A New Incompressibility Discretization for a Hybrid Particle MAC Grid Representation with Surface Tension

Wen Zheng*, Bo Zhu*, Byungmoon Kim**, Ronald Fedkiw*

Stanford University, 353 Serra Mall Room 207, Stanford, CA 94305

Adobe Systems Inc., 345 Park Avenue San Jose, CA 95110

Abstract

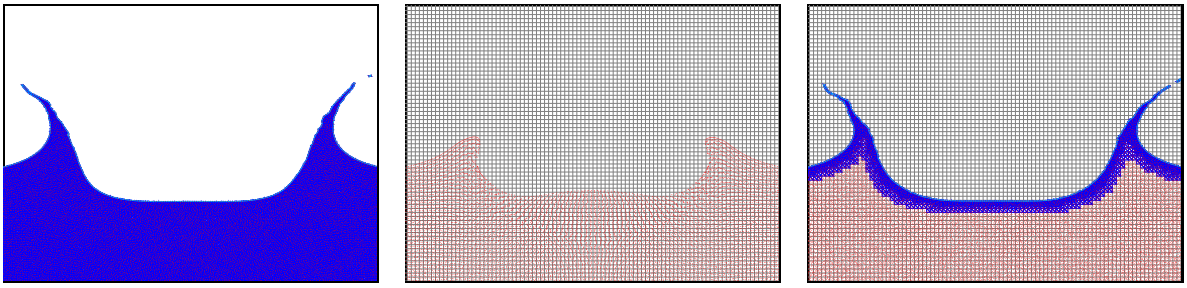
We take a particle based approach to incompressible free surface flow motivated by the fact that an explicit representation of the interface geometry and internal deformations gives precise feedback to an implicit solver for surface tension. Methods that enforce incompressibility directly on the particles are typically numerically inefficient compared to those that utilize a background grid. However, background grid discretizations suffer from inaccuracy near the free surface where they do not properly capture the interface geometry. Therefore, our incompressibility discretization utilizes a particle based projection near the interface and a background MAC grid based projection for efficiency in the vast interior of the liquid domain – as well as a novel method for coupling these two disparate projections together. We show that the overall coupled elliptic solver is second order accurate, and remains second order accurate when used in conjunction with an appropriate temporal discretization for parabolic problems. A similar second order accurate discretization is derived when the MAC grid unknowns are located on faces (as opposed to cell centers) so that Navier-Stokes viscosity can be solved for implicitly as well. Finally, we present a fully implicit approach to surface tension that is robust enough to achieve a steady state solution in a single time step. Beyond stable implicit surface tension for our novel hybrid discretization, we demonstrate preliminary results for both standard front tracking and the particle level set method.

1. Introduction

Incompressible free surface flows are of great interest due to their wide scope of applications ranging from small scale droplet dynamics to large scale dam breaking. Simulation of incompressible free surface flows is a challenging problem because the location of the free surface and hence the shape of the fluid domain is time dependent. Surface tension forces exacerbate the situation by imposing a strict $\Delta t = O(\Delta x^{3/2})$ time step restriction. A number of approaches have been proposed to alleviate this restriction. [24] derived an implicit formulation of surface tension by looking at the changes of curvature over time. This was simplified in [26] via the use of the Laplace-Beltrami operator. This simplified approach was also implemented for finite volume models in [41]. The approaches of [49, 61, 64] split the Laplace-Beltrami operator into a standard Laplacian term and an extra term allowing for a semi-implicit approach treating the standard Laplacian implicitly. There are also approaches that improve stability of VOF-based surface tension schemes [54, 35, 51]. A main difficulty faced by these approaches is the lack of an explicit representation of the interface when deriving an implicit model for surface tension. [43] discretized the surface tension force on a Lagrangian mesh attempting to avoid this difficulty. They coupled the Lagrangian surface tension force with the pressure on the Eulerian grid using an interpolation matrix. This significantly improved the stability, but due to the indirection of the interpolation matrix the surface tension force does not achieve an exact balance with the incompressible pressure and thus still suffers from instability with larger time steps.

*{zhw,boolzhu,rfedkiw}@stanford.edu, Stanford University

**{bmkim}@adobe.com, Adobe Systems Inc.



(a) fully Lagrangian (b) with a background grid (c) hybrid particle MAC grid

Figure 1: Subfigure (a) shows a simulation result using a fully Lagrangian approach where all particles are connected into a Lagrangian mesh to enforce incompressibility. Subfigure (b) shows the result of the same test but using a background grid to enforce incompressibility. This approach runs about 27 times faster than that in Subfigure (a), but it loses significant detail near the fluid surface. Subfigure (c) shows the result using our hybrid approach where particles near the fluid surface are connected into a Lagrangian mesh while a background grid is used for the majority of the interior fluid region. We use the same exact particles as in Subfigure (a) and the same exact background grid as in Subfigure (b). Incompressibility is enforced on this hybrid structure as described in Section 4. This hybrid approach runs about 9 times faster than that in Subfigure (a), and it also preserves all the sub-grid details near the fluid surface unlike Subfigure (b).

Motivated by [43], a Lagrangian description of the interface is desirable when deriving an implicit model for surface tension because interface positions, curvature and force derivatives are readily accessible. This motivates the use of a front tracking method [59, 56]. But front tracking alone does not completely alleviate the issues because one still has to balance surface tension with internal pressure forces, and this coupling is still difficult to address when the pressure forces are described on an Eulerian grid. An alternative solution is to take a fully Lagrangian approach to the problem and solve the incompressibility condition directly on particles. These particle-based methods have gained much attention for incompressible flow problems, see e.g. [14, 13, 25, 34, 62]. However, [55] pointed out that these solvers are expensive due to the inefficiency of solving the Poisson equation on an irregular particle mesh. They also pointed out that mapping particles to a background grid and solving for incompressibility on that grid is more efficient as in [23, 10, 36]. However, the results can be quite inaccurate in the case of free surface flows where a detailed sub-grid description of the interface is lacking. We address these issues by treating the large interior of the fluid region with a standard Eulerian method while (only) meshing up a thickened region near the free surface, so that both efficiency and accuracy are addressed (see Figure 1). Moreover, with the free surface and a thickened region of the fluid near the free surface represented by a Lagrangian mesh, an exact balance between the surface tension force and the pressure force can be achieved resulting in better stability. We derive a second order accurate symmetric positive definite discretization of the Poisson equation on this hybrid particle grid structure in both two and three spatial dimensions. We then extend this Poisson equation solver to the heat equation on the cell centers, and furthermore modify the discretization to solve for the viscous forces on the MAC grid faces. Although this hybrid solver is motivated by the need for surface tension stability, it is also beneficial for simulating incompressible free surface flows without surface tension.

A common issue with particle-based solvers for incompressible flows is drifting of particles due to numerical errors, and as discussed in [19] this drifting causes variations in the particle density which subsequently cause robustness issues. In order to control particle density, various authors proposed applying a second elliptic projection [40, 36, 25]. Unlike these approaches, we calculate the particle masses from their control volumes and a constant density. In this way, the particle density is independent of the fluid density avoiding robustness issues. Although we do not require a second projection, we point out that one could combine the projection for enforcing incompressibility and the projection for enforcing a constant particle density into a single projection using the technique proposed in [38].

