MONTANA STATE UNIVERSITY LIBRARY

A mass and momentum conserving unsplit semi-Lagrangian framework for simulating multiphase flows

Authors: Mark Owkes and Olivier Desjardins

NOTICE: this is the author's version of a work that was accepted for publication in Journal of Computational Physics. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Journal of Computational Physics, VOL# 332, ISSUE# 2, March 2017.
DOI#: 10.1016/j.jcp.2016.11.046.

Owkes, Mark, and Olivier Desjardins. "A mass and momentum conserving unsplit semi-Lagrangian framework for simulating multiphase flows." Journal of Computational Physics 332, no. 2 (March 2017): 21-46. DOI: 10.1016/j.jcp.2016.11.046.

Made available through Montana State University's ScholarWorks
scholarworks.montana.edu

A mass and momentum conserving unsplit semi-Lagrangian framework for simulating multiphase flows

Mark Owkes*

Mechanical and Industrial Engineering
Montana State University, Bozeman, MT, 59717, USA

Olivier Desjardins

Sibley School of Mechanical and Aerospace Engineering
Cornell University, Ithaca, NY 14853, USA

Abstract

In this work, we present a computational methodology for convection and advection that handles discontinuities with second order accuracy and maintains conservation to machine precision. This method can transport a variety of discontinuous quantities and is used in the context of an incompressible gas-liquid flow to transport the phase interface, momentum, and scalars. The proposed method provides a modification to the three-dimensional, unsplit, second-order semi-Lagrangian flux method of Owkes & Desjardins (JCP, 2014). The modification adds a refined grid that provides consistent fluxes of mass and momentum on a staggered grid and discrete conservation of mass and momentum, even with flows with large density ratios. Additionally, the refined grid doubles the resolution of the interface without significantly increasing the computational cost over previous non-conservative schemes. This is possible due to a novel partitioning of the semi-Lagrangian fluxes into a small number of simplices. The proposed scheme is tested using canonical verification tests, rising bubbles, and an atomizing liquid jet.

Keywords: Three-dimensional, Gas-liquid flow, Volume-of-fluid (VOF), Scalar transport, Semi-Lagrangian transport, Computational geometry

1. Introduction

In simulations of gas-liquid flows, many conserved quantities are discontinuous at the phase interface, e.g., mass, momentum, and species concentrations. Therefore, a scheme that sharply handles discontinuities is needed for accurate prediction of interface dynamics. Furthermore, it has been shown that inconsistencies in the transport of the discontinuous quantities leads to numerical instabilities near the phase interface [1]. The proposed method allows for second-order accurate transport of discontinuous quantities using a geometric semi-Lagrangian volume-of-fluid (VOF) framework. Furthermore, the scheme consistently transports all conserved quantities and achieves discrete (to machine precision) conservation of mass, momentum, and scalars. The result is a robust framework that is well suited for simulations of many interesting engineering applications with significant interface topology changes, large density ratios, and high shear at the gas-liquid interface.

Various numerical schemes have been developed to track the phase interface location and can be broadly classified as interface tracking and interface capturing. Interface tracking schemes use an explicit representation of the interface and include a mesh that deforms with the interface [2] or Lagrangian marker particles

*Corresponding author
Email address: mark.owkes@montana.edu (Mark Owkes)

on the interface [3]. Interface tracking schemes suffer from the need to frequently perform re-meshing or re-seeding of marker particles when the interface undergoes significant deformation and require manually handling topology changes. Interface capturing schemes use an implicit representation of the interface and include level set methods [3] and VOF methods [4]. Level set methods can be very accurate but do not provide discrete conservation of mass. VOF methods can provide discrete mass conservation with second-order accuracy and are the basis for this work.

VOF methods capture the interface by storing the liquid volume fraction within each computational cell. The first VOF methods were developed in the early 1970s by DeBar [5], Nichols and Hirt [6], and Noh and Woodward [7], who all proposed schemes within a few years of each other. Details of these early schemes and their many improvements are provided by Tryggvason et al. [8]. VOF schemes differ in how to interface is reconstructed within each cell and is transported. Reconstruction provides an explicit representation of the interface and is used in the calculation of updates during the transport step. A common reconstruction is the piecewise linear interface calculation (PLIC) method wherein the interface is represented by a straight line (2D) or plane (3D) within each computational cell [5, 9, 10]. PLIC methods depend on the interface normal vector to orient the reconstruction. In this work, we use the ELVIRA method [11] to compute the normal and orient the PLIC reconstruction. Other normal calculation methods could be used and would not significantly modify the proposed scheme.

Schemes to transport the liquid volume fraction can be categorized as split and unsplit. Early VOF schemes are split and successively compute fluxes in each direction. These schemes are straightforward to implement, but lead to flux splitting errors. Unsplit schemes have been developed and avoid splitting errors [11]. Initially unsplit schemes were developed for two-dimensional simulations [10, 12, 13, 14]. The extension of such schemes to three dimensions is challenging and has only occurred recently by Hernández et al. [15], Le Chenadec and Pitsch [16], Owkes and Desjardins [17], and Zhang [18, 19].

Consistent coupling between the VOF method and momentum transport is essential for robust simulations of realistic flows [1]. Le Chenadec and Pitsch [20] used a geometric semi-Lagrangian methodology [16] to transport both the VOF and momentum to improve consistency. Their approach transports a computational cell backwards or forwards in time during a time-step and an update is computed using geometric operations. Using the approach, they show robust simulations even at large density ratios. However, discrete conservation is not achieved. The conservation error is a direct consequence of the commonly used staggered or marker-and-cell (MAC) grid [8]. On a MAC grid, pressure and any scalars (including the liquid volume fraction) are stored at the cell center and velocity components are stored at cell faces as shown in Fig. 2. In Le Chenadec and Pitsch's scheme [20] the updates for liquid volume fraction (mass) and momentum are computed at different locations and are not consistent. This inconsistency manifests as a conservation error.

The work presented herein can be viewed as an extension to unsplit semi-Lagrangian schemes that adds discrete global and local conservation of mass and momentum. Conservation is achieved by discretizing the liquid volume fraction on a sub-mesh that is twice as fine as the regular grid and computing fluxes of mass that are used to transport both the liquid volume fraction and momentum at every sub-cell face. The sub-mesh follows the work of Rudman [1] and is essential for formulating a scheme on a staggered grid with conservation. Additionally, the sub-mesh provides increased resolution of the phase interface. In Section 3, details of the method are provided and results in Sections 4 and 5 show that the additional cost of the sub-mesh is small and the predicted benefit of conservation of mass, momentum, and scalars is achieved even when large discontinuities exist at the phase interface.

Additionally, the proposed method is built using convective fluxes. Fluxes provide two key benefits: 1) they can be constructed to respect the solenoidal condition ensuring the transported quantity (e.g., mass, momentum, or scalars) are discretely conserved, and 2) a consistent coupling can be made with other discretization. For example, near the interface the proposed geometric fluxes can be used and away from the interface a standard discretization (e.g., finite difference) can be employed. The proposed consistent coupling does not hinder conservation and alleviates excessive computational cost from using the proposed scheme away from the interface where it is not required as the solution is smooth.

In summary, the proposed scheme extends a state-of-the-art VOF method [17] to also transport momentum (and any scalars that may be present) using consistent fluxes. The extension is not straightforward due to the staggered grid, but obtains conservation of all conserved quantities, second order accuracy, and

robustness for complex interface topology changes.

2. Governing Equations

Conservation of mass and momentum for a low Mach number, variable density flow are given in both phases as

$$\frac{\partial \rho_i}{\partial t} + \nabla \cdot (\rho_i \boldsymbol{u}_i) = 0 \quad \text{and} \tag{1}$$

$$\frac{\partial \rho_i \boldsymbol{u}_i}{\partial t} + \nabla \cdot (\rho_i \boldsymbol{u}_i \otimes \boldsymbol{u}_i) = -\nabla p_i + \nabla \cdot \left(\mu \left[\nabla \boldsymbol{u}_i + \nabla \boldsymbol{u}_i^{\mathsf{T}}\right]\right) + \rho_i \boldsymbol{g} \tag{2}$$

where ρ_i is the density, $\boldsymbol{u}_i = [u, v, w]_i$ is the velocity field vector, t is time, p_i is the hydrodynamic pressure, μ is the dynamic viscosity, and $\boldsymbol{g}$ is the gravitational acceleration. The subscript $i = g$ or l indicates variables in the gas or liquid phase, respectively.

The equations above have been written in both the gas and liquid phases. They are connected through jump conditions at the phase interface. For example, the jumps in density and viscosity at the interface Γ are written as

$$[\rho]_\Gamma = \rho_l - \rho_g \quad \text{and} \tag{3}$$

$$[\mu]_\Gamma = \mu_l - \mu_g. \tag{4}$$

In the absence of phase change the velocity field is continuous in the normal direction, i.e., $[\boldsymbol{u} \cdot \boldsymbol{n}]_\Gamma = 0$, where $\boldsymbol{n}$ is the interface normal vector. Analogously to the no-slip assumption, the tangential velocity at the interface is assumed to be continuous and can be written as $[\boldsymbol{u} \cdot \boldsymbol{t}_d]_\Gamma = 0$, for $d = 1, 2$. Combining the two jump conditions for the velocity field, it is clear that the velocity is continuous, i.e.,

$$[\boldsymbol{u}]_\Gamma = 0. \tag{5}$$

The pressure is discontinuous due to contributions from surface tension and the normal component of the viscous stress, i.e.,

$$[p]_\Gamma = \sigma\kappa + 2[\mu]_\Gamma \boldsymbol{n}^{\mathsf{T}} \cdot \nabla \boldsymbol{u} \cdot \boldsymbol{n}, \tag{6}$$

where σ is the surface tension coefficient and κ is the interface curvature.

The transport of a conserved scalar f within the flow is described by

$$\frac{\partial f_i}{\partial t} + \nabla \cdot (f_i \boldsymbol{u}_i) = 0. \tag{7}$$

The scalar may exist in one or both phases and may be discontinuous at the phase interface.

3. Numerical Methodology

The proposed work is built into the NGA computational platform [21]. NGA uses a staggered Cartesian mesh with pressure and other scalars located at the center of each computational cell and velocity components located at the faces of the computational cell. Time discretization is performed using an iterative second-order Crank-Nicolson formulation that incorporates a semi-implicit correction on each sub-iteration similar to the method of Choi and Moin [22]. Away from the phase interface, NGA uses arbitrarily high-order finite difference operators that conservatively transport mass, momentum, and any other scalars [23]. These operators are well suited for simulations of turbulent flows. Near the phase interface the finite difference operators are inappropriate due to discontinuities. Alternatively, the proposed semi-Lagrangian framework is used and provides conservative convective transport even in the presence of the discontinuities.

The proposed scheme builds on the work of Owkes and Desjardins [17] who developed geometric routines to advect the VOF representation of the interface. In the proposed method, the geometric routines convect VOF, momentum, and scalars. All convective fluxes are computed with the same mass fluxes despite the staggered grid arrangement, which is required for a consistent and conservative scheme. Additionally, we present a conservative coupling between the semi-Lagrangian scheme and the finite difference scheme used near and away from the interface, respectively.

3.1. Semi-Lagrangian Formulation of the Conservation Laws

The material evolution of a conserved quantity ψ convected in a solenoidal velocity field is described by

$$\frac{\partial \psi}{\partial t} + \nabla \cdot (\psi \boldsymbol{u}) = 0. \tag{8}$$

This equation can be recast into the *exact* conservation law that describes the evolution of ψ within a control volume CV (e.g., a computational cell) over the discrete time-step $\Delta t = t^{n+1} - t^n$ as

$$\int_{CV} \big(\psi(\boldsymbol{x}, t^{n+1}) - \psi(\boldsymbol{x}, t^n)\big)\, \mathrm{d}V = \sum_{s=1}^{N_S} \int_{\Omega_s} \psi(\boldsymbol{x}, t^n)\, \mathrm{d}V, \tag{9}$$

where N_S is the number of faces that comprise the surface of CV and Ω_s is the signed streaktube emitted from the s^{th} face of a computational cell from $t = t^n$ to t^{n+1} in the velocity field $-\boldsymbol{u}$. The streaktube's orientation provides the volumes sign and is positive if the streaktube transports ψ into the control volume and negative if transport is out of the control volume. Figure 1a shows an example streaktube associated to a computational cell face.

The previous equation can be written more compactly as

$$\psi_{CV}^{n+1} = \psi_{CV}^{n} + \Delta t \sum_{s=1}^{N_S} \mathcal{A}_s \mathcal{F}_s, \tag{10}$$

where ψ_{CV}^n is the integral of $\psi(\boldsymbol{x}, t^n)$ over the control volume, $\mathcal{A}_s$ is the area of the s^{th} face, and $\mathcal{F}_s$ is the flux of ψ through the s^{th} face. The flux can be written as $\mathcal{F}_s = u_s \overline{\psi}_s$, where u_s is the fluxing velocity and $\overline{\psi}_s$ is the average value of $\psi(x, t^n)$ over the streaktube, i.e.,

$$u_s = \frac{\mathcal{V}(\Omega_s)}{\Delta t \mathcal{A}_s} \tag{11}$$

and

$$\overline{\psi}_s = \frac{\int_{\Omega_s} \psi(x, t^n)\, \mathrm{d}V}{\mathcal{V}(\Omega_s)}, \tag{12}$$

where $\mathcal{V}(\Omega_s) = \int_{\Omega_s} \mathrm{d}V$ is the signed volume of Ω_s and the sign comes from the orientation of the volume.

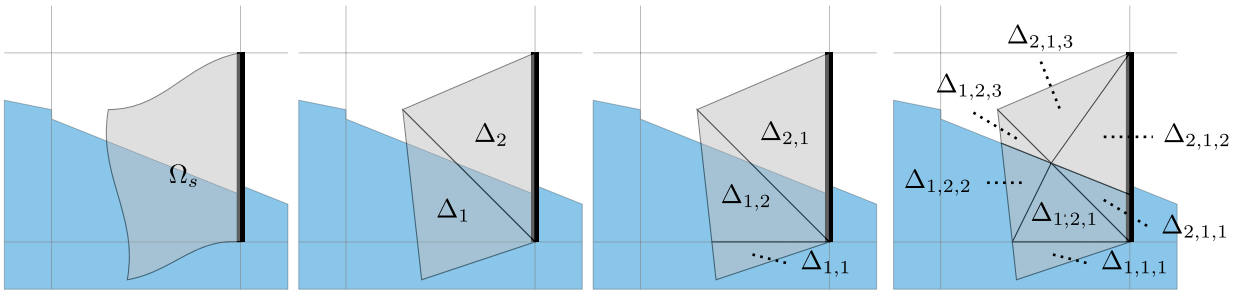
Equations 10 to 12 show the relationship of the semi-Lagrangian formulation with traditional discretization approaches. However, in practice computing $\overline{\psi}_s$ with Eq. 12 is not practical since the signed volume of the streaktube can go to zero. Therefore, the contribution from the flux is computed by combining the equations, resulting in

$$\psi_{CV}^{n+1} = \psi_{CV}^{n} + \sum_{s=1}^{N_S} \int_{\Omega_s} \psi(x, t^n)\, \mathrm{d}V. \tag{13}$$

3.2. Geometric Semi-Lagrangian Fluxes

In realistic velocity fields the streaktubes Ω_s used to compute fluxes can have complex geometries. In order to evaluate the integral in Eq. 12, the streaktube is approximated as a collection of signed simplices (triangles in two dimensions and tetrahedra in three dimensions). Figure 1b shows the streaktube of Fig. 1a approximated with two simplices. Using simplices, the problem of evaluating the integral in Eq. 12 over the streaktube Ω_s is reduced to evaluating the integral over a simplex. The integral is evaluated using the systematic methodology:

1. partition the streaktube Ω_s into a set of simplices Δ_α for $\alpha = 1, \ldots, n_\alpha$, where n_α is the number of simplices (Fig. 1b),



(a) Exact streaktube (b) Step 1: Approximate Ω_s with simplices (c) Step 2: Cut by mesh (d) Step 3: Cut by PLIC

Figure 1: Methodology to partition streaktube Ω_s into simplices within one computational cell and phase. Ω_s is associated with the s^{th} computational cell face indicated with the thick line. The PLIC representation of the interface is shown with liquid shaded in blue.

