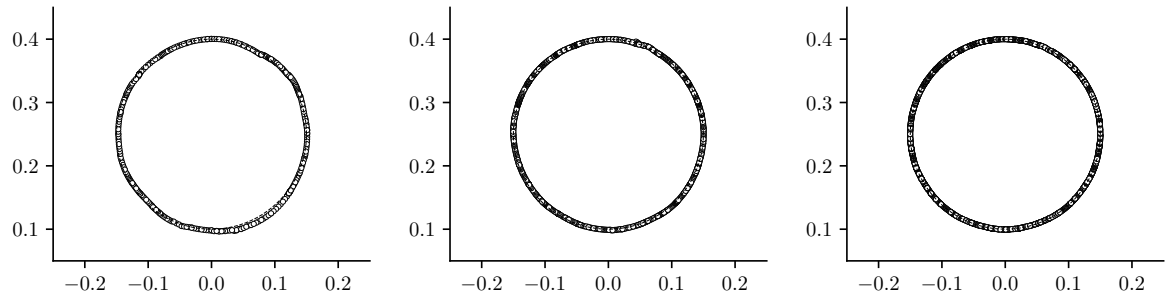

Figure 2: Single bubble velocity with $\mathbf{RT}_0$ velocity interpolation: comparison of the final interface position at $t = T$ (circular marker) with the initial circular geometry (dashed line) for the meshes with 32×32 (left), 64×64 (center) and 128×128 (right) cells.

We focused on multiphase flow simulations, emphasizing the significance of maintaining a divergence-free velocity in scenarios where mass conservation is critical. The novel algorithm introduced for surface advection using marker techniques has shown promising results, particularly in the context of interface tracking within multiphase flow simulations. Specifically, we provide the possibility to advect a marker cloud representing a multiphase interface, and by exploiting a best-fit interpolated quadratic function it is possible to create a new set of markers that represents the surface in the new position. An interesting comparison has been performed, exploiting specific characteristics of Lagrangian and Raviart-Thomas finite elements.

Further investigations will be performed, including the study of three-dimensional geometries and the simulation of dynamic multiphase tests, which involve the resolution of the Navier-Stokes system.

REFERENCES

[1] P. A. Raviart, and J. M. Thomas, *Introduction à l'Analyse Numérique des Équations aux Dérivées Partielles*. Masson, Paris, France, 1983.

[2] D. Boffi, F. Brezzi, and M. Fortin, *Mixed Finite Element Methods and Applications*. Springer, New York, NY, USA, vol. 44, 2013.

[3] F. Chen and H. Hagen, "A survey of interface tracking methods in multiphase fluid visualization," in Visualization of Large and Unstructured Data Sets-Applications in Geospatial Planning, Modeling and Engineering (IRTG 1131 Workshop), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2011.

[4] S. O. Unverdi and G. Tryggvason, "A front-tracking method for viscous, incompressible, multi-fluid flows," Journal of computational physics, vol. 100, no. 1, pp. 25–37, 1992.

[5] E. Aulisa, S. Manservisi, and R. Scardovelli, "A mixed markers and volume-of-fluid method for the reconstruction and advection of interfaces in two-phase and free-boundary flows," Journal of Computational Physics, vol. 188, no. 2, pp. 611–639, 2003.

[6] E. Aulisa, S. Manservisi, and R. Scardovelli, "A surface marker algorithm coupled to an area-preserving marker redistribution method for three-dimensional interface tracking," Journal of Computational Physics, vol. 197, no. 2, pp. 555–584, 2004.

[7] R. Goldman, "Curvature formulas for implicit curves and surfaces," Computer Aided Geometric Design, vol. 22, no. 7, pp. 632–658, 2005.

[8] E. Aulisa and J. Loftin, "Exact subdomain and embedded interface polynomial integration in finite elements with planar cuts," Numerical Algorithms, pp. 1–36, 2023.

[9] G. Guennebaud, B. Jacob, et al., "Eigen v3." http://eigen.tuxfamily.org, 2010.

[10] M. Lawiyuniarti, E. Rahmadiantri, I. Alamsyah, and G. Rachmaputri, "Application of least-squares fitting of ellipse and hyperbola for two dimensional data," in Journal of Physics: Conference Series, vol. 948, p. 012069, IOP Publishing, 2018.

[11] E. Aulisa, S. Bna, and G. Bornia, *Femus*. https://github.com/eaulisa/MyFEMuS.git, 2014.

[12] A. Cervone, *ProXPDE*. https://github.com/capitalaslash/proxpde.git, 2015.

[13] R. J. Leveque, "High-resolution conservative algorithms for advection in incompressible flow," SIAM Journal on Numerical Analysis, vol. 33, no. 2, pp. 627–665, 1996.