Section 7 addresses surface tension. For illustrative purposes, we start with a fully Lagrangian representation and we conduct the exploration using a step-by-step constructive approach. We stress that fully coupling the surface tension force to the pressure force seems to be a key for stability. In addition, imposing

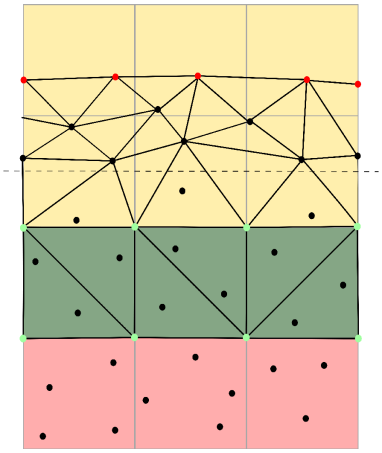


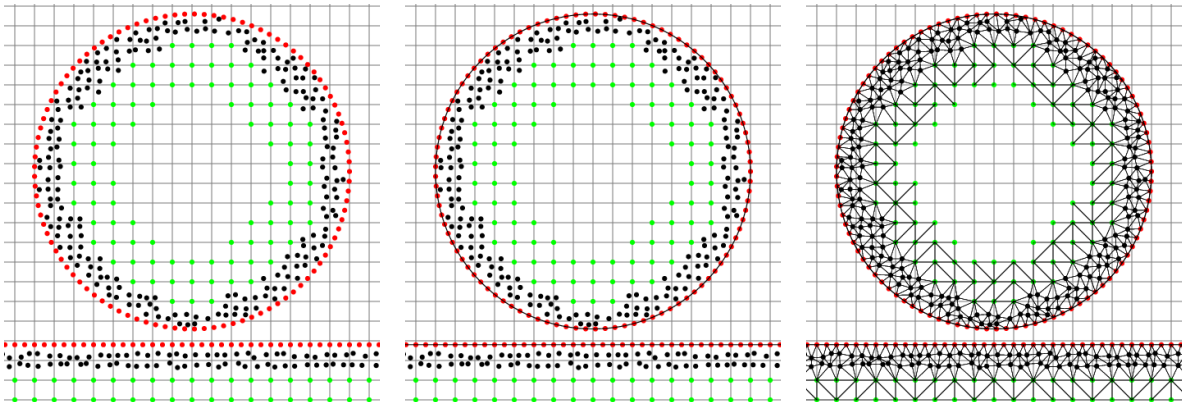
Figure 2: The hybrid mesh/grid structure. Surface cells are shaded yellow, buffer cells are shaded green, and interior cells are shaded pink. The red dots represent surface particles, the black dots represent regular particles, and the green dots represent the grid nodes of buffer cells. The dashed line is $.5\Delta x$ away from the green cells. Regular particles below the dashed line are ignored by the meshing algorithm. Regular particles above the dashed line together with surface particles and the nodes of buffer cells are connected into a triangle mesh.

a constant volume condition instead of the divergence free condition seems to greatly improve accuracy especially for large time steps as in the steady state case. The final scheme is a fully implicit volume conserving scheme that can achieve a steady state solution with very large time steps. This scheme is extended to our hybrid particle grid structure for the sake of efficiency. Finally, we show that the same ideas can be applied to the standard front tracking method and to the particle level set method. Note that our implicit approach requires formulating a tightly coupled system between the implicit surface tension force and the incompressible pressure force, and we utilize the technique proposed in [42] so that this coupled system can be made symmetric positive definite and subsequently be solved efficiently.

2. Mesh Generation

The entire computational domain is defined using a Marker-And-Cell (MAC) grid, and the part of the domain occupied by fluid is identified by the presence of marker particles. Particles are only needed in the thickened outer band and potentially a small buffer region, however for the sake of simplicity we utilize particles throughout the entire region occupied by fluid. We have two types of particles, *surface particles* and *regular particles*. Given an expression for the initial position and shape of the fluid, surface particles are placed exactly on the fluid surface and regular particles are randomly placed inside the fluid domain in such a way that there are roughly four particles per cell in two spatial dimensions and eight particles per cell in three spatial dimensions. Throughout the simulation, we maintain this roughly uniform distribution property by particle reseeding.

Each time step, grid cells containing particles are marked as *fluid cells*, while the others are marked as *exterior cells*. The fluid cells are further categorized into three types: *surface cells*, *interior cells* and *buffer cells*. Surface cells are fluid cells near the interface where we construct a triangle (or tetrahedral) mesh connecting particles, interior cells are deeper away from the interface and are discretized using a standard MAC grid, and buffer cells serve as a blending region that stitches the triangle mesh and the MAC grid together. Generally speaking, we use at least two layers of surface cells and one layer of buffer cells. Regular particles that are inside either buffer or interior cells are ignored by the meshing algorithm. We also ignore any regular particle whose center is within $.5\Delta x$ of the boundary of buffer cells. The other regular particles together with surface particles and the grid nodes of buffer cells are used in the meshing algorithm to generate a triangle mesh (Figure 2).

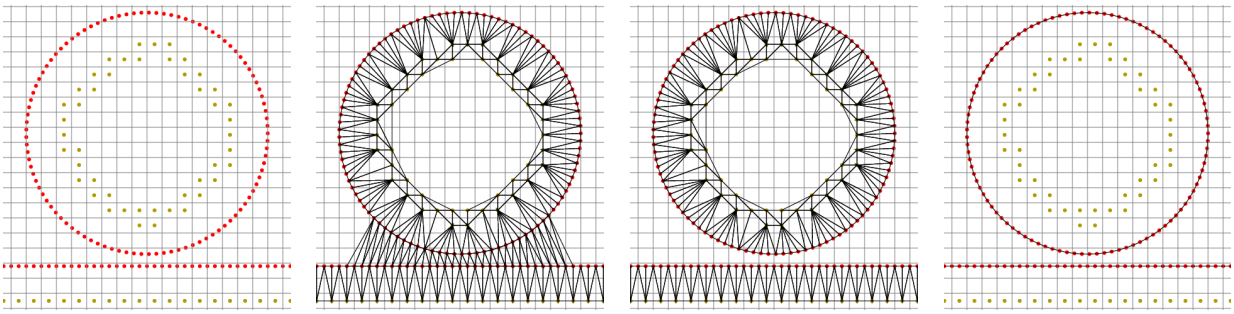


(a) Particle input (b) Step 1 (c) Step 2

Figure 3: The framework of our meshing algorithm. Red dots are surface particles, black dots are regular particles that need to be connected, and green dots are the grid nodes of buffer cells. Regular particles ignored by our meshing algorithm are not shown in the figure. Step 1 constructs a surface mesh from surface particles to represent the fluid surface. Step 2 fills the gap between the surface mesh and interior cells with a volumetric mesh.