2. cut each simplex Δ_α by the computational mesh and partition the simplex into a set of new simplices local to one computational cell. The new simplices are $\Delta_{\alpha,\beta}$ for $\beta = 1, \ldots, n_\beta$, where $n_\beta(\alpha)$ is the number of simplices that result from cutting Δ_α with the mesh (Fig. 1c),
3. cut each simplex $\Delta_{\alpha,\beta}$ by the gas-liquid interface and partition the simplex into a set of new simplices local to one computation cell and phase. These new simplices are $\Delta_{\alpha,\beta,\gamma}$ for $\gamma = 1, \ldots, n_\gamma$, where $n_\gamma(\alpha, \beta)$ is the number of simplices that result from cutting $\Delta_{\alpha,\beta}$ by the interface (Fig. 1d),
4. evaluate the integral over each simplex $I_{\Delta_{\alpha,\beta,\gamma}} = \int_{\Delta_{\alpha,\beta,\gamma}} \psi \, \mathrm{d}V$ using data local to the computational cell and the phase simplex $\Delta_{\alpha,\beta,\gamma}$ exists in.
5. compute the original integral in Eq. 12 using

$$\int_{\Omega_s} \psi(\boldsymbol{x}, t^n) \, \mathrm{d}V = \sum_{\alpha=1}^{n_\alpha} \sum_{\beta=1}^{n_\beta(\alpha)} \sum_{\gamma=1}^{n_\gamma(\alpha,\beta)} I_{\Delta_{\alpha,\beta,\gamma}}, \tag{14}$$

Step 1 partitions the streaktube into simplices using the framework shown in Section 3.4. Steps 2 and 3 require cutting a simplex by planes (the computational mesh and the PLIC representation of the phase interface, respectively). This cutting is performed with computational geometry routines [17]. Step 4 evaluates the integral of ψ over a simplex. This evaluation is simplified because the simplex exists in only one computational cell and phase. For example, for mass $\psi = \rho$ the integral is

$$I_{\Delta_{\alpha,\beta,\gamma}} = \int_{\Delta_{\alpha,\beta,\gamma}} \rho \, \mathrm{d}V = \begin{cases} \rho_g \mathcal{V}(\Delta_{\alpha,\beta,\gamma}) & \text{if } \Delta_{\alpha,\beta,\gamma} \text{ is in the gas} \\ \rho_l \, \mathcal{V}(\Delta_{\alpha,\beta,\gamma}) & \text{if } \Delta_{\alpha,\beta,\gamma} \text{ is in the liquid} \end{cases} \tag{15}$$

The evaluation of I for momentum and scalars is provided in Sec. 3.7. The signed volume of a three-dimensional simplex (tetrahedra) is computed using

$$\mathcal{V}(\Delta_{\alpha,\beta,\gamma}) = -\frac{1}{6}((\boldsymbol{a} - \boldsymbol{d}) \cdot ((\boldsymbol{b} - \boldsymbol{d}) \times (\boldsymbol{c} - \boldsymbol{d})), \tag{16}$$

where $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ and $\boldsymbol{d}$ are the ordered vertices of simplex $\Delta_{\alpha,\beta,\gamma}$. Finally, Step 5 sums of the integrals computed in Step 4.

3.3. Introduction of Sub-Cells

In the previous section, we introduced the semi-Lagrangian discretization of a conservation law using an arbitrary control volume CV. In this section, we introduce the cells and sub-cells that are the control volumes that store the conserved quantities.

Figure 2 shows a portion of a staggered Cartesian mesh. On the mesh, transport of of α, ρ, and $\rho\boldsymbol{u}$ requires fluxes at cell faces (x and y) and cell midpoint faces (x_m and y_m). To construct consistent fluxes, we follow the ideas of Rudman [1] and use a finer mesh to store α (and thus $\rho = \rho_l\alpha + \rho_g(1-\alpha)$). Figure 3 shows how the proposed scheme partitions a two-dimensional pressure cell into sub-cells, the streaktubes associated with these sub-cells, and the simplices used to partition the streaktubes. The same methodology is used to partition a three-dimensional pressure cell (Fig. 4) and define streaktubes. The sub-cells are indexed using $\alpha_{s,i,j,k}$ which denotes the liquid volume fraction within the i, j, k pressure cell and s sub-cell (note that on Fig. 4 the i, j, k subscripts are omitted for clarity).

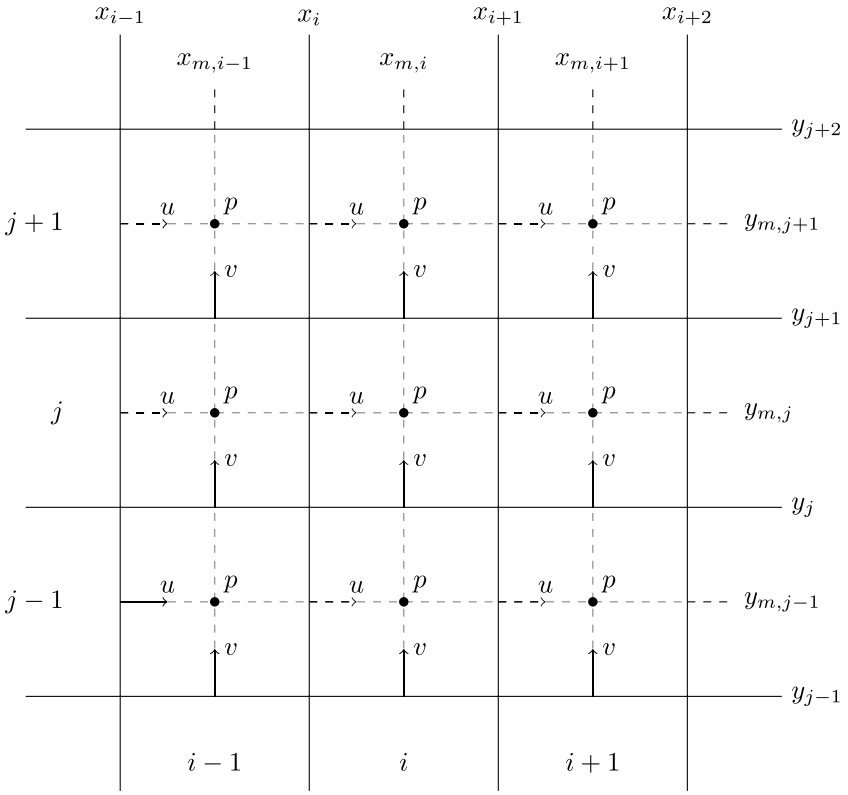


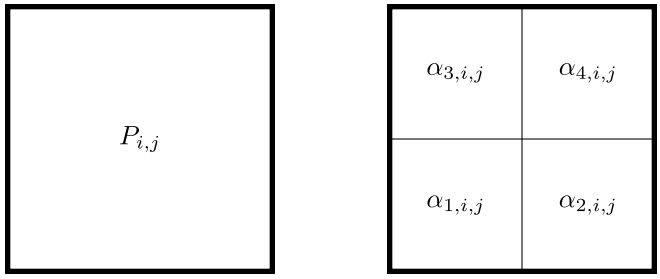
Figure 2: Computational mesh with location of the velocities u and v and pressure p. Location of scalars is at the cell center, with the exception of the liquid volume fraction α, which is stored on a finer mesh. The figure shows the x and y coordinates on computational cell faces and x_m and y_m located at the middle of computational cells.

3.4. Discrete Representation of Streaktube

The proposed scheme represents a streaktube using a collection of simplices. This discrete streaktube is created by transporting each vertex on the cell face along a streak-line from t to $t+\Delta t$ in a reversed velocity field. That is, the motion of the vertex $\boldsymbol{x}_v$ is described by

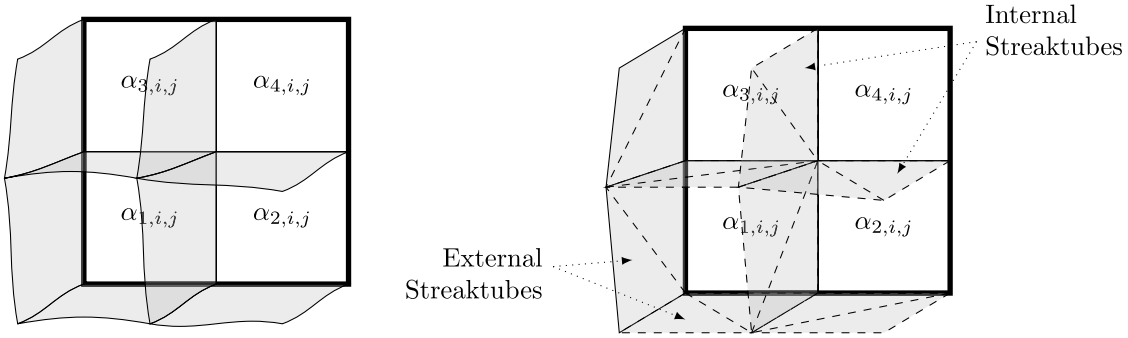
$$\frac{\mathrm{d}\boldsymbol{x}_v(t)}{\mathrm{d}t} = -\boldsymbol{u}(\boldsymbol{x}_v(t), t) \tag{17}$$

with initial condition $\boldsymbol{x}_v(t_0) = \boldsymbol{x}_{v,0}$ on the cell face. The proposed scheme discretizes the previous equation using a 4^{th}-order Runge-Kutta method. During each time-step when streaktubes are constructed, the velocity field is taken to be time-invariant. The proposed scheme uses the second-order mid-time step velocity from the Crank-Nicolson time iteration. This velocity field is interpolated to the vertex location using tri-linear interpolation.



(a) i^{th} and j^{th} pressure cell

(b) Partitioning of pressure cell into sub-cells



(c) Streaktubes on sub-cell faces shown with shaded regions.

(d) Approximation and partitioning of streaktubes using simplices. Streaktubes on external and internal faces indicated with labels.

Figure 3: Construction of simplices that represent semi-Lagrangian streaktubes on a two-dimensional computational cell.

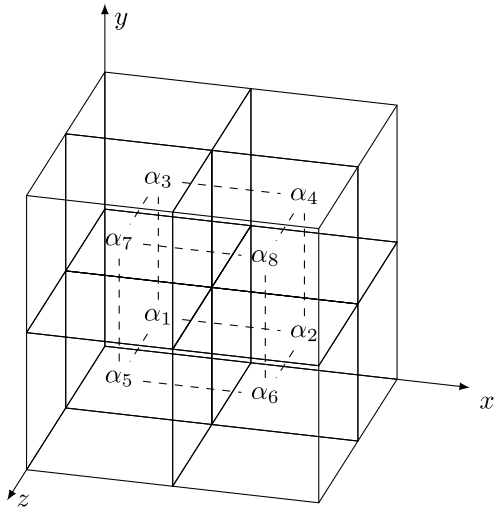
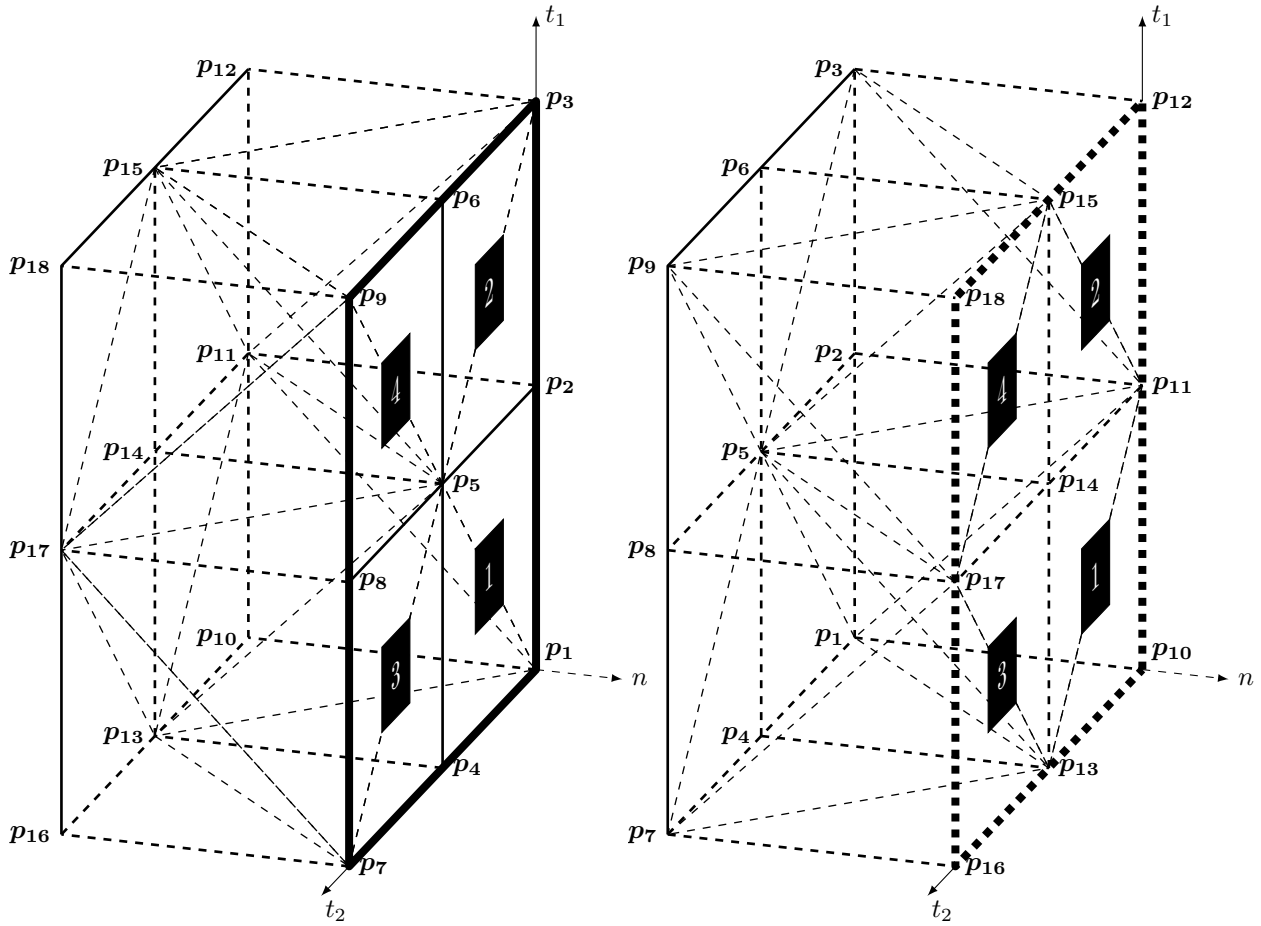


Figure 4: Sub-Cells within a pressure cell in three-dimensions. Dashed lines show connectivity of sub-cell center vertices.

Different partitionings are used on the external (x, y, and z) and internal (x_m, y_m, and z_m) sub-cell faces (see Fig. 3d). The different partition creates streaktubes that use the same representation of all shared faces without introducing gaps or overlaps between neighboring streaktubes. Figures 5a and 5b show the two partitions used to discretize the streaktubes on a external and internal faces, respectively. These external and internal partitions work in all directions and the local coordinate system $\{n, t_1, t_2\}$ in Figs. 5a and 5b takes the values $\{x, y, z\}$, $\{y, z, x\}$, and $\{z, x, y\}$.

The partitioning of an internal face is exactly reversed from the partitioning of an external face and our implementation uses the same lookup table to construct the 20 simplices from the location of the 18 vertices that form the streaktube (Table 1). However, due to reversal of simplices for internal faces, the sign of each simplex in Fig. 5b has a negative orientation. This reversals appear as sign changes to the orientation of the simplex and must be accounted for when evaluating the integrals in Eq. 14.



(a) Discrete streaktube for external face.

(b) Discrete streaktube for internal face.

Figure 5: Partitioning of streaktube into a set of simplices. External and internal pressure cell faces indicated with thick line. Sub-face numbers indicated with numbers in black rectangles.

3.5. Correction of Discrete Streaktubes

In an incompressible flow framework on a staggered mesh, the velocity vectors on the pressure cell faces are modified, through the pressure term, such that $\nabla \cdot \boldsymbol{u} = 0$. However, the *discrete* streaktubes shown in Fig. 5 are not guaranteed to be solenoidal. Methods have been proposed to improve conservation [14, 15, 17, 24, 25]. Many of these methods are limited to two-dimensions and/or introduce a conservation error. Owkes

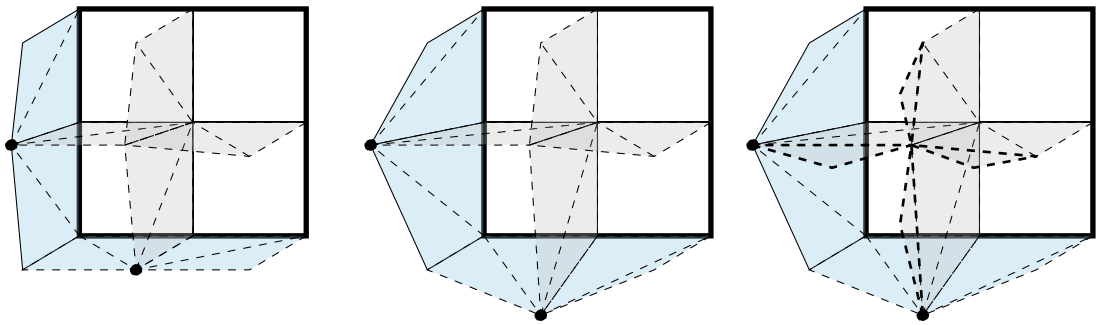
Table 1: Partitioning of a streaktube into a collection of 20 simplices. Numbers correspond to points shown in Fig. 5. Subscripts label the simplices as $S_{\text{sub-face,simplex}}$.

Sub-face 1					Sub-face 2					Sub-face 3					Sub-face 4				
$S_{1,1}$	1	4	5	13	$S_{2,1}$	2	5	3	11	$S_{3,1}$	4	7	5	13	$S_{4,1}$	5	6	15	9
$S_{1,2}$	1	13	11	10	$S_{2,2}$	3	5	6	15	$S_{3,2}$	7	17	13	16	$S_{4,2}$	8	9	5	17
$S_{1,3}$	1	2	11	5	$S_{2,3}$	3	5	15	11	$S_{3,3}$	5	7	8	17	$S_{4,3}$	9	18	15	17
$S_{1,4}$	1	5	11	13	$S_{2,4}$	3	12	11	15	$S_{3,4}$	5	7	17	13	$S_{4,4}$	5	9	15	17
$S_{1,5}$	5	11	13	14	$S_{2,5}$	5	15	11	14	$S_{3,5}$	5	13	17	14	$S_{4,5}$	5	17	15	14

and Desjardins proposed a three-dimensional conservative approach and is the basis for the correction used in this work [17]. The correction modifies the volume of the discrete streaktube such that the fluxing velocity u_s computed using Eq. 11 matches the solenoidal velocity u_{sol}. The modification changes the volume of the streaktube without modify streaktube faces shared with neighboring streaktubes. The proposed scheme uses different corrections for external and internal faces as described below.