Our meshing algorithm contains two steps (Figure 3). First, a surface mesh consisting of line segments (or triangles in three spatial dimensions) is constructed from surface particles to represent the fluid surface. Second, all the surface particles, appropriate regular particles, and the grid nodes of buffer cells are connected together to form an interior volumetric mesh. For the first step, the restricted Delaunay method [16] is used to construct the surface mesh. This method first constructs an auxiliary volumetric mesh from surface particles using a standard Delaunay method. For the sake of efficiency, we place one layer of auxiliary particles at the centers of buffer cells to localize this meshing process (see Figure 4(a) and 4(b)). The distance from auxiliary particles to surface particles must be large enough so that the topology of the surface mesh will not be affected. Since the distance from the centers of buffer cells to the surface particles is at least three times the average particle spacing, we found that adding auxiliary particles at the centers of buffer cells is sufficient in practice. Next, we delete all triangles whose circumcenters are outside the fluid domain (Figure 4(c)), and extract the surface mesh from the remaining triangles (Figure 4(d)). Extracting the surface mesh from a volumetric mesh guarantees that the surface mesh is a manifold. Moreover, it is provable that the restricted Delaunay method gives a good approximation of the fluid domain in both two and three spatial dimensions in the sense that it correctly captures the topology of the fluid domain given a sufficiently dense sampling [4]. The restricted Delaunay method requires a representation of the fluid domain to determine whether a circumcenter is inside it. For the initial data, we already have such a representation as input. For later time steps, we proceed as follows. If the location of a circumcenter is inside a cell that is at least one layer away from the cells containing surface particles, one can simply test whether the location is inside a fluid cell. Otherwise, a more accurate representation of the fluid domain is needed. For this purpose, we advect the volumetric mesh constructed at the previous time step to the current time step, and test whether the location of a circumcenter is inside any triangle of the advected mesh. Note that the advected mesh may have some inverted triangles, and these are ignored during the inside/outside test because they are invalid for the purpose of representing the current fluid domain. The advected mesh is only used for the inside/outside test and is discarded afterwards.

The second step of our meshing algorithm generates a volumetric mesh filling the gap between the surface mesh and interior cells. This volumetric mesh should conform to the surface mesh generated at the first step. We also require it to conform to the grid faces of buffer cells to facilitate discretization. These requirements motivate the use of the constrained Delaunay method [47]. However, the constrained Delaunay mesh may not exist in some cases in three spatial dimensions. Therefore, we construct the mesh as Delaunay as possible while guaranteeing a valid mesh structure. Maintaining a valid mesh structure during meshing can be expensive, because this requires doing intersection tests and other validity checks when generating



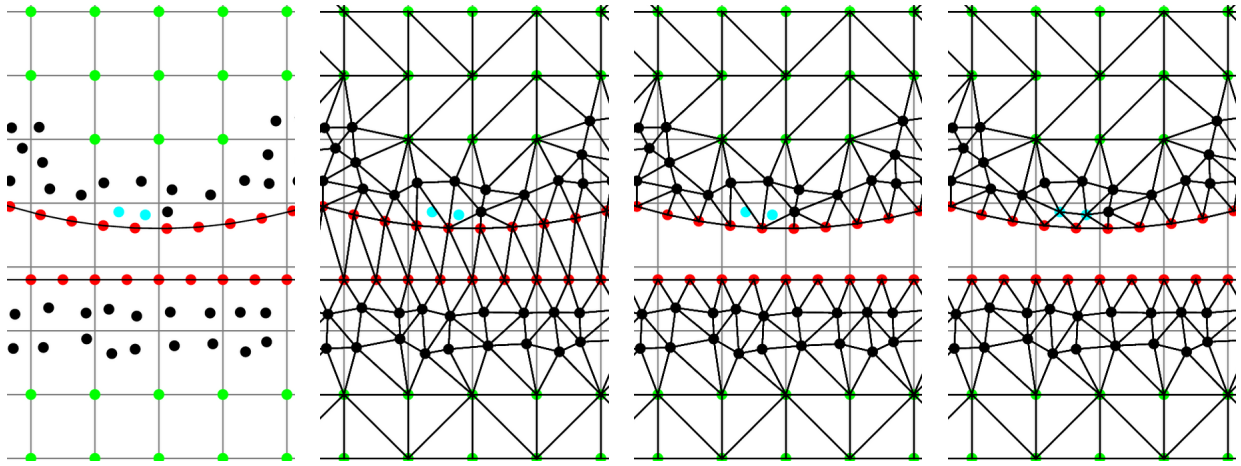
(a) Placing auxiliary particles (b) Constructing an auxiliary volumetric mesh (c) Deleting "outside" triangles (d) Extracting the surface mesh

Figure 4: Step 1 of our meshing algorithm. The input to this step are surface particles (red dots). We place one layer of auxiliary particles (yellow dots) to localize the meshing process (a). An auxiliary volumetric mesh is constructed first (b). The "outside" triangles whose circumcenters are outside the fluid domain are deleted (c). The surface mesh is then extracted from the remaining triangles (d).

each element. To improve the efficiency, we split this into two passes. The first pass does the meshing process without checking the validity of the mesh structure. To ensure the mesh structure is valid during the first pass, we identify particles that may endanger the conformation requirement (Figure 5(a), we call these particles *problematic particles*) and ignore them during the first pass. The other particles are connected into a volumetric mesh using the standard Delaunay method (Figure 5(b)). All triangles whose circumcenters are outside the fluid domain are deleted (Figure 5(c)). Note that some triangles may intersect interior cells, and we delete these as well. After we obtain a volumetric mesh from the first pass, we then do a second pass inserting the problematic particles back into the mesh (Figure 5(d)). We use the Bowyer-Watson algorithm [9, 60] to insert problematic particles, but during insertion we check the validity of the mesh structure and sacrifice the Delaunay property when necessary to maintain a valid mesh. Since we already have a valid mesh to start with, maintaining a valid mesh is relatively straightforward. And since the number of problematic particles is small, the cost of maintaining a valid mesh is minimized.

To decide which particles should be treated as problematic, we utilize a property of the Delaunay triangulation that states that a segment is generated in a Delaunay mesh if and only if at least one of its circumcircles contains no other particles. Therefore, to ensure that all segments in the surface mesh will be included in the volumetric mesh generated in the first pass, it is sufficient to identify problematic particles as regular particles that are within the minimum circumcircles of the segments in the surface mesh. In three spatial dimensions, similar properties hold and problematic particles are identified as regular particles inside the minimum circumspheres of both the segments and the triangles in the surface mesh. Note that for grid faces of buffer cells, the particles within their minimum circumcircles are already ignored by the meshing algorithm. Thus, in two spatial dimensions, there will be a segment connecting the two grid nodes for each grid face of buffer cells, i.e. the mesh will be conforming. Similarly in three spatial dimensions, there will be two triangles connecting the four grid nodes for each grid face of buffer cells.

Note that both steps of our meshing algorithm use a standard Delaunay method as a building block. An advancing front Delaunay meshing algorithm similar to [21] is employed as our standard Delaunay method. It starts by picking an initial triangle whose circumcircle contains no other particles. The edges of the initial triangle are added to the *front* which is a queue of segments waiting to be processed. Behind the front are particles already connected, and ahead of the front are particles waiting for connection. It then marches outwards by dequeuing one segment at a time from the front and searching for the particle that forms the minimum circumcircle with the dequeued segment. A new triangle is then generated from the dequeued segment and the chosen particle, and the front is updated accordingly. If the front is empty, it tries to find another initial triangle. If no such triangle is found, the algorithm ends. This algorithm is especially suitable for uniformly distributed particles, because we can simply localize the search region to a local neighborhood when choosing particles. This leads to constant time cost for each iteration and a linear time complexity



(a) Identifying "problematic" particles (b) First pass: constructing a volumetric mesh (c) First pass: deleting "outside" triangles (d) Second pass: inserting "problematic" particles

Figure 5: Step 2 of our meshing algorithm. The input to this step are surface particles (red dots), regular particles that need to be connected (black dots), grid nodes of buffer cells (green dots), and the surface mesh generated from Step 1. Regular particles that are inside the circumcircles of the segments in the surface mesh (cyan dots) are identified as "problematic" particles (a). The first pass of meshing constructs a volumetric mesh (b) and deletes all "outside" triangles whose circumcenters are outside the fluid domain (c). The second pass of meshing adds the "problematic" particles back into the mesh (d).

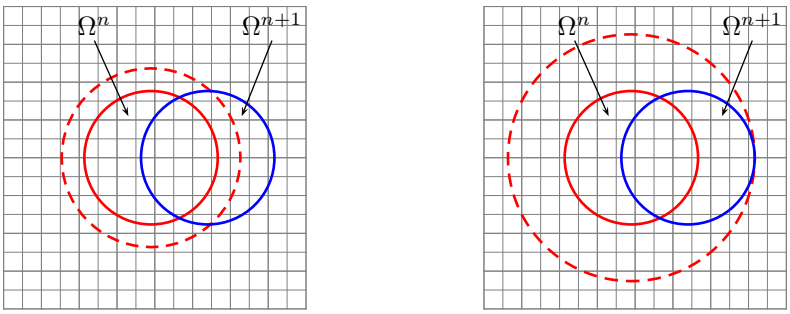
overall. Degenerate cases occur when more than three particles are cocircular which causes robustness issues. In order to eliminate degenerate cases, we use the symbolic perturbation method proposed by [15] together with the adaptive exact arithmetic predicates proposed by [46].

After meshing, we reseed particles to maintain a roughly uniform distribution. If a mesh segment is longer than 1.5 times the average particle spacing, a new particle is inserted in the middle of the segment with its velocity averaged from the two particles at the ends of the segment. We also randomly insert particles into fluid cells that are at least one layer away from the exterior cells and contain less than four particles (eight particles in three spatial dimensions). If the randomly inserted particles are inside the triangle mesh, their velocities are interpolated from the mesh using barycentric interpolation. Otherwise, their velocities are interpolated from the MAC grid faces using bilinear/trilinear interpolation. Although we add particles after meshing, particle deletion occurs before remeshing and directly after particle advection. This is accomplished by merging two particles if their distance is less than one third of the average particle spacing by placing a new particle equal distance between them. The velocity of the new particle is averaged from the two original particles. In order to maintain a good sampling of the surface, when a surface particle is within merging distance of a regular particle, we simply delete the regular particle and perform no merging operation. Particles that escape from the fluid domain due to topology changes are deleted.

To address cases near the boundary of the Cartesian computational domain, the mesh is generated against the boundary of the computational domain so that MAC grid cells do not touch the boundary. Further optimization would allow MAC grid cells to touch the boundary of the computational domain, but this would require more careful consideration in light of contact angles – which we leave to future work.

3. Advection

Consider the advection of an arbitrary scalar field ψ driven by a given velocity field $\mathbf{u}$. The ψ values are defined at the centers of the interior cells and on the particles of the Lagrangian mesh as well as the grid nodes of buffer cells. Particles that are not connected to the Lagrangian mesh do not have their ψ values readily defined, and these values can be interpolated either from the Lagrangian mesh using barycentric interpolation or from grid cells using bilinear/trilinear interpolation. For the purpose of the advection tests



(a) Advection of ψ (b) Self-advection of velocities

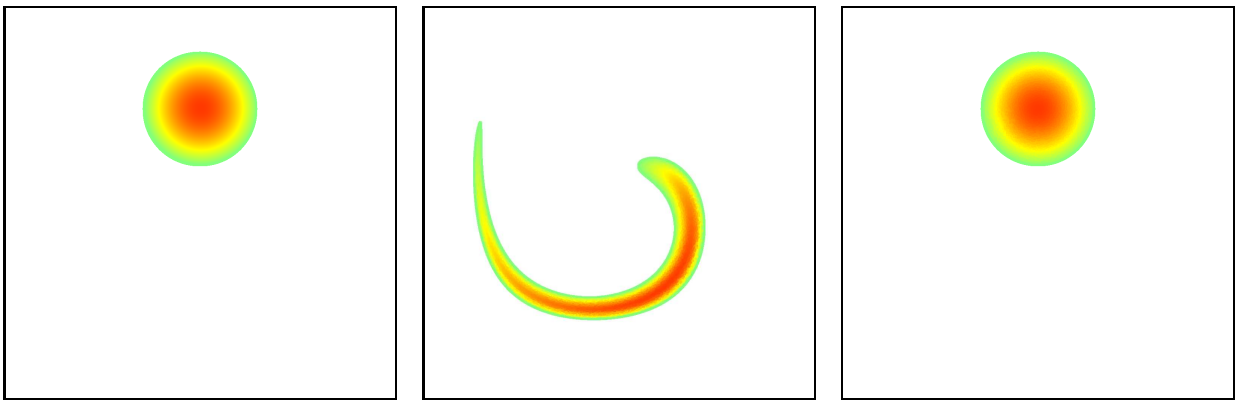
Figure 6: The range of cells that require valid data for the semi-Lagrangian advection. The values are advected from the red solid circle to the blue solid circle. For advection of the ψ values (a), the range of cells that have valid data (the red dashed circle) must include enough cells to ensure valid interpolation stencils at all backward traced locations. For self-advection of velocities (b), the red dashed circle must also include the blue solid circle so that the time t^n characteristic velocities are defined throughout Ω^{n+1}.

in this section, we simply initialize all particles with analytic ψ values. Before advection, extra layers of MAC grid cells surrounding the time t^n interior cells are filled with ψ values so that all relevant cells have valid data for advection. The advection of ψ can then be done separately on particles and on the MAC grid. Particles are advected forward using either the forward Euler method (FE) or the second order accurate Runge-Kutta method (RK2) carrying ψ values along with them, and then the hybrid mesh/grid structure is rebuilt. The ψ values of the time t^{n+1} interior cells are updated using either the semi-Lagrangian method (SL) or the semi-Lagrangian style MacCormack method (SL-MacCormack) [44], and the ψ values of the time t^{n+1} grid nodes of buffer cells are obtained from the MAC grid by averaging from neighboring cells after advection.