3.5.1. External Streaktube Correction

External faces are corrected by adjusting one point that defines the streaktube associated with the computational cell face. Figure 6 demonstrates the methodology applied to a two-dimensional streaktube. The external streaktubes are corrected by moving the two highlighted points. In three-dimensions, the correction modifies all four sub-cell streaktubes associated with the external cell faces (Fig. 5a) such that their aggregate flux velocity matches the solenoidal flux velocity. Instead of working with velocities, we introduce the solenoidal flux volume $\mathcal{V}_{\text{sol}} = u_{\text{sol}}\Delta t\mathcal{A}_s$. Stated with volumes, the correction modifies the aggregate volume of the 4 sub-cell streaktube to equate with the solenoidal flux volume.



(a) Uncorrected streaktubes (b) External streaktube correction (c) Internal streaktube correction

Figure 6: Correction methodology applied to a two-dimensional computational cell. For the external streaktube correction, the vertices highlighted with the circles are translated to set the external streaktube volume (blue). This correction also modifies the internal streaktubes (gray) since they share the highlighted vertices. The internal streaktube correction adds additional simplices (thick outlines) to correct streaktube volumes.

For external faces the simplices shown in Fig. 5a partition the sub-cell streaktubes. The proposed correction modifies the aggregate streaktube volume by translating p_{14} normal to the cell face. Translating p_{14} modifies the volume of 4 simplices: $S_{1,5}$, $S_{2,5}$, $S_{3,5}$, and $S_{4,5}$ listed in Table 1. The volume of these 4 simplices needed to make the streaktube solenoidal is computed using

$$\mathcal{V}_{\text{sims}} = \mathcal{V}_{\text{sol}} - \mathcal{V}\left(\widehat{\Omega_i}\right), \tag{18}$$

where $\mathcal{V}\left(\widehat{\Omega_i}\right)$ is the volume of the streaktube excluding simplices $S_{1,5}$, $S_{2,5}$, $S_{3,5}$, and $S_{4,5}$, i.e.,

$$\mathcal{V}\left(\widehat{\Omega_i}\right) = \sum_{f=1}^{4}\sum_{s=1}^{4}\mathcal{V}(S_{f,s}), \tag{19}$$

where f sums over the sub-faces and s sums over the simplices. Combining the corrected volume of the 4 simplices (Eq. 18) with the volume of a simplex (Eq. 16) leads to a relation which provides the normal component of p_{14} and can be written as

$$
\begin{aligned}
p_{14,n} = &- (6\mathcal{V}_{\text{sims}} + p_{11,n}p_{13,t_1}p_{14,t_2} - p_{11,n}p_{13,t_2}p_{14,t_1} - p_{11,t_1}p_{13,n}p_{14,t_2} + p_{11,t_2}p_{13,n}p_{14,t_1} \\
&+ p_{11,n}p_{14,t_1}p_{15,t_2} - p_{11,n}p_{14,t_2}p_{15,t_1} + p_{11,t_1}p_{14,t_2}p_{15,n} - p_{11,t_2}p_{14,t_1}p_{15,n} - p_{13,n}p_{14,t_1}p_{17,t_2} \\
&+ p_{13,n}p_{14,t_2}p_{17,t_1} - p_{13,t_1}p_{14,t_2}p_{17,n} + p_{13,t_2}p_{14,t_1}p_{17,n} + p_{14,t_1}p_{15,n}p_{17,t_2} - p_{14,t_1}p_{15,t_2}p_{17,n} \\
&- p_{14,t_2}p_{15,n}p_{17,t_1} + p_{14,t_2}p_{15,t_1}p_{17,n} - p_{11,n}p_{13,t_1}p_{5,t_2} + p_{11,n}p_{13,t_2}p_{5,t_1} + p_{11,t_1}p_{13,n}p_{5,t_2} \\
&- p_{11,t_1}p_{13,t_2}p_{5,n} - p_{11,t_2}p_{13,n}p_{5,t_1} + p_{11,t_2}p_{13,t_1}p_{5,n} + p_{11,n}p_{15,t_1}p_{5,t_2} - p_{11,n}p_{15,t_2}p_{5,t_1} \\
&- p_{11,t_1}p_{15,n}p_{5,t_2} + p_{11,t_1}p_{15,t_2}p_{5,n} + p_{11,t_2}p_{15,n}p_{5,t_1} - p_{11,t_2}p_{15,t_1}p_{5,n} - p_{13,n}p_{17,t_1}p_{5,t_2} \\
&+ p_{13,n}p_{17,t_2}p_{5,t_1} + p_{13,t_1}p_{17,n}p_{5,t_2} - p_{13,t_1}p_{17,t_2}p_{5,n} - p_{13,t_2}p_{17,n}p_{5,t_1} + p_{13,t_2}p_{17,t_1}p_{5,n} \\
&+ p_{15,n}p_{17,t_1}p_{5,t_2} - p_{15,n}p_{17,t_2}p_{5,t_1} - p_{15,t_1}p_{17,n}p_{5,t_2} + p_{15,t_1}p_{17,t_2}p_{5,n} + p_{15,t_2}p_{17,n}p_{5,t_1} \\
&- p_{15,t_2}p_{17,t_1}p_{5,n})/(p_{11,t_1}p_{13,t_2} - p_{11,t_2}p_{13,t_1} - p_{11,t_1}p_{15,t_2} + p_{11,t_2}p_{15,t_1} + p_{13,t_1}p_{17,t_2} \\
&- p_{13,t_2}p_{17,t_1} - p_{15,t_1}p_{17,t_2} + p_{15,t_2}p_{17,t_1})
\end{aligned}
\tag{20}
$$

where n, t_1, and t_2 are the normal and tangential components of each point with respect to the cell face. The equation ensures the volume of the simplices that depend on p_{14} matches $\mathcal{V}_{\text{sims}}$. It is well-posed for positive and negative signed volumes and simply moves p_{14} in the normal direction until the entire streaktube has a volume that satisfies the solenoidal condition. The previous equation was computed using MATLAB and the code is provided in Appendix A.1.

This correction enforces the volume of the streaktube to discretely match the solenoidal flux volume. The correction modifies the face of the streaktube that is not shared with neighboring streaktubes and ensures no gaps or overlaps are introduced between external streaktubes. However, the correction does translate points that are shared with internal streaktubes. For example, in Fig. 6, when the highlighted points are moved to correct the external streaktubes, the internal streaktubes are also modified. Similar shared points exist in three-dimensions and in Fig. 7 point A is p_{14} on external streaktube Ω_{y_j} and p_4 on internal streaktube $\Omega_{x_{m,i}}$. Since p_{14} on Ω_{y_j} was translated in the correction of the external streaktube, p_4 on $\Omega_{x_{m,i}}$ must also be displaced so the two points remain coincident. Table 2 provides a list of points on all internal streaktubes that need to be displaced and the point on an external streaktube they need to be coincident with. Practically, this can be achieved by first computing fluxes on all external faces and storing the location of the translated p_{14} points, then computing fluxes on all internal faces including the stored displacement of shared points.

Table 2: Points on internal streaktubes that are coincident with translated p_{14} on external streaktubes (see Fig. 5).

x_m streaktube	y_m streaktube	z_m streaktube
p_2 on $\Omega_{x_{m,i}}$ = p_{14} on Ω_{z_k}	p_2 on $\Omega_{y_{m,j}}$ = p_{14} on Ω_{x_i}	p_2 on $\Omega_{z_{m,k}}$ = p_{14} on Ω_{y_j}
p_4 on $\Omega_{x_{m,i}}$ = p_{14} on Ω_{y_j}	p_4 on $\Omega_{y_{m,j}}$ = p_{14} on Ω_{z_k}	p_4 on $\Omega_{z_{m,k}}$ = p_{14} on Ω_{x_i}
p_6 on $\Omega_{x_{m,i}}$ = p_{14} on $\Omega_{y_{j+1}}$	p_6 on $\Omega_{y_{m,j}}$ = p_{14} on $\Omega_{z_{k+1}}$	p_6 on $\Omega_{z_{m,k}}$ = p_{14} on $\Omega_{x_{i+1}}$
p_8 on $\Omega_{x_{m,i}}$ = p_{14} on $\Omega_{z_{k+1}}$	p_8 on $\Omega_{y_{m,j}}$ = p_{14} on $\Omega_{x_{i+1}}$	p_8 on $\Omega_{z_{m,k}}$ = p_{14} on $\Omega_{y_{j+1}}$

3.5.2. Internal Streaktube Correction

Modifying the internal streaktubes to uphold the solenoidal condition could be done by rescaling the internal flux volumes in a similar strategy we propose for external streaktubes. This approach would shift p_5, shown in Fig. 5b. However, all of the internal streaktubes share p_5 and moving it for one streaktube requires the same displacement for the other two internal streaktubes. A unique location of the shared p_5 is not always available to create solenoidal fluxes for all internal faces. Alternatively, we propose to modify each internal sub-face streaktube by adding additional simplices with the volume necessary for the correction. The simplices are added on the face of the streaktube that is not shared with neighboring streaktubes and

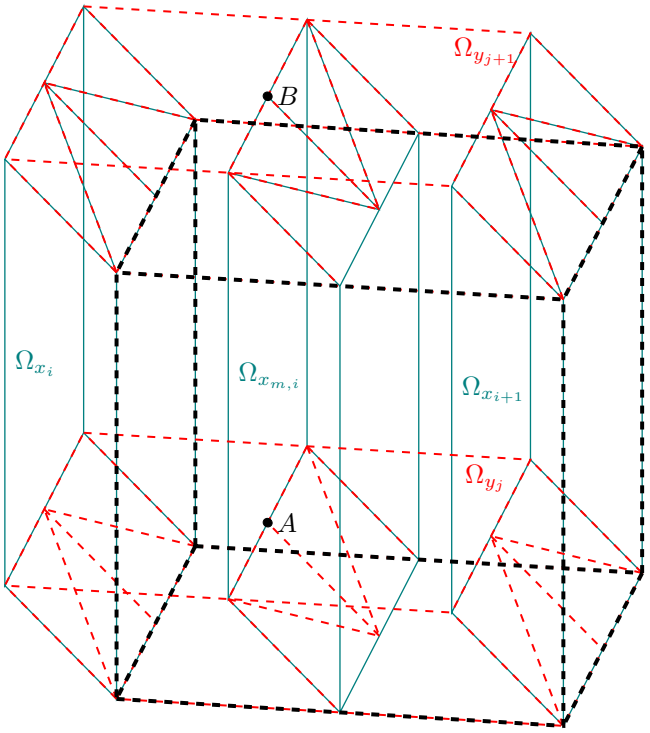


Figure 7: Streaktubes at x_i, $x_{m,i}$, and x_{i+1} are shown with teal solid lines. Streaktubes at y_j and y_{j+1} are shown with red dashed lines. Point A is shared between $\Omega_{x_{m,i}}$ and Ω_{y_j}. Point B is shared between $\Omega_{x_{m,i}}$ and $\Omega_{y_{j+1}}$.

does not introduce gaps or overlaps between streaktubes. Figure 6 shows a two-dimensional example of how simplices are added to internal streaktubes to correct their volume. In three dimensions, two additional simplices are added to each internal streaktube according to Fig. 8 and Table 3.

Table 3: Correction simplices added to those in Table 1 to enforce solenoidal criteria on internal streaktubes.

Sub-face 1					Sub-face 2					Sub-face 3					Sub-face 4				
$S_{1,6}$	1	2	5	19	$S_{2,6}$	2	3	5	20	$S_{3,6}$	4	5	7	21	$S_{4,6}$	5	6	9	22
$S_{1,7}$	1	5	4	19	$S_{2,7}$	3	6	5	20	$S_{3,7}$	5	8	7	21	$S_{4,7}$	5	9	8	22

Enforcing solenoidal fluxes for each sub-cell requires computing sub-cell solenoidal flux velocities. These velocities are found by enforcing the solenoidal condition on each sub-cell and minimizes the correction using a constrained least squares method. The proposed approach follows closely ideas of Cervone et al. [26].

The divergence of the 8 sub-cells within the i, j, k computational cell (Fig. 4) can be written as

$$
\begin{aligned}
D_{1,i,j,k} &= (u_{1,i,j,k} - u_{2,i\quad,j,k})\Delta y\Delta z + (v_{1,i,j,k} - v_{3,i,j\quad,k})\Delta x\Delta z + (w_{1,i,j,k} - w_{5,i,j,k\quad})\Delta x\Delta y,\\
D_{2,i,j,k} &= (u_{2,i,j,k} - u_{1,i+1,j,k})\Delta y\Delta z + (v_{2,i,j,k} - v_{4,i,j\quad,k})\Delta x\Delta z + (w_{2,i,j,k} - w_{6,i,j,k\quad})\Delta x\Delta y,\\
D_{3,i,j,k} &= (u_{3,i,j,k} - u_{4,i\quad,j,k})\Delta y\Delta z + (v_{3,i,j,k} - v_{1,i,j+1,k})\Delta x\Delta z + (w_{3,i,j,k} - w_{7,i,j,k\quad})\Delta x\Delta y,\\
D_{4,i,j,k} &= (u_{4,i,j,k} - u_{3,i+1,j,k})\Delta y\Delta z + (v_{4,i,j,k} - v_{2,i,j+1,k})\Delta x\Delta z + (w_{4,i,j,k} - w_{8,i,j,k\quad})\Delta x\Delta y,\\
D_{5,i,j,k} &= (u_{5,i,j,k} - u_{6,i\quad,j,k})\Delta y\Delta z + (v_{5,i,j,k} - v_{7,i,j\quad,k})\Delta x\Delta z + (w_{5,i,j,k} - w_{1,i,j,k+1})\Delta x\Delta y,\\
D_{6,i,j,k} &= (u_{6,i,j,k} - u_{5,i+1,j,k})\Delta y\Delta z + (v_{6,i,j,k} - v_{8,i,j\quad,k})\Delta x\Delta z + (w_{6,i,j,k} - w_{2,i,j,k+1})\Delta x\Delta y,\\
D_{7,i,j,k} &= (u_{7,i,j,k} - u_{8,i\quad,j,k})\Delta y\Delta z + (v_{7,i,j,k} - v_{5,i,j+1,k})\Delta x\Delta z + (w_{7,i,j,k} - w_{3,i,j,k+1})\Delta x\Delta y,\\
D_{8,i,j,k} &= (u_{8,i,j,k} - u_{7,i+1,j,k})\Delta y\Delta z + (v_{8,i,j,k} - v_{6,i,j+1,k})\Delta x\Delta z + (w_{8,i,j,k} - w_{4,i,j,k+1})\Delta x\Delta y,
\end{aligned}
\tag{21}
$$

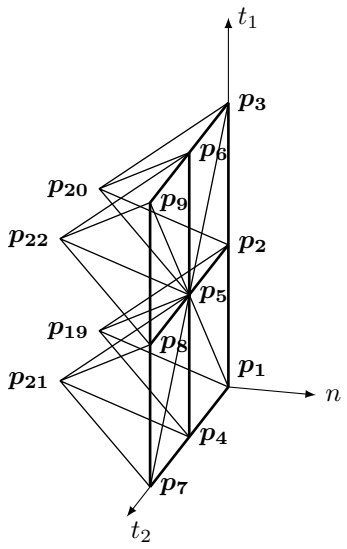


Figure 8: Additional correction simplices added to internal streaktubes to uphold solenoidal criteria. p_1 to p_9 are the same vertices shown in Fig. 5b.

where Δx, Δy, and Δz are the dimensions of the sub-cells. These sub-cell divergences are related to the divergence of the entire computational cell through

$$\mathcal{D}^{\text{cell}}_{i,j,k} = D_{1,i,j,k} + D_{2,i,j,k} + D_{3,i,j,k} + D_{4,i,j,k} + D_{5,i,j,k} + D_{6,i,j,k} + D_{7,i,j,k} + D_{8,i,j,k}. \tag{22}$$

Therefore, only seven of the eight equalities in Eq. 21 are independent and provide 7 constraints for the 12 unknown internal sub-cell velocities

$$\boldsymbol{U} = [u_2, u_4, u_6, u_8, v_3, v_4, v_7, v_8, w_5, w_6, w_7, w_8.]^{\mathsf{T}}, \tag{23}$$

where the subscript i, j, k that appears on each unknown velocity was not written to aid readability. The velocities on external faces in Eq. 21 are computed using Eq. 11 and the volume of the corrected streaktubes on external faces.

The divergence of the computational cell ($\mathcal{D}^{\text{cell}}_{i,j,k}$) is not identically zero in realistic numerical simulations. This divergence error is evenly distributed to the 8 sub-cells and we write the 7 constrains as

$$D_{n,i,j,k} = \frac{1}{8} D^{\text{cell}}_{i,j,k} \tag{24}$$

for $n = 1, \ldots, 7$.