3.1. Semi-Lagrangian Advection

On the MAC grid we define the region that contains the time t^n interior cells as Ω^n and the region that contains the time t^{n+1} interior cells as Ω^{n+1}. Consider the semi-Lagrangian method which advects ψ from Ω^n to Ω^{n+1} by tracing backwards from the cell centers in Ω^{n+1}. One needs to have valid ψ values defined in a band of ghost cells surrounding Ω^n before each time step of advection in order to deal with the fact that some of the semi-Lagrangian characteristic rays may land outside of Ω^n because of inaccuracy as well as inconsistency in the meshing that determines Ω^n and Ω^{n+1}. Moreover, one needs additional ghost cells so that there is a valid interpolation stencil for every location where semi-Lagrangian interpolation is required, see Figure 6(a). In the case of incompressible flow, velocities are self-advected and thus must be defined in Ω^{n+1} to enable the backward tracing. This requires a potentially larger range of ghost cells in order to cover Ω^{n+1} (see Figure 6(b)). Although self-advection is not important for the advection tests in this section, it will be important for incompressible flow which we consider later.

Ghost cells are filled by interpolating their ψ values from the Lagrangian mesh using barycentric interpolation. Note that as mentioned above mesh elements (triangles/tetrahedra) connecting the grid nodes of buffer cells obtain their ψ values from the MAC grid after each time step of advection. Using these for interpolation creates a cyclic dependency which seems to degrade the order of accuracy of the scheme, see Section 3.4. Therefore, the interior regions Ω^n and Ω^{n+1} include all grid cells interior to the fluid domain whose cell centers cannot be interpolated from the subset of the Lagrangian mesh that includes no grid nodes of buffer cells. Ghost cells that are exterior to the fluid region must be filled by extrapolation, which could be avoided if the Lagrangian mesh is thick enough to ensure that no ghost cells exterior to the Lagrangian mesh are required.



(a) t=0 (b) t=.5 (c) t=1

Figure 7: The shape of the fluid region and the distribution of the scalar field at various times in the single vortex flow test. The warmer colors indicate larger values of the scalar field.

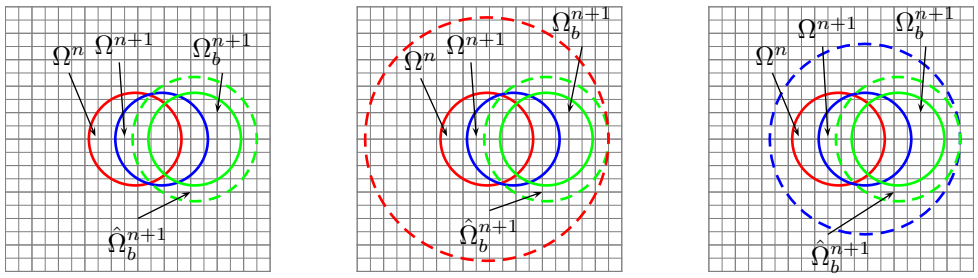
Table 1: The order of accuracy of the single vortex flow test using FE+SL with a thickened mesh.

	$t = .5$				$t = 1$			
Grid Cells	L^1 Diff	Order	L^∞ Diff	Order	L^1 Error	Order	L^∞ Error	Order
256	–	–	–	–	9.53×10^{-2}	–	1.96×10^{-1}	–
512	2.34×10^{-2}	–	7.72×10^{-2}	–	7.17×10^{-2}	0.41	1.55×10^{-1}	0.34
1024	1.62×10^{-2}	0.53	4.36×10^{-2}	0.82	4.28×10^{-2}	0.74	9.87×10^{-2}	0.65
2048	9.29×10^{-3}	0.80	2.45×10^{-2}	0.83	2.39×10^{-2}	0.84	5.62×10^{-2}	0.81

We have implemented the time-varying single vortex flow test from [20]. The domain size is $[0, 1] \times [0, 1]$, and the initial shape is a disk centered at $(.5, .75)$ with a radius of .15. A Gaussian scalar field $\psi = \exp(-(\mathbf{x} - \mathbf{x}_c)^2)/2r^2$ is initialized inside the disk. The velocity field is defined by the stream function $\Phi = \sin^2(\pi x)\sin^2(\pi y)/\pi$, which is multiplied by $\cos(\pi t)$ in order to achieve maximum stretching at $t = .5$ and return to the initial state at $t = 1$ (Figure 7). Particles are advected using the forward Euler method, while the semi-Lagrangian method is used on the MAC grid. The orders of convergence are computed at both $t = .5$ and $t = 1$. For $t = .5$ the differences between pairs of consecutive resolutions are computed by comparing ψ defined at the low resolution with that interpolated at the same location from the high resolution hybrid mesh/grid structure. The order of convergence is then computed between each two consecutive pairs of resolutions. At $t = 1$, the scalar field ψ should return to its initial state, and thus the results at various resolutions are compared to analytic solution. Table 1 shows the results using a thick enough Lagrangian mesh to preclude the need for extrapolation.

3.2. Semi-Lagrangian-MacCormack Advection

Next consider the SL-MacCormack scheme which consists of a forward advection step followed by a backward advection step. The forward advection step uses the semi-Lagrangian method to advect ψ from Ω^n to Ω^{n+1}. The backward advection step then does another semi-Lagrangian advection that advects ψ from a region Ω_b^{n+1} to Ω^{n+1} by tracing backward along the negated velocity field, see Figure 8(a). The backward advected ψ values in Ω^{n+1} are then compared with the original ψ values at time t^n to compute the errors and correct the results of the first semi-Lagrangian forward advection step to be second order accurate. The requirement for ghost cells solely imposed by the forward advection step is identical to that of the semi-Lagrangian method. However, the backward advection step imposes an additional requirement that the ψ values at time t^{n+1} must be defined inside Ω_b^{n+1} as well as inside a band around Ω_b^{n+1} to ensure a valid interpolation stencil for every backward traced location. We define this region as $\hat{\Omega}_b^{n+1}$ (the green dashed circle in Figure 8(a)). Note that in the case of self-advection for incompressible flow, the velocities



(a) Backward advection (b) Single filling (c) Double filling

Figure 8: The range of cells that require valid data for the SL-MacCormack advection. The forward advection step advects values from the red solid circle to the blue solid circle, while the backward advection step advects values from the green solid circle to the blue solid circle. For the backward advection step (a), the valid data must be defined in the green dashed circle. This data can be updated using the single filling scheme (b), which requires that the red dashed circle must have valid data defined at time t^n, and it must include the green dashed circle (when considering self-advection). Alternatively, when using the double filling scheme (c), the valid data must be defined at time t^{n+1} inside the blue dashed circle, and it must include the green dashed circle.

Table 2: The order of accuracy of the single vortex flow test using RK2+SL-MacCormack with a thickened mesh.

	$t = .5$				$t = 1$			
Grid Cells	L^1 Error	Order	L^∞ Error	Order	L^1 Error	Order	L^∞ Error	Order
256	–	–	–	–	7.06×10^{-4}	–	6.32×10^{-3}	–
512	1.77×10^{-3}	–	6.79×10^{-3}	–	1.14×10^{-4}	2.63	2.00×10^{-3}	1.66
1024	5.29×10^{-4}	1.74	1.83×10^{-3}	1.89	2.04×10^{-5}	2.48	5.12×10^{-4}	1.97
2048	1.43×10^{-4}	1.89	5.05×10^{-4}	1.85	3.89×10^{-6}	2.39	1.29×10^{-4}	1.99

required in Ω^{n+1} for the backward advection step have already been defined by the first forward advection step.

There are two ways to define ψ in $\hat{\Omega}_b^{n+1}$ before the backward advection step, which we call the *single filling* scheme and the *double filling* scheme (both explained for the case of self-advection). The single filling scheme defines ψ values in $\hat{\Omega}_b^{n+1}$ via the first forward advection step and thus requires more ghost cells at time t^n, see Figure 8(b) and compare the red dashed circle containing $\hat{\Omega}_b^{n+1}$ to the red dashed circle in Figure 6(b) that contains Ω^{n+1}. Alternatively, the double filling scheme places ghost cells around Ω^{n+1} that are thick enough to cover $\hat{\Omega}_b^{n+1}$ as seen in Figure 8(c). This second step of ghost cell population allows for a thinner Lagrangian mesh. We have implemented the single vortex flow test using the second order accurate Runge-Kutta method on particles along with the SL-MacCormack method using both single and double filling. Table 2 shows the results with double filling, and we note that the single filling scheme gives commensurate results.

Note that the back and forth error compensation and correction method (BFECC) proposed in [17, 31, 32, 18] is a possible alternative to the SL-MacCormack method. Its first step is a forward advection step, the same as that of SL-MacCormack. Its second step, however, backward advects ψ from Ω^{n+1} to Ω^n instead of from Ω_b^{n+1} to Ω^{n+1} as in SL-MacCormack. The errors are then computed in Ω^n by comparing the backward advected ψ values with the original ψ values at time t^n. Finally, the errors are forward advected again to Ω^{n+1} and used to correct the results of the first forward advection step. The BFECC method exactly inverts the source and target regions advecting the red solid circle to the blue solid circle and then back to the red solid circle to compute the error. In this sense, the BFECC method better estimates the error of the first forward advection step especially when considering that one might encounter boundary conditions in the green solid circle such as a free surface or a solid object, which would pollute the error estimate.

Table 3: The order of accuracy of the Eulerian single vortex flow test using SL with constant extrapolation.

	$t = .5$				$t = 1$			
Grid Cells	L^1 Diff	Order	L^∞ Diff	Order	L^1 Error	Order	L^∞ Error	Order
256	–	–	–	–	1.02×10^{-1}	–	3.85×10^{-1}	–
512	1.96×10^{-2}	–	5.30×10^{-2}	–	8.59×10^{-2}	0.24	3.33×10^{-1}	0.21
1024	1.83×10^{-2}	0.09	4.63×10^{-2}	0.20	6.17×10^{-2}	0.48	2.83×10^{-1}	0.23
2048	1.29×10^{-2}	0.50	3.51×10^{-2}	0.40	3.91×10^{-2}	0.66	2.23×10^{-1}	0.34

Table 4: The order of accuracy of the Eulerian single vortex flow test using SL with linear extrapolation.

	$t = .5$				$t = 1$			
Grid Cells	L^1 Error	Order	L^∞ Error	Order	L^1 Error	Order	L^∞ Error	Order
256	–	–	–	–	1.34×10^{-1}	–	2.48×10^{-1}	–
512	3.04×10^{-2}	–	6.64×10^{-2}	–	8.65×10^{-2}	0.63	1.70×10^{-1}	0.55
1024	1.90×10^{-2}	0.68	4.32×10^{-2}	0.62	4.96×10^{-2}	0.80	1.02×10^{-1}	0.74
2048	1.06×10^{-2}	0.85	2.48×10^{-2}	0.80	2.68×10^{-2}	0.89	5.65×10^{-2}	0.85

3.3. Extrapolation

Next we consider a more practical unthickened Lagrangian mesh, and thus extrapolation needs to be considered in ghost cells. The extrapolation is accomplished using a fast marching type method as in [2] or a higher order accurate method as in [5] which both require a level set function defined locally on the MAC grid. The level set is constructed by initializing the cells adjacent to the interface [37] and then using a fast marching method to populate grid cells in a band [57, 58, 45, 11].

In order to examine the influence of extrapolation on the accuracy of advection, we have implemented the single vortex flow test using purely Eulerian methods to exclude the influence of interacting with a Lagrangian mesh. The particle level set method [20] is used to track the free surface, where the marker particles and the level set are separately advected using a third order accurate TVD Runge-Kutta method [48], and the spatial derivatives are calculated with a fifth order accurate Hamilton-Jacobi WENO scheme [29]. The scalar field ψ defined at the centers of MAC grid cells is advected by either the semi-Lagrangian scheme or the SL-MacCormack scheme. Ghost cells are extrapolated from inside the fluid region using the methods in [5] with various orders of accuracy. The results using the semi-Lagrangian method are shown in Tables 3 and 4, and the results using the SL-MacCormack method are shown in Table 5. For the semi-Lagrangian method, although constant extrapolation is theoretically sufficient to achieve first order accuracy, it slows down the convergence, whereas linear extrapolation gives convergence rates more commensurate with that in Table 1—and thus we prefer linear extrapolation. For the SL-MacCormack method, first order accuracy is achieved using constant extrapolation, and second order accuracy is achieved using either linear or quadratic extrapolation as expected.

Next we test the efficacy of the full hybrid mesh/grid approach using extrapolation. The first test uses the forward Euler method on particles along with the semi-Lagrangian method and linear extrapolation on the MAC grid, shown in Table 6. The second test uses the second order accurate Runge-Kutta method on particles along with the SL-MacCormack method and linear extrapolation on the MAC grid, shown in Table 7.

3.4. Cyclic Dependency

For the purpose of the advection tests in this section, we do not interpolate from the MAC grid to particles that are not connected to the Lagrangian mesh or interpolate from the mesh elements connecting the grid nodes of buffer cells to the MAC grid because this would result in a cyclic dependency which seems to degrade the order of accuracy. However, in a real simulation, one cannot avoid these interpolations because the degrees of freedom of the hybrid mesh/grid structure change due to other equations, such as the Poisson

Table 5: The order of accuracy of the Eulerian single vortex flow test using SL-MacCormack with linear extrapolation.