This underdetermined system of 7 equations with 12 unknowns is solved with a least squares formulation that also minimizes changes to internal sub-cell velocities weighted by their associated areas. Weighting by the areas greatly simplifies the final equations and leads to velocities wherein the modification to fluxes is minimized. Using area weighted velocities is the main difference between our approach and the method proposed by Cervone et al. [26]. The constrained least squares problem minimizes the functional

$$\begin{aligned} Q =& \Big((u_2 - \widehat{u}_2)^2 + (u_4 - \widehat{u}_4)^2 + (u_6 - \widehat{u}_6)^2 + (u_8 - \widehat{u}_8)^2 \Big) (\Delta y \Delta z)^2 \\ &+ \Big((v_3 - \widehat{v}_3)^2 + (v_4 - \widehat{v}_4)^2 + (v_7 - \widehat{v}_7)^2 + (v_8 - \widehat{v}_8)^2 \Big) (\Delta x \Delta z)^2 \\ &+ \Big((w_5 - \widehat{w}_5)^2 + (w_6 - \widehat{w}_6)^2 + (w_7 - \widehat{w}_7)^2 + (w_8 - \widehat{w}_8)^2 \Big) (\Delta x \Delta y)^2 \\ &+ g_1 \big(D_1 - \mathcal{D}^{\text{cell}}/8 \big) + g_2 \big(D_2 - \mathcal{D}^{\text{cell}}/8 \big) + g_3 \big(D_3 - \mathcal{D}^{\text{cell}}/8 \big) + g_4 \big(D_4 - \mathcal{D}^{\text{cell}}/8 \big) \\ &+ g_5 \big(D_5 - \mathcal{D}^{\text{cell}}/8 \big) + g_6 \big(D_6 - \mathcal{D}^{\text{cell}}/8 \big) + g_7 \big(D_7 - \mathcal{D}^{\text{cell}}/8 \big). \end{aligned} \tag{25}$$

The velocity components with hats are the internal velocities computed using Eq. 11 from the pre-corrected streaktubes. The subscripts i, j, k on the functional Q, the sub-cell divergences D, and velocity components u, v, and w are omitted to aid readability.

The least squares minimum is found by taking derivatives of Q with respect to each of the unknown velocities and g_n for $n = 1, \ldots, 7$ and setting the 19 derivatives equal to zero. Solving the system of equations leads to a closed form solution for the unknowns $\boldsymbol{U}$ that can be written as

$$\begin{aligned} \boldsymbol{A}\boldsymbol{U} &= \boldsymbol{M}\boldsymbol{b} \\ \boldsymbol{U} &= \boldsymbol{A}^{-1}\boldsymbol{M}\boldsymbol{b} \end{aligned} \tag{26}$$

where

$$\boldsymbol{A} = \mathrm{diag}[\Delta y\Delta z, \Delta y\Delta z, \Delta y\Delta z, \Delta y\Delta z, \Delta x\Delta z, \Delta x\Delta z, \Delta x\Delta z, \Delta x\Delta z, \Delta x\Delta y, \Delta x\Delta y, \Delta x\Delta y, \Delta x\Delta y]$$

$$\boldsymbol{M}^{\mathsf{T}} = \frac{1}{24}\begin{bmatrix}
7 & 2 & 2 & 1 & 7 & 2 & 2 & 1 & 7 & 2 & 2 & 1 \\
10 & -4 & -4 & -2 & -5 & 5 & -1 & 1 & -5 & 5 & -1 & 1 \\
2 & 7 & 1 & 2 & -7 & -2 & -2 & -1 & 2 & 1 & 7 & 2 \\
-4 & 10 & -2 & -4 & 5 & -5 & 1 & -1 & -1 & 1 & -5 & 5 \\
2 & 1 & 7 & 2 & 2 & 1 & 7 & 2 & -7 & -2 & -2 & -1 \\
-4 & -2 & 10 & -4 & -1 & 1 & -5 & 5 & 5 & -5 & 1 & -1 \\
1 & 2 & 2 & 7 & -2 & -1 & -7 & -2 & -2 & -1 & -7 & -2 \\
-2 & -4 & -4 & 10 & 1 & -1 & 5 & -5 & 1 & -1 & 5 & -5 \\
7 & 2 & 2 & 1 & -2 & -7 & -1 & -2 & -2 & -7 & -1 & -2 \\
2 & 7 & 1 & 2 & 2 & 7 & 1 & 2 & -1 & -2 & -2 & -7 \\
2 & 1 & 7 & 2 & -1 & -2 & -2 & -7 & 2 & 7 & 1 & 2 \\
1 & 2 & 2 & 7 & 1 & 2 & 2 & 7 & 1 & 2 & 2 & 7 \\
7 & 2 & 2 & 1 & 7 & 2 & 2 & 1 & 7 & 2 & 2 & 1 \\
-7 & -2 & -2 & -1 & 2 & 7 & 1 & 2 & 2 & 7 & 1 & 2 \\
-5 & 5 & -1 & 1 & 10 & -4 & -4 & -2 & -5 & -1 & 5 & 1 \\
5 & -5 & 1 & -1 & -4 & 10 & -2 & -4 & -1 & -5 & 1 & 5 \\
2 & 1 & 7 & 2 & 2 & 1 & 7 & 2 & -7 & -2 & -2 & -1 \\
-2 & -1 & -7 & -2 & 1 & 2 & 2 & 7 & -2 & -7 & -1 & -2 \\
-1 & 1 & -5 & 5 & -4 & -2 & 10 & -4 & 5 & 1 & -5 & -1 \\
1 & -1 & 5 & -5 & -2 & -4 & -4 & 10 & 1 & 5 & -1 & -5 \\
-2 & -7 & -1 & -2 & 7 & 2 & 2 & 1 & -2 & -1 & -7 & -2 \\
2 & 7 & 1 & 2 & 2 & 7 & 1 & 2 & -1 & -2 & -2 & -7 \\
-1 & -2 & -2 & -7 & 2 & 1 & 7 & 2 & 2 & 1 & 7 & 2 \\
1 & 2 & 2 & 7 & 1 & 2 & 2 & 7 & 1 & 2 & 2 & 7 \\
7 & 2 & 2 & 1 & 7 & 2 & 2 & 1 & 7 & 2 & 2 & 1 \\
-7 & -2 & -2 & -1 & 2 & 7 & 1 & 2 & 2 & 7 & 1 & 2 \\
2 & 7 & 1 & 2 & -7 & -2 & -2 & -1 & 2 & 1 & 7 & 2 \\
-2 & -7 & -1 & -2 & -2 & -7 & -1 & -2 & 1 & 2 & 2 & 7 \\
-5 & -1 & 5 & 1 & -5 & -1 & 5 & 1 & 10 & -4 & -4 & -2 \\
5 & 1 & -5 & -1 & -1 & -5 & 1 & 5 & -4 & 10 & -2 & -4 \\
-1 & -5 & 1 & 5 & 5 & 1 & -5 & -1 & -4 & -2 & 10 & -4 \\
1 & 5 & -1 & -5 & 1 & 5 & -1 & -5 & -2 & -4 & -4 & 10 \\
-2 & -1 & -7 & -2 & -2 & -1 & -7 & -2 & 7 & 2 & 2 & 1 \\
2 & 1 & 7 & 2 & -1 & -2 & -2 & -7 & 2 & 7 & 1 & 2 \\
-1 & -2 & -2 & -7 & 2 & 1 & 7 & 2 & 2 & 1 & 7 & 2 \\
1 & 2 & 2 & 7 & 1 & 2 & 2 & 7 & 1 & 2 & 2 & 7
\end{bmatrix} \qquad \boldsymbol{b} = \begin{bmatrix}
u_{1,i} \quad_{,j,k}\Delta y\Delta z \\
\widehat{u}_{2,i} \quad_{,j,k}\Delta y\Delta z \\
u_{3,i} \quad_{,j,k}\Delta y\Delta z \\
\widehat{u}_{4,i} \quad_{,j,k}\Delta y\Delta z \\
u_{5,i} \quad_{,j,k}\Delta y\Delta z \\
\widehat{u}_{6,i} \quad_{,j,k}\Delta y\Delta z \\
u_{7,i} \quad_{,j,k}\Delta y\Delta z \\
\widehat{u}_{8,i} \quad_{,j,k}\Delta y\Delta z \\
u_{1,i+1,j,k}\Delta y\Delta z \\
u_{3,i+1,j,k}\Delta y\Delta z \\
u_{5,i+1,j,k}\Delta y\Delta z \\
u_{7,i+1,j,k}\Delta y\Delta z \\
v_{1,i,j} \quad_{,k}\Delta x\Delta z \\
v_{2,i,j} \quad_{,k}\Delta x\Delta z \\
\widehat{v}_{3,i,j} \quad_{,k}\Delta x\Delta z \\
\widehat{v}_{4,i,j} \quad_{,k}\Delta x\Delta z \\
v_{5,i,j} \quad_{,k}\Delta x\Delta z \\
v_{6,i,j} \quad_{,k}\Delta x\Delta z \\
\widehat{v}_{7,i,j} \quad_{,k}\Delta x\Delta z \\
\widehat{v}_{8,i,j} \quad_{,k}\Delta x\Delta z \\
v_{1,i,j+1,k}\Delta x\Delta z \\
v_{2,i,j+1,k}\Delta x\Delta z \\
v_{5,i,j+1,k}\Delta x\Delta z \\
v_{6,i,j+1,k}\Delta x\Delta z \\
w_{1,i,j,k} \quad \Delta x\Delta y \\
w_{2,i,j,k} \quad \Delta x\Delta y \\
w_{3,i,j,k} \quad \Delta x\Delta y \\
w_{4,i,j,k} \quad \Delta x\Delta y \\
\widehat{w}_{5,i,j,k} \quad \Delta x\Delta y \\
\widehat{w}_{6,i,j,k} \quad \Delta x\Delta y \\
\widehat{w}_{7,i,j,k} \quad \Delta x\Delta y \\
\widehat{w}_{8,i,j,k} \quad \Delta x\Delta y \\
w_{1,i,j,k+1}\Delta x\Delta y \\
w_{2,i,j,k+1}\Delta x\Delta y \\
w_{3,i,j,k+1}\Delta x\Delta y \\
w_{4,i,j,k+1}\Delta x\Delta y
\end{bmatrix} \tag{27}$$

$\boldsymbol{A}$ is a diagonal matrix of the areas associated with each unknown velocity and removes the area weighting from the solution. $\boldsymbol{M}$ is a constant matrix that can be created in an initialization routine, note the factor $1/24$ that has been factored out of the matrix. $\boldsymbol{b}$ is a vector of all the known area-weighted velocities on the

exterior of the cell and the pre-corrected area-weighted velocities on the interior of the cell. $\boldsymbol{M}$ is computed with the MATLAB code provided in Appendix A.2

Solving Eq. 26 provides solenoidal flux velocities for each sub-cell within a given computational cell. These velocities are related to volumes by Eq. 11. The difference between the solenoidal volume $\mathcal{V}_{\text{sol}}$ and pre-corrected volume $\mathcal{V}\left(\widehat{\Omega_i}\right)$ provides the volume of the correction simplices on each sub-face, i.e., $\mathcal{V}_{\text{sims}} = \mathcal{V}_{\text{sol}} - \mathcal{V}\left(\widehat{\Omega_i}\right)$. With the correction volume and the volume of a simplex, Eq. 16, the location of points p_{19}, p_{20}, p_{21}, and p_{22} in Fig. 8 can be computed. The location of the points are computed using the relation

$$v_{5,t_1} = \frac{1}{4}(v_{1,t_1} + v_{2,t_1} + v_{4,t_1} + v_{5,t_1}) \tag{28}$$

$$v_{5,t_2} = \frac{1}{4}(v_{1,t_2} + v_{2,t_2} + v_{4,t_2} + v_{5,t_2}) \tag{29}$$

$$\begin{aligned} v_{5,n} = &- (6\mathcal{V}_{\text{sims}} - v_{1,n}v_{2,t_1}v_{4,t_2} + v_{1,n}v_{2,t_2}v_{4,t_1} + v_{1,t_1}v_{2,n}v_{4,t_2} - v_{1,t_1}v_{2,t_2}v_{4,n} - v_{1,t_2}v_{2,n}v_{4,t_1} \\ &+ v_{1,t_2}v_{2,t_1}v_{4,n} + v_{1,n}v_{2,t_1}v_{5,t_2} - v_{1,n}v_{2,t_2}v_{5,t_1} + v_{1,n}v_{3,t_1}v_{4,t_2} - v_{1,n}v_{3,t_2}v_{4,t_1} - v_{1,t_1}v_{2,n}v_{5,t_2} \\ &- v_{1,t_1}v_{3,n}v_{4,t_2} + v_{1,t_1}v_{3,t_2}v_{4,n} + v_{1,t_2}v_{2,n}v_{5,t_1} + v_{1,t_2}v_{3,n}v_{4,t_1} - v_{1,t_2}v_{3,t_1}v_{4,n} - v_{1,n}v_{3,t_1}v_{5,t_2} \\ &+ v_{1,n}v_{3,t_2}v_{5,t_1} + v_{1,t_1}v_{3,n}v_{5,t_2} - v_{1,t_2}v_{3,n}v_{5,t_1} + v_{2,n}v_{4,t_1}v_{5,t_2} - v_{2,n}v_{4,t_2}v_{5,t_1} - v_{2,t_1}v_{4,n}v_{5,t_2} \\ &+ v_{2,t_2}v_{4,n}v_{5,t_1} - v_{3,n}v_{4,t_1}v_{5,t_2} + v_{3,n}v_{4,t_2}v_{5,t_1} + v_{3,t_1}v_{4,n}v_{5,t_2} - v_{3,t_2}v_{4,n}v_{5,t_1}) \\ &/(v_{1,t_1}v_{2,t_2} - v_{1,t_2}v_{2,t_1} - v_{1,t_1}v_{3,t_2} + v_{1,t_2}v_{3,t_1} + v_{2,t_1}v_{4,t_2} - v_{2,t_2}v_{4,t_1} - v_{3,t_1}v_{4,t_2} + v_{3,t_2}v_{4,t_1}), \end{aligned} \tag{30}$$

which is defined using the vertices of the generalized correction simplices shown in Fig. 9. In the figure, v_1 to v_4 are on the original streaktube and v_5 is the added vertex used to define the correction simplices. For each sub-face the vertices take the coordinates of the appropriate points. Table 4 provides the definition of the vertices on each of the four sub-faces. This relation is computed using MATLAB and the code is provided in Appendix A.3. After the two correction simplices are added to the streaktube, solenoidal semi-Lagrangian fluxes can be computed using the cutting routines described in Section 3.2 on page 4.

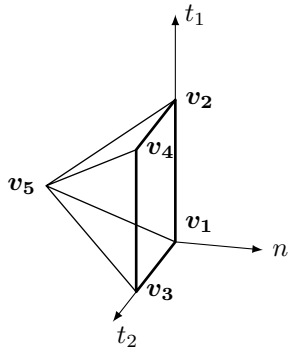


Figure 9: Additional correction simplices added on an internal face using the generalized points used in Eqs. 28 to 30.

Table 4: Definition of the vertices $[v_1, v_2, v_3, v_4, v_5]$ used in Eqs. 28 to 30 for the four sub-faces.

Sub-face	Points
1	$p_1,\ p_2,\ p_4,\ p_5,\ p_{19}$
2	$p_5,\ p_2,\ p_6,\ p_3,\ p_{20}$
3	$p_7,\ p_4,\ p_8,\ p_5,\ p_{21}$
4	$p_5,\ p_6,\ p_8,\ p_9,\ p_{22}$

3.5.3. Summary of Streaktube Discretization

The streaktubes are represented by a collection of simplices. On external faces of a pressure cell, the streaktube is represented with the 20 simplices shown in Fig. 5a. These simplices are corrected to be solenoidal by moving p_{14} using Eq. 20. Streaktubes associated with internal faces are represented with the 20 simplices shown in Fig. 5b. These simplices are first adjusted by moving the point that need to be coincident with the p_{14} points on the external streaktubes. Next, the internal streaktubes are corrected to be solenoidal by adding 8 correction simplices (2 per sub-face) as shown by Fig. 8. The volume of the correction is set to ensure each sub-cell is solenoidal by solving for solenoidal velocities with Eq. 26 and computing the coordinates of the correction simplices vertices using Eqs. 28 to 30.

3.6. Coupling with Other Discretizations

The proposed scheme uses semi-Lagrangian fluxes only near the phase interface where discontinuities exist. Away from the interface, the fluxes are evaluated using a second-order finite difference discretization. The two schemes are conservatively coupled by incorporating both discretizations into the calculation of sub-cell solenoidal flux velocities (Section 3.5.2).

Figure 10 shows a portion of a computational mesh, the phase interface, and the type of discretization (semi-Lagrangian or finite difference). Bands around the interface has been introduced to determine where to use semi-Lagrangian fluxes. The band is defined to be $\mathcal{B}_{i,j,k} = 0$ for each i, j, k computational cell that contains the phase-interface. Then the band is extended outward from the interface by incrementing neighboring cells in all directions. Fluxes within the bands $\mathcal{B} = 0$ and $\mathcal{B} = 1$ are computed using the semi-Lagrangian framework. Outside these bands and at cell faces between $\mathcal{B} = 1$ and $\mathcal{B} = 2$, the finite difference method is employed. Note that, for Courant-Friedrichs-Lewy (CFL) conditions larger than 1, the number of bands N_{band} that use the semi-Lagrangian discretization must be extended such that $N_{\text{band}} = \text{ceiling(CFL)}$.