	$t = .5$				$t = 1$			
Grid Cells	L^1 Diff	Order	L^∞ Diff	Order	L^1 Error	Order	L^∞ Error	Order
256	–	–	–	–	1.91×10^{-3}	–	1.32×10^{-1}	–
512	3.32×10^{-3}	–	4.26×10^{-2}	–	3.47×10^{-4}	2.47	4.09×10^{-2}	1.69
1024	7.94×10^{-4}	2.06	1.20×10^{-2}	1.83	7.09×10^{-5}	2.29	9.84×10^{-3}	2.06
2048	1.96×10^{-4}	2.02	3.35×10^{-3}	1.84	1.59×10^{-5}	2.16	2.70×10^{-3}	1.86

Table 6: The order of accuracy of the single vortex flow test using FE+SL with linear extrapolation.

	$t = .5$				$t = 1$			
Grid Cells	L^1 Diff	Order	L^∞ Diff	Order	L^1 Error	Order	L^∞ Error	Order
256	–	–	–	–	1.13×10^{-1}	–	2.25×10^{-1}	–
512	2.65×10^{-2}	–	6.95×10^{-2}	–	7.60×10^{-2}	0.58	1.61×10^{-1}	0.48
1024	1.69×10^{-2}	0.65	4.13×10^{-2}	0.75	4.37×10^{-2}	0.80	9.95×10^{-2}	0.69
2048	9.43×10^{-3}	0.84	2.46×10^{-2}	0.75	2.41×10^{-2}	0.86	5.62×10^{-2}	0.82

equation which enforces incompressibility. This then forces various interpolations in order to keep the hybrid mesh/grid structure consistent.

To examine the influence of the cyclic dependency, we modified the tests in Tables 6 and 7 by including those interpolations before each time step of advection. Table 8 shows the results using the forward Euler method on particles and the semi-Lagrangian method on the MAC grid which remains first order accurate. Table 9 shows the results using the second order accurate Runge-Kutta method on particles and SL-MacCormack on the MAC grid, where the errors are higher than that in Table 7 but the order of accuracy is roughly the same.

We have also implemented the rigid body rotation test from [20, 63] where cyclic dependency can slow down convergence of the SL-MacCormack method. The domain size is $[0, 1] \times [0, 1]$, and the initial shape is a slotted circle centered at $(.5, .75)$ with a radius of .15, a slot width of .05, and a slot length of .25. The same Gaussian function in the single vortex flow test is used to initialize the scalar field ψ inside the shape. The constant vorticity field is given by $u = 2\pi(.5 - y)$ and $v = 2\pi(x - .5)$ so that the disk finishes one resolution at $t = 1$. The second order accurate Runge-Kutta method is used on particles, and the SL-MacCormack method with linear extrapolation is used on the MAC grid. Table 10 shows the comparison between the results with and without cyclic dependency after one revolution. The order of convergence of the L^∞ norms drops by about .5. However, the L^1 norms still converge at a second order rate because the region being polluted by the cyclic dependency occurs on a lower dimensional subset of the simulation.

Table 7: The order of accuracy of the single vortex flow test using RK2+SL-MacCormack with linear extrapolation.

	$t = .5$				$t = 1$			
Grid Cells	L^1 Diff	Order	L^∞ Diff	Order	L^1 Error	Order	L^∞ Error	Order
256	–	–	–	–	6.21×10^{-4}	–	8.17×10^{-3}	–
512	2.13×10^{-3}	–	6.58×10^{-3}	–	1.15×10^{-4}	2.44	2.45×10^{-3}	1.74
1024	5.64×10^{-4}	1.91	1.81×10^{-3}	1.86	2.23×10^{-5}	2.36	6.51×10^{-4}	1.91
2048	1.47×10^{-4}	1.94	4.97×10^{-4}	1.86	4.47×10^{-6}	2.32	1.95×10^{-4}	1.74

Table 8: The order of accuracy of the single vortex flow test using FE+SL with linear extrapolation and cyclic dependency.

	$t = .5$				$t = 1$			
Grid Cells	L^1 Diff	Order	L^∞ Diff	Order	L^1 Error	Order	L^∞ Error	Order
256	–	–	–	–	1.25×10^{-1}	–	2.36×10^{-1}	–
512	2.98×10^{-2}	–	6.37×10^{-2}	–	7.85×10^{-2}	0.67	1.62×10^{-1}	0.54
1024	1.74×10^{-2}	0.78	4.14×10^{-2}	0.62	4.42×10^{-2}	0.83	9.95×10^{-2}	0.70
2048	9.50×10^{-3}	0.87	2.46×10^{-2}	0.75	2.42×10^{-2}	0.87	5.62×10^{-2}	0.82

Table 9: The order of accuracy of the single vortex flow test using RK2+SL-MacCormack with linear extrapolation and cyclic dependency.

	$t = .5$				$t = 1$			
Grid Cells	L^1 Diff	Order	L^∞ Diff	Order	L^1 Error	Order	L^∞ Error	Order
256	–	–	–	–	6.39×10^{-3}	–	7.71×10^{-2}	–
512	6.18×10^{-3}	–	5.55×10^{-2}	–	8.98×10^{-4}	2.83	2.33×10^{-2}	1.73
1024	1.14×10^{-3}	2.43	1.30×10^{-2}	2.09	1.62×10^{-4}	2.47	5.92×10^{-3}	1.98
2048	2.47×10^{-4}	2.21	3.26×10^{-3}	2.00	3.33×10^{-5}	2.28	1.82×10^{-3}	1.70

4. Poisson Equation

Aiming at incompressible flow in Section 6, we first consider the variable coefficient Poisson equation

$$\nabla \cdot \beta(\vec{x}) \nabla \phi(\vec{x}) = f(\vec{x}), \qquad \vec{x} \in \Omega \tag{1}$$

$$\phi(\vec{x}) = g(\vec{x}), \qquad \vec{x} \in \partial\Omega_D \tag{2}$$

$$\vec{n} \cdot \nabla \phi(\vec{x}) = h(\vec{x}), \qquad \vec{x} \in \partial\Omega_N \tag{3}$$

where Ω is the interior region, $\partial\Omega_D$ and $\partial\Omega_N$ are the portions of the boundary where Dirichlet and Neumann boundary conditions are enforced, and $\vec{n}$ is the outward pointing normal. We test boundary conditions similar to the incompressible flow problems of interest, i.e. Dirichlet boundary conditions everywhere, except where Ω touches the Cartesian computational boundary where Neumann boundary conditions are applied.