For computational cells in the band $\mathcal{B} = 1$, the finite difference discretization is used on exterior faces between $\mathcal{B} = 1$ and $\mathcal{B} = 2$. This discretization is incorporated into the calculation of sub-cell solenoidal flux velocities to construct a consistent coupling. In Eq. 26, the velocities on external faces will be based on both finite difference and semi-Lagrangian discretizations. For semi-Lagrangian discretized sub-cell faces, the velocity is computed using Eq. 11. For the other sub-cell faces that are discretized with the finite difference scheme, the velocity is taken to be the cell face velocity defined at this location. For example, for computational cell i, j in Fig. 10, the velocity $u_{i,j}$ is used to construct the two finite difference fluxes on the left face and $v_{i,j+1}$ is used on the top face. With this coupling the finite difference and semi-Lagrangian discretizations are combined without hindering conservation.

The proposed coupling is not dependent on the finite difference discretization and could be extended to other discretizations (e.g., finite volume) to be used away from the interface. For any discretization scheme used away from the interface, it is important that the computational stencil does not extend across the phase interface and the discontinuities that may exist there. This can be achieved by extending the width of the band over which semi-Lagrangian fluxes are calculated. As long as the computed fluxes respect the solenoidal property and are incorporated into the flux corrections (Sec. 3.5.2) the coupling methodology should maintain the discrete conservation properties.

3.7. Constructing Second-Order Fluxes

The calculation of semi-Lagrangian fluxes requires computing integrals of conserved quantities over simplices. This operation appears in step 4 in Section 3.2, where $I_{\Delta_{\alpha,\beta,\gamma}} = \int_{\Delta_{\alpha,\beta,\gamma}} \psi \, \mathrm{d}V$ is computed using data local to the computation cell and phase in which simplex $\Delta_{\alpha,\beta,\gamma}$ exists.

This integral is clearly defined when $\psi = \alpha$, the liquid distribution function given by

$$\alpha(\boldsymbol{x}, t) = \begin{cases} 0 & \text{if } \boldsymbol{x} \text{ is in the gas phase,} \\ 1 & \text{if } \boldsymbol{x} \text{ is in the liquid phase.} \end{cases} \tag{31}$$

For this case the integral reduces to either 0 or the volume $\mathcal{V}(\Delta_{\alpha,\beta,\gamma})$ if the simplex is in the gas or liquid, respectively. Similarly, when density is transported, $\psi = \rho$ and the integral is either $\rho_g \mathcal{V}(\Delta_{\alpha,\beta,\gamma})$ or $\rho_l \mathcal{V}(\Delta_{\alpha,\beta,\gamma})$.

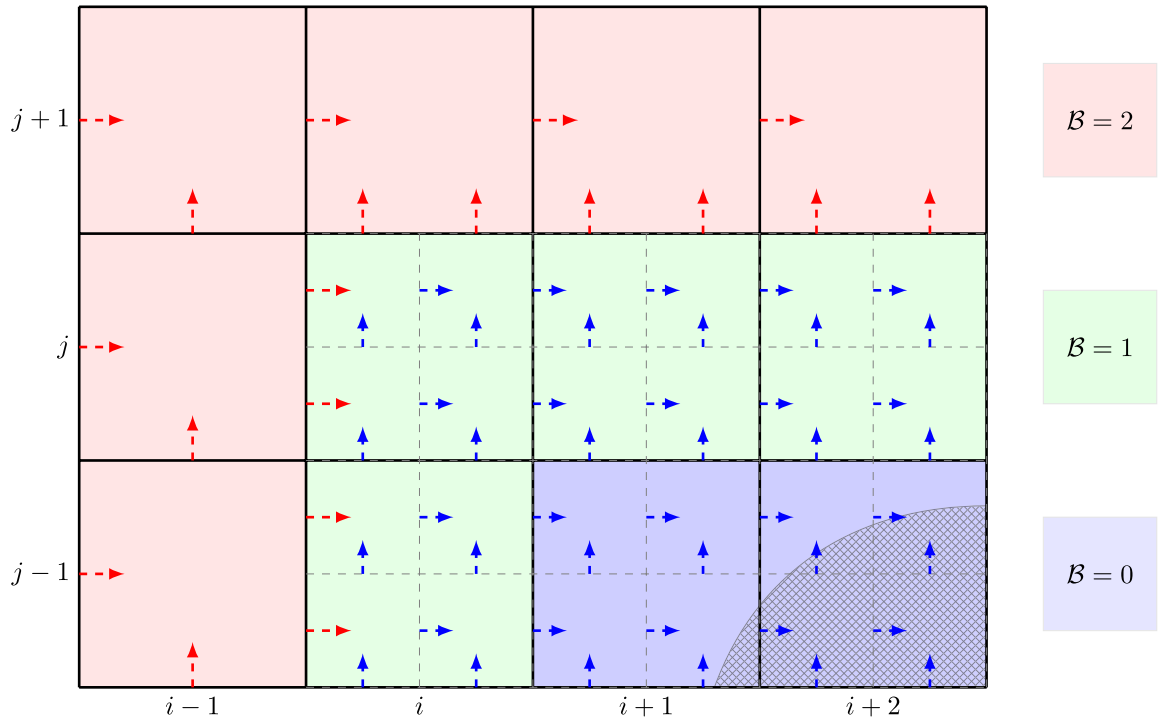


Figure 10: Portion of a two-dimensional computational mesh where coupling between semi-Lagrangian (solid blue arrows) and finite difference (dashed red arrows) fluxes is required. Bands $\mathcal{B} = 0, 1, 2$ are shown with blue, green, and red, respectively. Liquid is located in the bottom right corner and is indicated with cross-hatching.

For momentum and scalar fluxes with $\psi = \rho \boldsymbol{u}$ and $\psi = f$, respectively, the integral can be defined multiple ways. The ambiguity is in the definition of the velocity or scalar which can be taken as a constant within each computational cell or more accurately computed with a higher-order reconstruction. Following the ideas of Le Chenadec and Pitsch [20], we use a second-order, conservative method with a linear reconstruction that does not alter the magnitude of the conserved quantity within each cell. For example, the x component of velocity is reconstructed within a u-momentum cell using

$$u^r_{i,j,k}(\boldsymbol{x}, t) = u_{i,j,k} + \nabla u \cdot (\boldsymbol{x} - \overline{\boldsymbol{x}}) \tag{32}$$

with

$$\overline{\boldsymbol{x}}_{i,j,k} = \frac{\int_{CV_{u,i,j,k}} \rho(\boldsymbol{x}, t) \boldsymbol{x} \, \mathrm{d}V}{\int_{CV_{u,i,j,k}} \rho(\boldsymbol{x}, t) \, \mathrm{d}V}, \tag{33}$$

which is the center of mass of the i, j, k^{th} u-momentum cell. Similar reconstructions are used for the other velocity components v and w. In the equation, ∇u is computed using a combination of forward, backward, and central finite difference operators. For example, the gradient of u is computed using

$$\begin{aligned}
\frac{\partial u}{\partial x} &\approx \min_{\text{signed}} \left(\frac{u_{i+1,j,k} - u_{i,j,k}}{\Delta x}, \frac{u_{i,j,k} - u_{i-1,j,k}}{\Delta x}, \frac{u_{i+1,j,k} - u_{i-1,j,k}}{2\Delta x} \right) \\
\frac{\partial u}{\partial y} &\approx \min_{\text{signed}} \left(\frac{u_{i,j+1,k} - u_{i,j,k}}{\Delta y}, \frac{u_{i,j,k} - u_{i,j-1,k}}{\Delta y}, \frac{u_{i,j+1,k} - u_{i,j-1,k}}{2\Delta y} \right) \\
\frac{\partial u}{\partial z} &\approx \min_{\text{signed}} \left(\frac{u_{i,j,k+1} - u_{i,j,k}}{\Delta z}, \frac{u_{i,j,k} - u_{i,j,k-1}}{\Delta z}, \frac{u_{i,j,k+1} - u_{i,j,k-1}}{2\Delta z} \right)
\end{aligned} \tag{34}$$

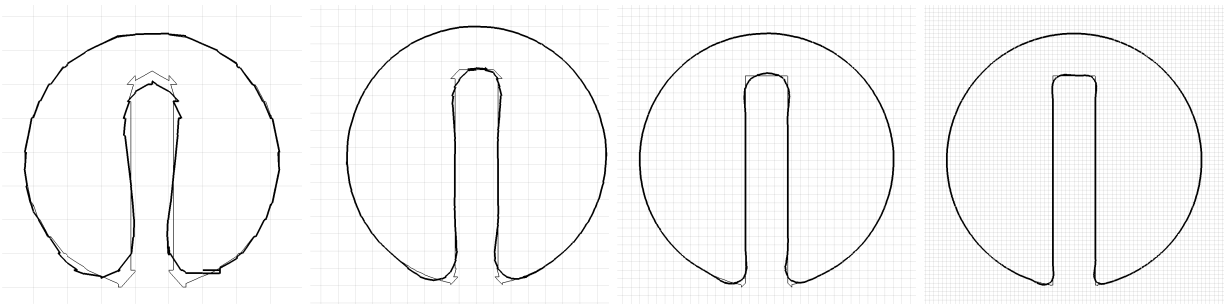
where the signed min provides the minimum of the three gradients ignoring the sign. This expression provides an estimate for the gradient that was found to remain stable for the simulations presented in this paper. The integral in Eq. 33 is computed using computational geometry routines that provide a center of mass that is consistent with the PLIC interface reconstruction to machine precision [17]. The density ρ is set by the PLIC reconstruction and the velocity within the u-momentum cell, respectively to uphold conservation. Similar expressions are used for the gradient the other velocity components v and w and for scalars.

This formulation provides a velocity and scalar reconstruction that maintain mass, momentum, and scalar conservation. Furthermore, the reconstruction can easily be incorporated into the flux calculations (Step 4 in Section 3.2) since the integral of a linear function over a simplex is the volume of the simplex multiplied by the reconstruction evaluated at the simplex barycenter.

4. Verification Tests

We investigate the properties of the proposed scheme by performing a series of verification tests. Namely, we analyze the method in the transport of a notched circle (Zalesak's Disk), two- and three-dimensional deformation tests, transport in a random solenoidal velocity field, and momentum convection. The method performs similarly to other unsplit, geometric, semi-Lagrangian schemes [16, 17] on these tests with the additional benefit of conserving mass and momentum to machine precision.

4.1. Zalesak's Disk



(a) $N_p = 25^2$ (50^2) (b) $N_p = 50^2$ (100^2) (c) $N_p = 100^2$ (200^2) (d) $N_p = 200^2$ (400^2)

Figure 11: Effect of mesh size on Zalesak's disk test case. Figures show the PLIC at the beginning and end of the simulation with the thin and thick lines, respectively.

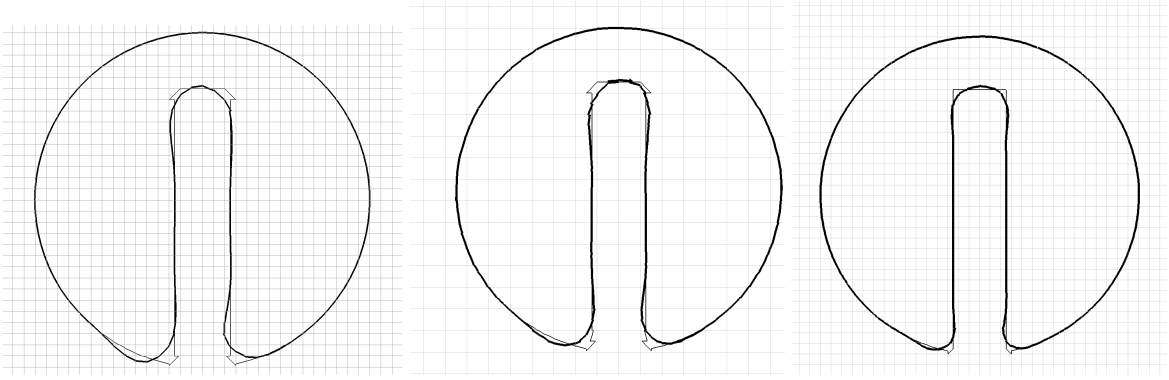
This case tests the ability of the proposed scheme to transport a notched disk by a rotational vortex [27]. The notched disk has diameter 0.3, notch width of 0.05, initially centered at $(x, y) = (0, 0.25)$ within a square domain $[-0.5, 0.5]^2$. The velocity is specified as

$$\begin{aligned} u &= -2\pi y, \\ v &= +2\pi x, \end{aligned} \tag{35}$$

which should not modify the shape of the disk as it rotates about the origin. The disk is rotated for one revolution on various meshes. Fig. 11 shows the final shape of the transported disk. The figures (and subsequent figures with interfaces) show the PLIC representation of the phase interface. The results are reported with respect to the mesh resolution N_p, i.e., the number of grid points on the velocity mesh. The α mesh, which is twice as fine, is reported in parentheses. Even on the coarsest mesh $N_p = 25^2$ (50^2) the shape of the disk resembles the initial condition and the notch remains. On the finer meshes the shape remains very close to the initial condition.

The results are minimally impacted by using a coarse mesh for the velocity and a fine mesh for α. This is apparent by comparing results obtain with the method of Owkes and Desjardins [17], which is similar

with the exception that the same mesh is employed for the velocity and α. Figure 12 shows a comparison for Zalesak's disk wherein similar results are obtained when the α mesh resolution is matched to the mesh used by Owkes and Desjardins [17].



(a) Owkes [17]: $N_p = 100^2$ (b) Proposed: $N_p = 50^2$ (100^2) (c) Proposed: $N_p = 100^2$ (200^2)

Figure 12: Comparison of Zalesak's disk computing with the proposed scheme and the method of Owkes and Desjardins [17]. Figures show the PLIC at the beginning and end of the simulation with the thin and thick lines, respectively.

To assess the results quantitatively, we define the following errors [14, 15, 17]

$$E_{\text{shape}}(t) = \sum_{n=1}^{N_{\text{CV}}} |\alpha_n(t) - \alpha_n^e(t)| \mathcal{V}_n, \tag{36}$$

$$E_{\text{mass}}(t) = \left| \sum_{n=1}^{N_{\text{CV}}} \alpha_n(t)\mathcal{V}_n - \sum_{n=1}^{N_{\text{CV}}} \alpha_n^e(t)\mathcal{V}_n \right|, \quad \text{and} \tag{37}$$

$$E_{\text{bound}}(t) = \max\left(- \min_{n=1,\ldots,N_{\text{CV}}} \alpha_n(t)\mathcal{V}_n, \max_{n=1,\ldots,N_{\text{CV}}} (\alpha_n(t) - 1)\mathcal{V}_n \right), \tag{38}$$

where $\alpha_n(t)$, $\alpha_n^e(t)$ and $\mathcal{V}_n$ are the liquid volume fraction, exact liquid volume fraction, and volume in the n^{th} computational cell at time t. The liquid volume fractions are defined using Eq. 12 with $\psi = \alpha$.

Table 5: Error norms and timing data at the end of the Zalesak's disk simulations.

N_p		E_{shape}	E_{mass}	E_{bound}	Time/Timestep (s) Proposed	Time/Timestep (s) Owkes [17]
25^2	(50^2)	3.976e-03	2.567e-15	2.038e-18	0.060	0.07
50^2	(100^2)	1.177e-03	4.614e-14	2.835e-18	0.046	0.16
100^2	(200^2)	5.290e-04	4.531e-13	1.385e-18	0.095	0.31
200^2	(400^2)	2.020e-04	3.666e-12	7.488e-19	0.251	0.66
400^2	(800^2)	8.077e-05	3.643e-13	3.775e-19	0.455	1.47
800^2	(1600^2)	3.351e-05	1.295e-11	2.095e-19	2.009	3.38

Table 5 provides the errors at different mesh resolutions. The mass and boundedness errors are near machine precision as expected. The shape errors are small even on the coarse meshes. Figure 13 provides the shape errors versus N_p and shows that first-order convergence is achieved. This convergence rate is lower than the expected second-order, however it is consistent with similar schemes [17] and is attributed to transporting sharp corners that are not well represented by the PLIC interface reconstruction.

Table 5 also shows timing data. The simulations used one compute node containing dual 8-core Intel Xeon E5-2650 v2 2.6 GHz processors with 64 GB of RAM. The reported metric is the time per time-step

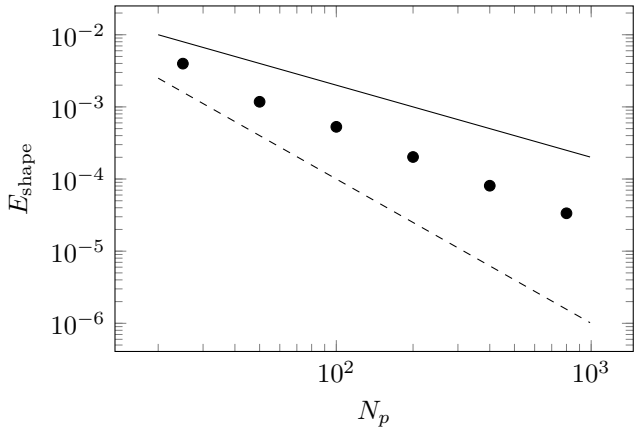


Figure 13: Convergence of the shape error E_{shape} at the end of the Zalesaks disk simulations. Solid and dashed lines show first- and second-order convergence, respectively.

averaged throughout the simulation. Even with the finer mesh for α, the timing data is similar in magnitude to Owkes and Desjardins [17]. The differences in timing can be attributed to a few things including 1) the additional fluxes required in the proposed scheme that increase the cost, 2) the consistent coupling with finite difference fluxes reduces the number of faces with semi-Lagrangian fluxes and reduces the cost, and 3) the numerical algorithms in the computational geometry toolbox have been optimized to minimize the number of planes used in the recursive cutting procedure, improving the efficiency of the code.

A theoretically comparison of cost of the semi-Lagrangian scheme can be obtained by comparing the number of simplices that must be cut by the mesh and gas-liquid interface using the computational geometry routines. The proposed numerical method transports density, momentum, and scalars using 20 simplices for each external face and 28 simplices on internal faces. Each cell is associated with 3 external fluxes and 3 internal fluxes for a total of 144 simplices. The method developed by Owkes and Desjardins [17] applied to both density and momentum computes fluxes independently for quantities stored at different locations on the staggered grid for a total of 4 different sets of fluxes. The scheme uses 8 simplices to compute the flux on each of the 3 faces associated with a computational cell. In total, 96 simplices are used to compute the required fluxes. Therefore, the proposed scheme should be 50% more expensive than the non-conservative formulation, however the differences in the bands and the number of semi-Lagrangian fluxes and the differences in the computational geometry toolbox lead to a departure from this theoretical prediction.

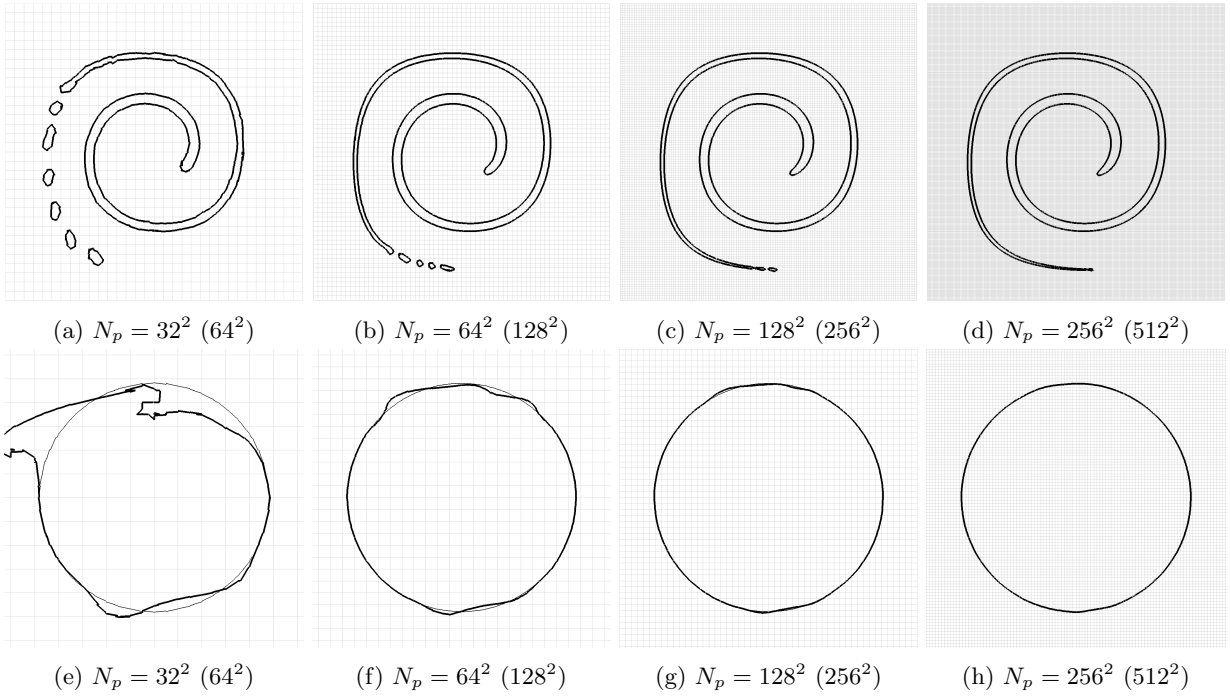
Note that only a three-dimensional implementation of the proposed method was developed. Zalesak's disk and the other two-dimensional tests are computed using a one-cell thick three-dimensional domain. This setup is more computationally expensive than an optimized two-dimensional implementation due to extra simplices needed to discretize each three-dimensional flux volume. However, since most simulations will be three-dimensional it was chosen to focus on creating an optimized three-dimensional implementation.

4.2. Two-Dimensional Deformation

This test case, proposed by Leveque [28], assesses the ability of the proposed scheme to transport under-resolved interface features such as ligaments. The test consists of stretching and un-stretching a disk in a vortex. A disk of diameter 0.3 is initially centered at $(x, y) = (0, 0.25)$ within a unit square domain $[-0.5, 0.5]^2$. The disk is deformed by the velocity field

$$\begin{aligned} u &= -2\sin^2(\pi x)\sin(\pi y)\cos(\pi y)\cos(\pi t/8), \\ v &= +2\sin^2(\pi y)\sin(\pi x)\cos(\pi x)\cos(\pi t/8), \end{aligned} \tag{39}$$

which completes one period (stretch/un-stretch cycle) in 8 time units.



(a) $N_p = 32^2$ (64^2) (b) $N_p = 64^2$ (128^2) (c) $N_p = 128^2$ (256^2) (d) $N_p = 256^2$ (512^2)

(e) $N_p = 32^2$ (64^2) (f) $N_p = 64^2$ (128^2) (g) $N_p = 128^2$ (256^2) (h) $N_p = 256^2$ (512^2)

Figure 14: Effect of mesh size on two-dimensional deformation test case. The top images show the disk at maximum deformation. The bottom images show the result at the end of the simulation with the thick and thin lines showing the computed and exact solution, respectively.

Figure 14 shows the PLIC interface at maximum stretching ($t = 4$) and at the end of the simulation ($t = 8$) when the disk should return to its initial state. With coarse meshes, the tail of the stretched disk becomes under-resolved and breaks into a series of droplets. This is consistent with other schemes that have good mass conservation properties [23]. Furthermore the results are similar to those obtained with other unsplit geometric semi-Lagrangian schemes [16, 17]. This is true when the mesh that resolves α (finer mesh) is used for the comparison. Consequently, resolving the velocity on a coarser mesh does not significantly impact the interface dynamics. The interface at the end of the simulations shows good agreement with the exact solution, especially on the finer meshes.

Table 6 shows the shape, mass, and boundedness errors. For all the different meshes, the mass and boundedness errors remain near machine precision. The shape error shows second-order convergence as shown by Fig. 15. Additionally, timing data is provided for the simulations which were each performed on one compute node.

Table 6: Error norms and timing data at the end of the simulations for the two-dimensional deformation simulations.

N_p		E_{shape}	E_{mass}	E_{bound}	Time/Timestep (s) Proposed	Time/Timestep (s) Owkes [17]
32^2	(64^2)	1.041e-02	2.912e-14	1.965e-18	0.110	-
64^2	(128^2)	1.344e-03	1.787e-14	2.846e-18	0.173	0.01
128^2	(256^2)	3.491e-04	8.132e-15	1.603e-18	0.376	0.04
256^2	(512^2)	8.941e-05	9.966e-14	1.028e-18	1.360	0.15
512^2	(1024^2)	3.503e-05	7.963e-14	4.821e-19	1.368	0.62
1024^2	(2048^2)	1.430e-05	1.544e-13	2.483e-19	2.908	2.41

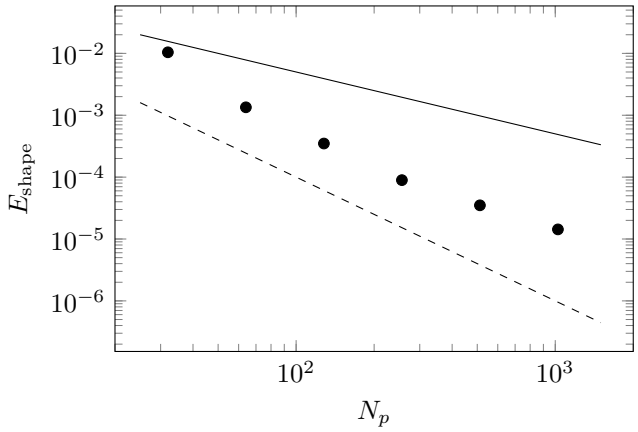


Figure 15: Convergence of the E_{shape} error at the end of the simulation for the two-dimensional deformation test case. Solid and dashed lines show first- and second-order convergence, respectively.

4.3. Three-Dimensional Deformation

This test case was also proposed by Leveque [28] and assesses the scheme's ability to transport an under-resolved three-dimensional interface. The simulation consists of the stretching and un-stretching of a droplet by a vortex. The droplet is initialized with diameter 0.3 centered at $(x, y, z) = (0.35, 0.35, 0.35)$ within a cubic domain of $[0, 1]^3$. The droplet is deformed by the velocity field

$$\begin{aligned} u &= 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos(\pi t/3), \\ v &= -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)\cos(\pi t/3), \\ w &= -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos(\pi t/3). \end{aligned} \tag{40}$$

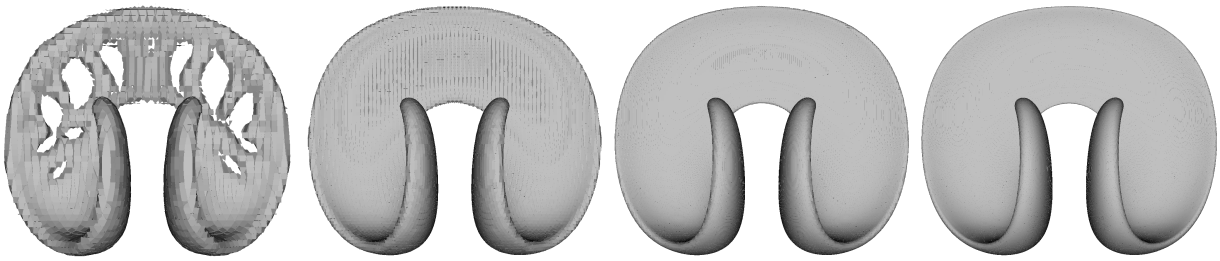
With this velocity the droplet is stretched until $t = 1.5$ and un-stretched until $t = 3$.

Figure 16 shows the PLIC representation of the gas-liquid interface at maximum stretching and at the end of the simulation when the initial droplet should be recovered. At maximum stretching a thin liquid sheet is formed that breaks into ligaments on the coarse meshes. As the mesh is refined, this sheet is maintained. Discrepancies in the final shape occur due to the break-up of the liquid sheet on the coarse meshes and the discrepancies reduce with increasing mesh resolution. Furthermore, in the figures, all PLIC surfaces are shown and the simulations do not produce "floatsam" and "jetsam", which are common in many VOF schemes [8].

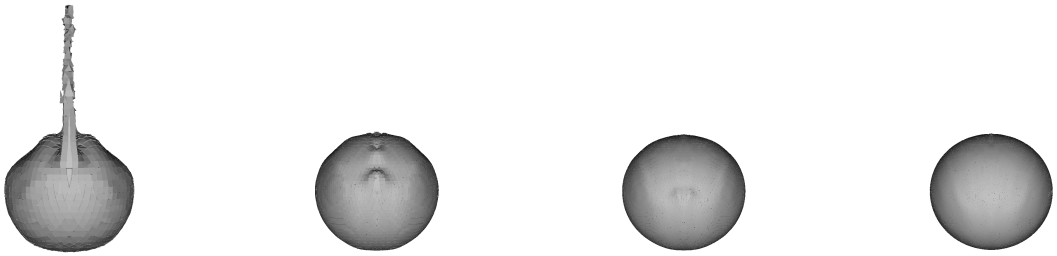
Table 7 provides the errors and timing results. Similar to the other cases, the mass and boundedness errors are near machine precision for all the simulations. The shape error is small and converges with a second-order rate, see Fig. 17. Timing data shows the three-dimensional simulations, which were performed on four compute nodes, have similar, albeit slightly larger, computational costs to previously reported times [17].

Table 7: Error norms at the end of the simulations and timing data for the transport of three-dimensional deformation simulations.

N_p		E_{shape}	E_{mass}	E_{bound}	Time/Timestep (s) Proposed	Owkes [17]
32^2	(64^2)	2.305e-03	3.073e-14	1.820e-17	3.822	0.78
64^2	(128^2)	5.675e-04	6.011e-14	1.112e-16	9.799	2.85
128^2	(256^2)	1.372e-04	9.327e-14	3.332e-20	26.636	12.2
256^2	(512^2)	4.802e-05	4.783e-15	7.013e-19	79.337	45.5



(a) $N_p = 32^2$ (64^2) (b) $N_p = 64^2$ (128^2) (c) $N_p = 128^2$ (256^2) (d) $N_p = 256^2$ (512^2)



(e) $N_p = 32^2$ (64^2) (f) $N_p = 64^2$ (128^2) (g) $N_p = 128^2$ (256^2) (h) $N_p = 256^2$ (512^2)

Figure 16: Effect of mesh size on two-dimensional deformation test case. The top and bottom images show the disk at maximum deformation ($t = 1.5$) and at the end of the simulation ($t = 3$), respectively.

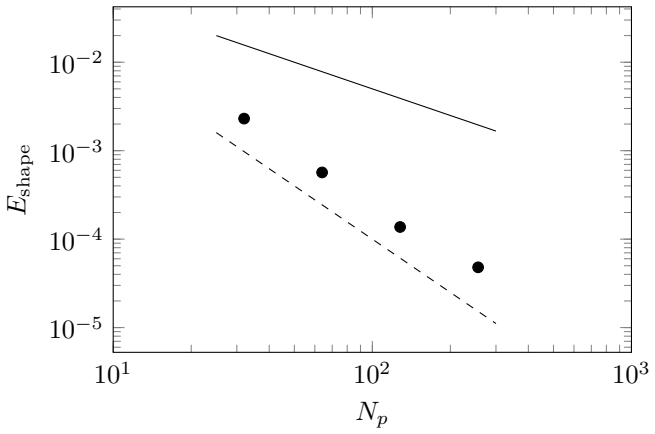
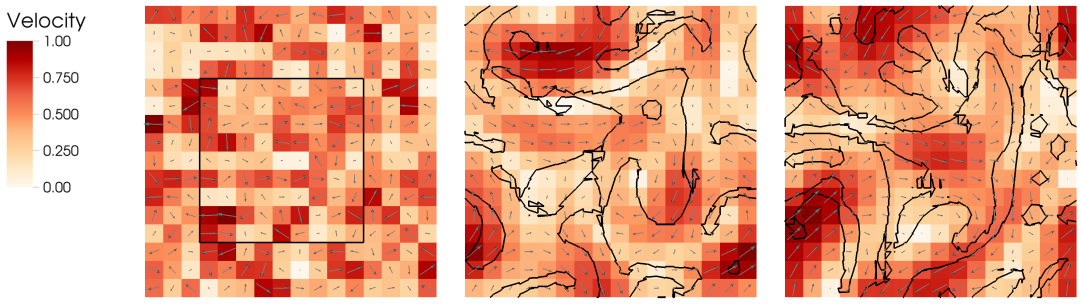


Figure 17: Convergence of the E_{shape} error at the end of the simulation for the three-dimensional deformation test case. Solid and dashed lines show first- and second-order convergence, respectively.

4.4. Conservation in a Random Solenoidal Inviscid Velocity Field

The proposed scheme should conserve mass and momentum to machine precision. To test this property, transport in a random solenoidal velocity field without surface tension or viscous dissipation is modeled. The solenoidal velocity field is computed from a stream function constructed from normally distributed random numbers [29]. The test uses a $2\pi \times 2\pi$ domain discretized with $N_x \times N_y = 16 \times 16$ grid cells for the flow solver mesh. The interface is initialized as a square of size $\pi \times \pi$ located in the center of the domain. Two tests are performed with density ratios of $\rho_l/\rho_g = 1$ and 1000.

Simulations are performed with a non-conservative implementation and the proposed conservative implementation to assess the importance of discrete consistency and conservation. The non-conservative scheme is based on Owkes & Desjardins [17] and uses semi-Lagrangian fluxes to transport both density and momentum. The scheme remains stable at large density ratios by computing density fluxes that are consistent with the momentum fluxes [30, 31]. These density fluxes are computed for each momentum cell to allow for a stable velocity to be evaluated, but are not consistent with the density fluxes used to transport the VOF representation of the phase interface stored at the cell center. It is precisely this inconsistency in density fluxes that is rectified with the introductions of the sub-cells in the proposed numerical framework and leads to discrete conservation of both mass and momentum.



(a) $tu/\Delta x = 0$ (b) $tu/\Delta x = 5$ (c) $tu/\Delta x = 10$

Figure 18: Velocity field (colormap and vectors) and interface (solid line) at different non-dimensional times for the random solenoidal velocity field test. Simulation performed with $\rho_l/\rho_g = 1$ and the proposed conservative implementation.

Figure 18 shows the temporal evolution of the velocity field and phase interface computed with the conservative scheme and a unity density ratio. Figure 19 shows the results computed at both density ratios and with both implementations at a non-dimensional time $tu/\Delta x = 10$. The figures show the interface undergoes significant deformation within the random flow and the sub-cells used in the conservative implementation capture smaller interface features.

Conservation errors are introduced and defined for a generic conserved quantify χ as

$$E_\chi(t) = \sum_{n=1}^{N_{\mathrm{CV}}} \chi_n(t)\mathcal{V}_n - \int_D \chi^e(\boldsymbol{x}, t)\,\mathrm{d}V, \tag{41}$$

where D is the computational domain and χ^e is the exact solution. Figure 20 shows the conservation errors for density and momentum over the simulated time for the two density ratios (top and bottom) and both implementations (left and right). For the unity density ratio cases the errors remain near machine precisions for both implementations. At the larger density ratio $\rho_l/\rho_g = 1000$, the non-conservative scheme is hindered by very large momentum conservation errors. The conservative implementation maintains errors close to machine precision, albeit slightly larger than the unity density ratio case due to the larger round-off errors associated with the momentum flux calculation.

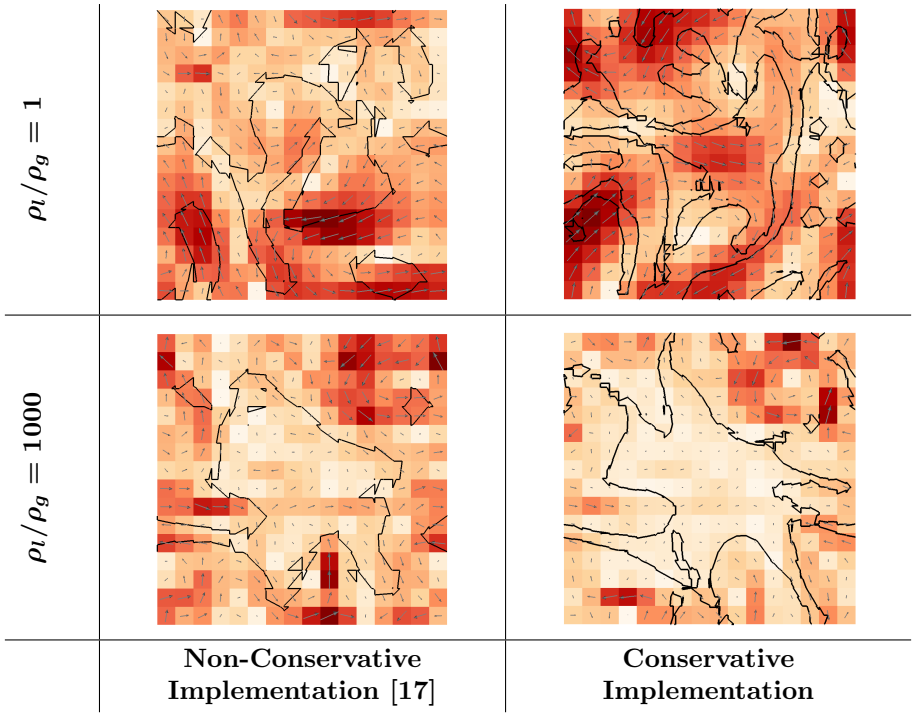


Figure 19: Velocity field (colormap and vectors) and interface (solid line) at different non-dimensional times for the random solenoidal velocity field test. Results shown at $tu/\Delta x = 10$ for both density ratios and using the non-conservative and conservative implementations.

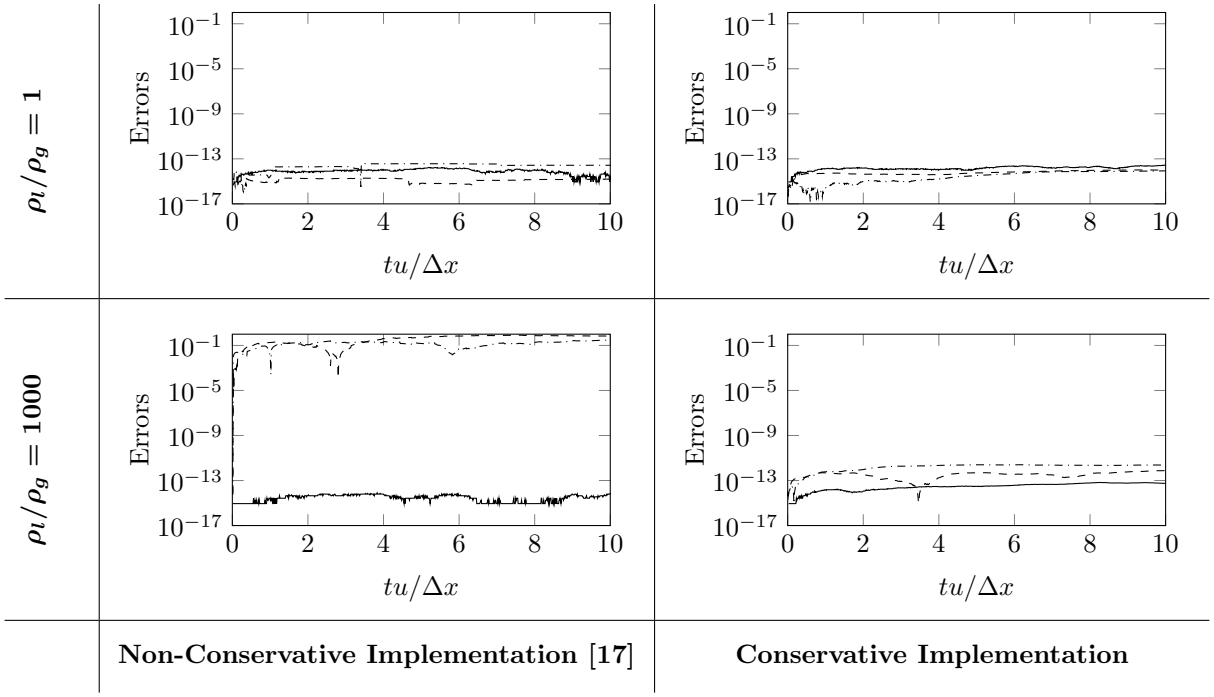


Figure 20: Conservation errors for the random solenoidal inviscid velocity test case with varying density ratios and implementations. Errors versus time are provided for density ρ (solid) and the two components of momentum ρu and ρv (dashed and dash-dotted).

4.5. Momentum Transport Test

This test case studies the accuracy of the semi-Lagrangian fluxes to convect momentum by solving the Euler equations. The setup follows previous works [32, 33, 34] and consists of a unit square domain $[-0.5, 0.5]^2$ with initial velocity field

$$u(x, y) = 1 - 2\cos(2\pi x)\sin(2\pi y) \quad \text{and} \tag{42}$$
$$v(x, y) = 1 + 2\sin(2\pi x)\cos(2\pi y). \tag{43}$$

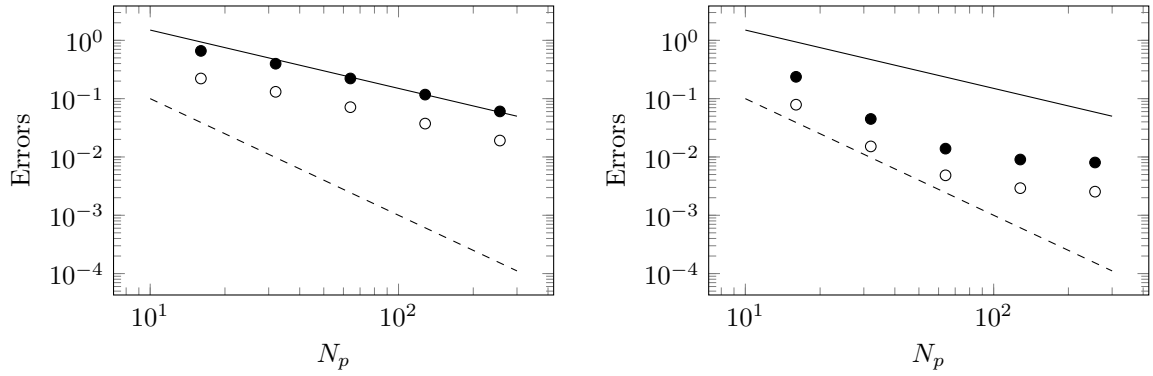
For these initial conditions, the exact solution is

$$u(x, y, t) = 1 - 2\cos(2\pi(x - t))\sin(2\pi(y - t)) \quad \text{and} \tag{44}$$
$$v(x, y, t) = 1 + 2\sin(2\pi(x - t))\cos(2\pi(y - t)). \tag{45}$$

The proposed semi-Lagrangian fluxes is used to compute all of the fluxes in the domain to test the accuracy of the approach. The mesh is refined and the accuracy is accessed through the L_2 and L_∞ errors defined for a general variable χ as

$$L_2(\chi) = \frac{\sqrt{\sum_{n=1}^{N_{\mathrm{CV}}} (\chi_n - \chi_n^e)^2}}{\sqrt{\sum_{n=1}^{N_{\mathrm{CV}}} (\chi_n^e)^2}} \quad \text{and} \tag{46}$$
$$L_\infty(\chi) = \max|\chi_n - \chi_n^e|. \tag{47}$$

Simulations are performed with CFL=0.75 to a time $t = 0.5$. Convergence results for the x component of momentum are shown in Fig. 21a. First-order convergence is demonstrated, which is the expected result for a second-order semi-Lagrangian scheme that loses an order due to accumulated temporal errors. To further test the method, the timestep was kept constant at a value obtained by setting CFL=0.75 on a mesh with $N_p = 2048$. With the smaller timestep, temporal errors a significantly reduced and the dominant errors come from the spatial discretization. Figure 21b shows that second-order convergence is achieved. On the finer meshes, the temporal errors dominate and further spatial refinement does not reduce the error.



(a) CFL=0.75, Timestep varies with N_p

(b) Timestep Fixed at $0.75\Delta t/2048$

Figure 21: Convergence of errors for the momentum transport test. $L_2(\rho u)$ and $L_\infty(\rho u)$ errors shown with closed and open circles, respectively. Solid and dashed lines show first- and second-order convergence, respectively.

5. Validation Tests

In this section the proposed scheme is used to simulate realistic flows including rising bubbles and an atomizing liquid jet under the influence of electrohydrodynamic (EHD) forces.

5.1. Rising Bubble

This test consists of bubbles rising in viscous liquids. Comparisons are made on bubble shape and bubble rise velocity against experiments by Bhaga and Weber [35]. Two simulations are conducted and Table 8 provides the Reynolds, Eötvös, and Morton numbers for each case defined as

$$\text{Reynolds:} \quad \text{Re} = \frac{\rho D U}{\mu}, \tag{48}$$

$$\text{Eötvös:} \quad \text{Eö} = \frac{g D^2 \rho}{\sigma}, \tag{49}$$

$$\text{Morton:} \quad \text{Mo} = \frac{g \mu^4}{\rho \sigma^3}. \tag{50}$$

where D is the bubble diameter. The simulations are performed on a domain of $L_x \times L_y \times L_z = 8D \times 30D \times 8D$ using a flow solver mesh of $N_x \times N_y \times N_z = 64 \times 240 \times 64$.

Figure 22 shows the steady-state shape of the bubbles. Profiles of the bubbles computed with the proposed scheme are overlaid on images from Bhaga and Weber [35]. The shapes compare well although it is difficult to determine the interface profile in the recirculation region at the bottom of the bubble from the experimental results. Bubble terminal velocity was computed and compared against the empirical correlation by Angelino [36]:

$$v_\text{t} = K \mathcal{V}^m \tag{51}$$

with

$$K = \frac{25}{1 + 0.33\text{M}^{0.29}} \tag{52}$$

$$m = 0.167\big(1 + 0.34\text{M}^{0.24}\big) \tag{53}$$

provided by Bhaga and Weber [35]. In the previous equations, v_t is the terminal velocity in cm/s and $\mathcal{V}$ is the volume of the bubble in cm^3. This correlation is valid for large bubbles or when $E > 40$ and $R > 2$. Table 9 provides the terminal velocities from the simulations and the correlation.

The proposed scheme is expected to conserve mass and momentum to machine precision. For this test case, mass is expected to be conserved and the liquid volume fraction should remain bounded. Momentum will not be conserved due to the gravitational body force. Table 10 provides the mass conservation and boundedness errors for the test cases. Near machine precision is obtained for all quantities.

Table 8: Non-dimensional numbers for the rising bubble test cases.

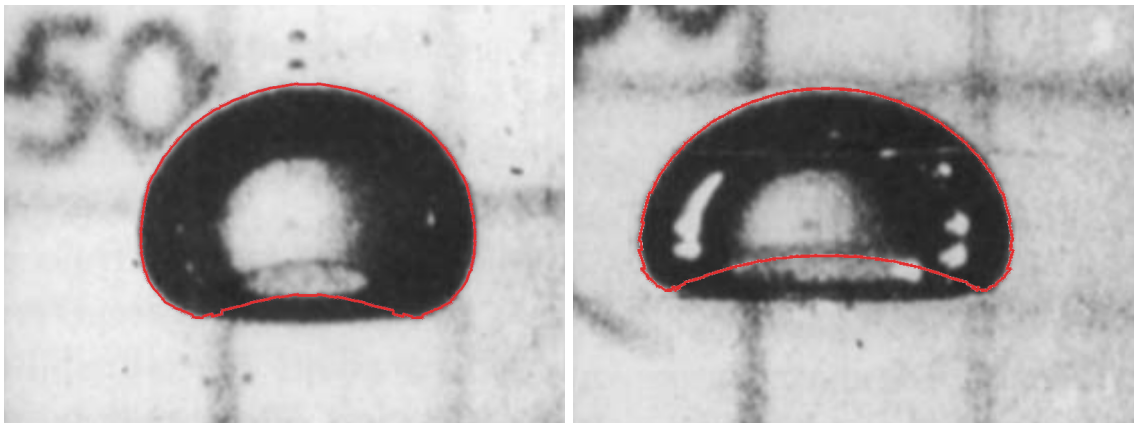
Case	Reynolds	Eötvös	Morton
1	2.47	116	848
2	7.16	116	41.1

Table 9: Terminal velocity of rising bubbles.

	Terminal velocity (m/s)	
Case	Proposed scheme	Correlation [36]
1	0.175	0.181
2	0.251	0.230

5.2. Electrohydrodynamic Atomization

The atomization of a charged liquid jet is used to validate the proposed method for transporting a complex interface topology. The simulation consists of injecting a dielectric liquid into quiescent air. The liquid is



(a) Case 1 (b) Case 2

Figure 22: Comparison of bubble shapes. Experimental results of Bhaga and Weber [35] overlaid with bubble outlines computed with proposed scheme.

Table 10: Mass and boundedness errors for the rising bubble test cases.

Quantity	$\max\|E_{\text{mass}}(t)\|$	$\max\|E_{\text{bound}}(t)\|$
Case 1	1.20e-16	1.42e-13
Case 2	5.33e-16	1.50e-13

injected with a uniform electric charge distribution and a turbulent velocity computed with a preliminary periodic pipe flow simulation. The charged jet forms a self-precipitating electric field that produces the enhanced atomization through the Coulomb force. The simulation is performed with a Reynolds number of 4000, Weber number of 1700, Electric Reynolds of 28.8, and Electric Peclet number of 17,000, density ratio of 664, viscosity ratio of 51, and electric permittivity ratio of 2.2. The simulation used a flow solver mesh of $N_x \times N_y \times N_z = 512 \times 256 \times 256$. Additional details of the simulation setup and methodology are provided in Sheehy and Owkes [37]. Here we do not study that atomization process, but rather highlight the capability of the proposed scheme to simulate such flows.

Figure 23 provides an snapshot of the PLIC interface representation. The interface shows many ligaments, droplets, and corrugations. The maximum errors of mass, boundedness, and conservation of electric charge density q are shown in Table 11. The maximum is taken over all time. As expected, all the errors remain small and close to machine precision.

Table 11: Mass, boundedness, and conservation of charge density errors for the electrohydrodynamic atomization test case.

Quantity	$\max\|E_{\text{mass}}(t)\|$	$\max\|E_{\text{bound}}(t)\|$	$\max\|E_q(t)\|$
Value	9.82e-11	4.98e-12	3.71e-15

6. Conclusions

A numerical framework for transporting gas-liquid phase interfaces is developed that conserves mass, momentum, and scalars to near machine precision. The framework leverages unsplit, geometric, semi-Lagrangian fluxes to transport a VOF representation of the phase interface, momentum, and any scalars. Conservation is achieved through constructing consistent fluxes of all conserved quantities. A staggered computational mesh is employed and consistent fluxes are computed using a twice as fine mesh for α, the liquid volume fraction.

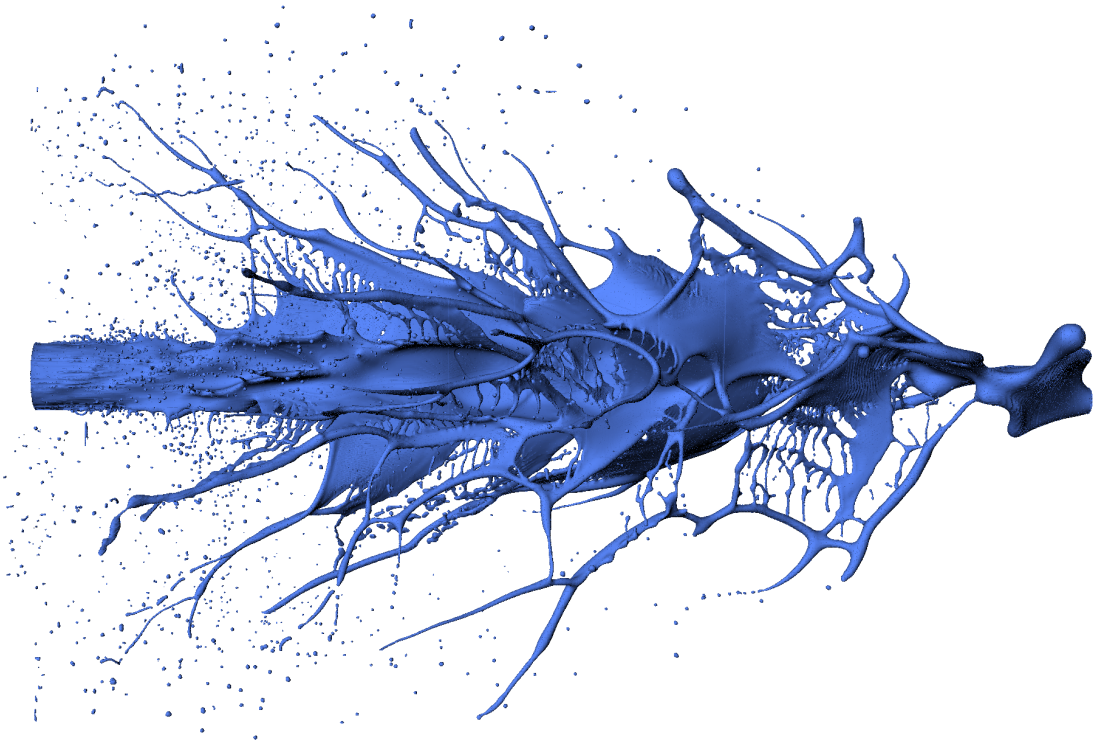


Figure 23: Electrohydrodynamic assisted atomizing jet. PLIC representation of gas-liquid interface is shown.

The proposed scheme was applied to a number of canonical verification tests including Zalesak's disk, two- and three-dimensional deformation problems, motion within a random velocity field, and convection of momentum. For all the tests, mass and momentum were conserved to near machine precision. Additionally, the computational cost of the proposed method is comparable to previously published results for other unsplit, geometric, semi-Lagrangian schemes, even though previous schemes did not employ a finer mesh for α.

Finally, the method was used to simulate a rising bubble and an atomizing liquid jet under electrohydrodynamic effects and conservation was demonstrated. These simulation showcase the ability of the proposed scheme to accurately predict realistic flows.

7. Acknowledgements

The authors acknowledge support through computational resources by the Research Computing Group at Montana State University and the Department of Defenses' High Performance Computing Modernization Program (HPCMP).

Visualizations in this article were created using the VisIt visualizing and analysis tool. VisIt is supported by the Department of Energy with funding from the Advanced Simulation and Computing Program and the Scientific Discovery through Advanced Computing Program [38].

References

[1] M. Rudman, A volume-tracking method for incompressible multifluid flows with large density variations, International Journal for Numerical Methods in Fluids 28 (2) (1998) 357–378.
[2] W. Dettmer, P. H. Saksono, D. Peri\'c, On a finite element formulation for incompressible Newtonian fluid flows on moving domains in the presence of surface tension, Commun. Numer. Meth. En. 19 (9) (2003) 659–668.
[3] S. Osher, J. A. Sethian, Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations, J. Comp. Phys. 79 (1) (1988) 12–49.
[4] C. Hirt, B. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, Journal of Computational Physics 39 (1) (1981) 201–225.
[5] R. DeBar, Fundamentals of the KRAKEN code, Tech. Rep. UCIR-760, LLNL (1974).
[6] B. Nichols, C. Hirt, Methods for Calculating Multi-Dimensional, Transient Free Surface Flows Past Bodies, Tech. Rep. LA-UR-75-1932, Los Alamos National Laboratory (1975).
[7] W. F. Noh, P. Woodward, SLIC (Simple Line Interface Calculation), in: Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics, no. 59 in Lecture Notes in Physics, Springer Berlin Heidelberg, 1976, pp. 330–340.
[8] G. Tryggvason, R. Scardovelli, S. Zaleski, Direct Numerical Simulations of Gas-Liquid Multiphase Flows, Cambridge University Press, 2011.
[9] D. Youngs, Time-dependent multi-material flow with large fluid distortion, Numerical Methods for Fluid Dynamics (1982) 273–285.
[10] W. J. Rider, D. B. Kothe, Reconstructing Volume Tracking, Journal of Computational Physics 141 (2) (1998) 112–152.
[11] J. E. Pilliod, E. G. Puckett, Second-order accurate volume-of-fluid algorithms for tracking material interfaces, Journal of Computational Physics 199 (2) (2004) 465–502.
[12] D. J. Harvie, D. F. Fletcher, A New Volume of Fluid Advection Algorithm: The Stream Scheme, J. Comp. Phys. 162 (1) (2000) 1–32.
[13] D. J. E. Harvie, D. F. Fletcher, A new volume of fluid advection algorithm: the defined donating region scheme, International Journal for Numerical Methods in Fluids 35 (2) (2001) 151–172.
[14] J. Lpez, J. Hernndez, P. Gmez, F. Faura, A volume of fluid method based on multidimensional advection and spline interface reconstruction, Journal of Computational Physics 195 (2) (2004) 718–742.
[15] J. Hernndez, J. Lpez, P. Gmez, C. Zanzi, F. Faura, A new volume of fluid method in three dimensions - {P}art {I}: {M}ultidimensional advection method with face-matched flux polyhedra, International Journal for Numerical Methods in Fluids 58 (8) (2008) 897–921.
[16] V. Le Chenadec, H. Pitsch, A {3D} Unsplit Forward/Backward Volume-of-Fluid Approach and Coupling to the Level Set Method, Journal of Computational Physics 233 (15) (2013) 10–33.
[17] M. Owkes, O. Desjardins, A computational framework for conservative, three-dimensional, unsplit, geometric transport with application to the volume-of-fluid (VOF) method, Journal of Computational Physics 270 (1) (2014) 587–612.
[18] Q. Zhang, On a Family of Unsplit Advection Algorithms for Volume-of-Fluid Methods, SIAM Journal on Numerical Analysis 51 (5) (2013) 2822–2850.
[19] Q. Zhang, On generalized donating regions: Classifying Lagrangian fluxing particles through a fixed curve in the plane, Journal of Mathematical Analysis and Applications 424 (2) (2015) 861–877.

[20] V. Le Chenadec, H. Pitsch, A monotonicity preserving conservative sharp interface flow solver for high density ratio two-phase flows, Journal of Computational Physics 249 (2013) 185–203.
[21] O. Desjardins, G. Blanquart, G. Balarac, H. Pitsch, High order conservative finite difference scheme for variable density low Mach number turbulent flows, Journal of Computational Physics 227 (15) (2008) 7125–7159.
[22] H. Choi, P. Moin, Effects of the Computational Time Step on Numerical Solutions of Turbulent Flow, J. Comput. Phys. 113 (1) (1994) 1–4.
[23] O. Desjardins, V. Moureau, H. Pitsch, An accurate conservative level set/ghost fluid method for simulating turbulent atomization, Journal of Computational Physics 227 (18) (2008) 8395–8416.
[24] P. Liovic, M. Rudman, J.-L. Liow, D. Lakehal, D. Kothe, A 3d unsplit-advection volume tracking algorithm with planarity-preserving interface reconstruction, Computers & Fluids 35 (10) (2006) 1011–1032.
[25] J. Mencinger, I. un, A PLICVOF method suited for adaptive moving grids, Journal of Computational Physics 230 (3) (2011) 644–663.
[26] A. Cervone, S. Manservisi, R. Scardovelli, An optimal constrained approach for divergence-free velocity interpolation and multilevel VOF method, Computers & Fluids 47 (1) (2011) 101–114.
[27] S. T. Zalesak, Fully multidimensional flux-corrected transport algorithms for fluids, J. Comput. Phys. 31 (3) (1979) 335–362.
[28] R. J. Leveque, High-Resolution Conservative Algorithms for Advection in Incompressible Flow, SIAM Journal on Numerical Analysis 33 (2) (1996) 627–665, articleType: research-article / Full publication date: Apr., 1996 / Copyright 1996 Society for Industrial and Applied Mathematics.
[29] Y. Morinishi, T. S. Lund, O. V. Vasilyev, P. Moin, Fully Conservative Higher Order Finite Difference Schemes for Incompressible Flow, Journal of Computational Physics 143 (1) (1998) 90–124.
[30] O. Desjardins, V. Moureau, Methods for multiphase flows with high density ratio, Center for Turbulence Research Proceedings of the Summer Program (2010) 313.
[31] O. Desjardins, J. McCaslin, M. Owkes, P. Brady, Direct numerical and large-eddy simulation of primary atomization in complex geometries, Atomization and Sprays 23 (11) (2013) 1001–1048.
[32] M. L. Minion, A Projection Method for Locally Refined Grids, Journal of Computational Physics 127 (1) (1996) 158–178.
[33] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, M. L. Welcome, A Conservative Adaptive Projection Method for the Variable Density Incompressible NavierStokes Equations, Journal of Computational Physics 142 (1) (1998) 1–46.
[34] S. Popinet, Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries, Journal of Computational Physics 190 (2) (2003) 572–600.
[35] D. Bhaga, M. E. Weber, Bubbles in viscous liquids: shapes, wakes and velocities, Journal of Fluid Mechanics 105 (1981) 61–85.
[36] H. Angelino, Hydrodynamique des grosses bulles dans les liquides visqueux, Chemical Engineering Science 21 (6) (1966) 541–550.
[37] P. Sheehy, M. Owkes, Numerical Study of Electric Reynolds Number on Electrohydrodynamic (EHD) Assisted Atomization, Atomization and Sprays (under review).
[38] H. Childs, E. Brugger, B. Whitlock, J. Meredith, S. Ahern, D. Pugmire, K. Biagas, M. Miller, C. Harrison, G. H. Weber, H. Krishnan, T. Fogal, A. Sanderson, C. Garth, E. W. Bethel, D. Camp, O. Rbel, M. Durant, J. M. Favre, P. Navrtil, VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data, in: High Performance VisualizationEnabling Extreme-Scale Scientific Insight, 2012, pp. 357–372.

Appendix A. MATLAB Codes Used to Compute Equations and Matrices

Appendix A.1. Correction Simplices for External Streaktubes

This MATLAB code computes Eq. 20 used in the adjustment of the streaktube associated with external faces.

```
% Code to find coordinates of point needed to construct correction tets
clear; clc

% Create symbolic points
p5 =sym('p5_%d' ,[1  3]); % [p5_n,  p1_t1,  p1_t2]
p11=sym('p11_%d',[1  3]); % [p11_n, p11_t1, p11_t2]
p13=sym('p13_%d',[1  3]); % [p13_n, p13_t1, p13_t2]
p14=sym('p14_%d',[1  3]); % [p14_n, p14_t1, p14_t2]
p15=sym('p15_%d',[1  3]); % [p15_n, p15_t1, p15_t2]
p17=sym('p17_%d',[1  3]); % [p17_n, p17_t1, p17_t2]

% Volume the for simplices that will be adjusted
vol=     vol_sim(p5,p11,p13,p14);  % S_{1,5} (see Table 1)
```

```
vol=vol+vol_sim(p5,p15,p11,p14);  % S_{2,5}
vol=vol+vol_sim(p5,p13,p17,p14);  % S_{3,5}
vol=vol+vol_sim(p5,p17,p15,p14);  % S_{4,5}

% Solve for p5_n=p5(1) with constraint that V_sims=vol1+vol2
syms V_sims
p14_1=simplify(expand(solve(V_sims==vol,p14(1))));

% Display result
fprintf('p5_1= %s \n',char(p14_1))
```

The above code calls the function vol_sims which computes the volume of a simplex using Eq. 16.

```
function [vol] = vol_sim(p1,p2,p3,p4)
% Volume of a simplex defined with the four points p1,p2,p3, & p4
a=p1-p4;
b=p2-p4;
c=p3-p4;
vol=1/6* ...
     (a(1)*(b(2)*c(3)-c(2)*b(3)) ...
    -a(2)*(b(1)*c(3)-c(1)*b(3)) ...
    +a(3)*(b(1)*c(2)-c(1)*b(2)));
end
```

Appendix A.2. Solenoidal Sub-cell Velocities

This MATLAB code computes the matrix shown in Eq. 26 that provides solenoidal sub-cell velocities used to build the internal streaktubes.

```
% Divg-free sub-cell velocities
clear; clc

% Note that u,v,w represent 'fluxes' vel*area
syms u1  u2  u3  u4  u5  u6  u7  u8
syms u1p u2s u3p u4s u5p u6s u7p u8s
syms v1  v2  v3  v4  v5  v6  v7  v8
syms v1p v2p v3s v4s v5p v6p v7s v8s
syms w1  w2  w3  w4  w5  w6  w7  w8
syms w1p w2p w3p w4p w5s w6s w7s w8s
syms g1 g2 g3 g4 g5 g6 g7 g8 g9 g10
syms Dc

% Sub-cell divergence
Df(1)=(u1 -u2 ) + (v1 -v3 ) + (w1 -w5 );
Df(2)=(u2 -u1p) + (v2 -v4 ) + (w2 -w6 );
Df(3)=(u3 -u4 ) + (v3 -v1p) + (w3 -w7 );
Df(4)=(u4 -u3p) + (v4 -v2p) + (w4 -w8 );
Df(5)=(u5 -u6 ) + (v5 -v7 ) + (w5 -w1p);
Df(6)=(u6 -u5p) + (v6 -v8 ) + (w6 -w2p);
Df(7)=(u7 -u8 ) + (v7 -v5p) + (w7 -w3p);
Df(8)=(u8 -u7p) + (v8 -v6p) + (w8 -w4p);

Dc=  u1 +u3 +u5 +u7 +v1 +v2 +v5 +v6 +w1 +w2 +w3 +w4 ...
```

```
      -u1p-u3p-u5p-u7p-v1p-v2p-v5p-v6p-w1p-w2p-w3p-w4p;
% Form functional
P=((u2-u2s)^2+(u4-u4s)^2+(u6-u6s)^2+(u8-u8s)^2  ...
      +  (v3-v3s)^2+(v4-v4s)^2+(v7-v7s)^2+(v8-v8s)^2  ...
      +  (w5-w5s)^2+(w6-w6s)^2+(w7-w7s)^2+(w8-w8s)^2)  ...
      +g1*(Df(1)-Dc/8)+g2*(Df(2)-Dc/8)+g3*(Df(3)-Dc/8)+g4*(Df(4)-Dc/8)  ...
      +g5*(Df(5)-Dc/8)+g6*(Df(6)-Dc/8)+g7*(Df(7)-Dc/8);

% Unknown variables
vars=[u2,u4,u6,u8,v3,v4,v7,v8,w5,w6,w7,w8,g1,g2,g3,g4,g5,g6,g7];
nvars=length(vars);

% Compute derivatives
disp('Derivatives')
dP=sym(zeros(1,nvars));
for n=1:nvars
      dP(n)=diff(P,vars(n))==0;
      disp(dP(n))
end

% Form matrices Ax=b
[A,b]=equationsToMatrix(dP,vars);

% Solve for unknowns
disp('Solving linear system')
x=linsolve(A,b);

% Simplify the equations
disp('Simplifying the equations')
x=simplify(x);

% Display result
disp('Computed Solenoidal velocities')
for n=1:12
      out{n}=[char(vars(n)),' = (',char(x(n)*24),')/24;'];
      disp(out{n});
end

% Write in Matrix form: unknowns=A*invars
invars=[...
      u1,u2s,u3,u4s,u5,u6s,u7,u8s,u1p,u3p,u5p,u7p,...
      v1,v2,v3s,v4s,v5,v6,v7s,v8s,v1p,v2p,v5p,v6p,...
      w1,w2,w3,w4,w5s,w6s,w7s,w8s,w1p,w2p,w3p,w4p];

%% Matrix Form  unknowns=M*b
disp(' ---------- Matrix Form 24*M---------- ')
nin=length(invars);
M=zeros(12,nin);
for n=1:12;
      [C,T]=coeffs(x(n),invars);
      for nn=1:length(C)
            for nnn=1:nin
```

```
                if isequal(T(nn),invars(nnn))
                    M(n,nnn)=double(C(nn))*24;  % Note factor of 24
                end
            end
        end
    end
disp(M')
```

Appendix A.3. Correction Simplices for Internal Streaktubes

This MATLAB code computes Eq. 30 used in the construction of correction simplices for streaktubes associated with internal faces.

```
% Code to find coordinates of point needed to construct correction tets
clear; clc

% Create symbolic points
p1=sym('p1_%d',[1 3]); % [p1_n, p1_t1, p1_t2]
p2=sym('p2_%d',[1 3]); % [p2_n, p2_t1, p2_t2]
p3=sym('p3_%d',[1 3]); % [p3_n, p3_t1, p3_t2]
p4=sym('p4_%d',[1 3]); % [p4_n, p4_t1, p4_t2]
p5=sym('p5_%d',[1 3]); % [p5_n, p5_t1, p5_t2]

% Volume of 1st simplex: p1,p2,p4,p5
vol1=vol_sim(p1,p2,p4,p5);
% Volume of 2nd simplex: p1,p4,p3,p5
vol2=vol_sim(p1,p4,p3,p5);

% Solve for p5_n=p5(1) with constraint that V_sims=vol1+vol2
syms V_sims
p5_1=simplify(expand(solve(V_sims==vol1+vol2,p5(1))));

% Display result
fprintf('p5_1= %s \n',char(p5_1))
```