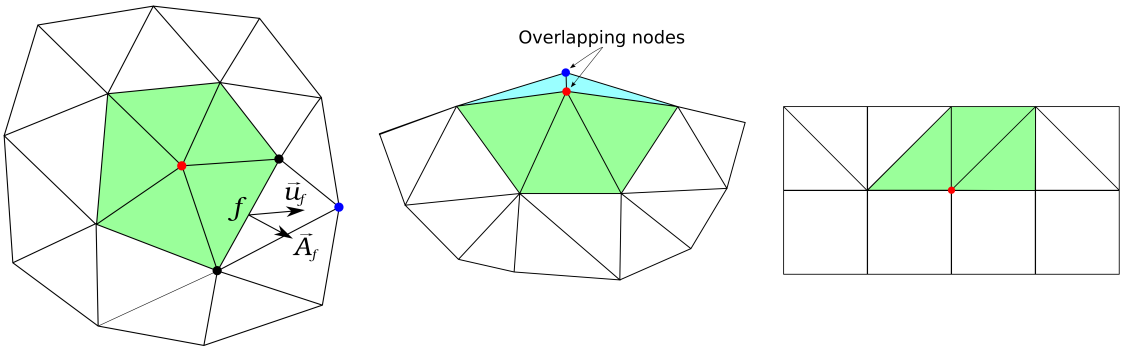
4.1. MAC Grid Discretization

As noted in Section 2, we place MAC grid degrees of freedom sufficiently interior to the domain Ω such that boundary conditions need not be considered on the MAC grid. Both ϕ and f are defined at cell centers, and β is defined on cell faces where the components of $\nabla\phi$ are evaluated. The volume-weighted divergence operator is discretized by integrating the divergence of the vector field $\vec{u} = \beta\nabla\phi$ over the volume of each cell. This is computed as

$$V_c \nabla \cdot \vec{u} = \sum_{f \in F_c} (u_f \vec{n}_f) \cdot \vec{A}_f, \tag{4}$$

Table 10: The order of accuracy of the rigid body rotation test using RK2+SL-MacCormack with linear extrapolation.

	Without cyclic dependency				With cyclic dependency			
Grid Cells	L^1 Error	Order	L^∞ Error	Order	L^1 Error	Order	L^∞ Error	Order
256	1.11×10^{-3}	–	3.23×10^{-3}	–	3.86×10^{-3}	–	1.30×10^{-2}	–
512	3.27×10^{-4}	1.76	9.07×10^{-4}	1.83	9.58×10^{-4}	2.01	5.08×10^{-3}	1.35
1024	8.96×10^{-5}	1.87	2.53×10^{-4}	1.84	2.46×10^{-4}	1.96	1.98×10^{-3}	1.36
2048	2.30×10^{-5}	1.96	7.06×10^{-5}	1.84	6.29×10^{-5}	1.97	7.63×10^{-4}	1.37



(a) a full one ring of elements (b) a Dirichlet boundary condition (c) near the MAC grid

Figure 9: In each subfigure, all elements of the tessellated Lagrangian mesh incident to the red node are shaded green. In the case of Dirichlet boundary conditions (b), cyan triangles are created as virtual elements in order to complete a full one ring of elements. Note that the blue node in subfigure (b) is displaced away from the red node for illustration purposes only, despite the fact that they actually overlap in the implementation.

where V_c is the volume of cell c, F_c are the faces of cell c, u_f is the component of $\vec{u}$ defined at face f, $\vec{n}_f$ is the unit normal pointing in the positive direction, and $\vec{A}_f$ is the outward pointing area-weighted normal. Equation (4) defines a matrix-vector multiplication where the vector of u_f values is mapped to a vector of volume-weighted divergences. The coefficients of this matrix are $\vec{n}_f \cdot \vec{A}_f$, and thus the volume-weighted gradient can be defined via the negated transpose of this matrix obtaining an expression of the form $(\phi_{f^+} - \phi_{f^-})(\vec{n}_f \cdot \vec{A}_f)$ where ϕ_{f^+} and ϕ_{f^-} are the ϕ values of the two cells incident to face f. Since each face maps to two cells, there are two ϕ values involved in the gradient. For a grid face that connects an interior cell with a buffer cell, only one side of the face is incident to a MAC grid cell. Thus the discretization defined via the negated transpose is either $\phi_{f^+}(\vec{n}_f \cdot \vec{A}_f)$ or $-\phi_{f^-}(\vec{n}_f \cdot \vec{A}_f)$. The volume-weighted Poisson equation is then discretized on the MAC grid as

$$-\mathbf{G}^T\mathbf{M}^{-1}\mathbf{G}\boldsymbol{\phi} = \mathbf{V}_c\mathbf{f}, \tag{5}$$

where $\boldsymbol{\phi}$ and $\mathbf{f}$ are vectors of ϕ and f values, $\mathbf{V}_c$ is a diagonal matrix of V_c values, $\mathbf{G}$ is the volume-weighted gradient matrix, $-\mathbf{G}^T$ is the volume-weighted divergence matrix, and $\mathbf{M}$ is a diagonal matrix assembled from the V_c/β values defined for each face. Technically speaking, equation (5) is incomplete since no boundary conditions or other equations are defined in order to couple the boundary of the interior MAC grid region with the Lagrangian mesh.

4.2. Lagrangian Mesh Discretization

We define the *tessellated Lagrangian mesh* as the subset of particles and the grid nodes of buffer cells that are connected into a mesh as described in Section 2. On the tessellated Lagrangian mesh, equations (1), (2) and (3) are discretized using the node-based finite volume formulation of [8, 27]. At every node of the tessellated Lagrangian mesh, we have degrees of freedom for ϕ and f. In addition, $\vec{u} = \beta\nabla\phi$ is defined at the nodes in a colocated fashion. Control volumes V_n for each node are defined as $1/(d+1)$ times the volume of all incident elements, where d is the spatial dimension ($d = 2$ or $d = 3$). This evenly splits the volume of each element among all of its incident nodes.

In order to define the volume-weighted divergence, we consider the incident elements shown as green elements in Figure 9. First, $\vec{u}$ is averaged from nodes to faces along the outer boundary of all incident elements, e.g. from the two black nodes to face f in Figure 9(a). This can be expressed as

$$\vec{u}_f = \frac{1}{d}\sum_{i \in N_f} \vec{u}_i, \tag{6}$$

where N_f are the nodes incident to face f, and $\vec{u}_i$ is the $\vec{u}$ value at node i. The volume-weighted divergence is then computed by summing over the faces along the outer boundary of all incident elements (the boundary of the green region) to obtain

$$V_n \nabla \cdot \vec{u} = \frac{1}{d+1} \sum_{f \in F_b} \vec{u}_f \cdot \vec{A}_f, \tag{7}$$

where F_b are the boundary faces of the incident elements. Without $1/(d+1)$ the summation is the volume-weighted divergence for the green region, and the $1/(d+1)$ factor accounts for the fact that only a fraction of this is assigned to the node similar to the way that volume is partitioned. In summary, a node in equation (6) contributes to a face value $\vec{u}_f$ for every face connected to that node, and then in equation (7) each face contributes to the volume-weighted divergence for the two nodes element-wise opposite that face. For example, a black node in Figure 9(a) contributes to the $\vec{u}_f$ values for all its incident faces including face f, and then face f contributes to the volume-weighted divergence for both the red and blue node. Equations (6) and (7) combined define a matrix that maps a vector of $\vec{u}$ values for each node to a vector of volume-weighted divergences for each node, and the coefficients of this matrix are described in the block form by $\vec{A}_f^T/(d(d+1))$.

The volume-weighted gradient is defined via the negated transpose of this matrix obtaining

$$V_n \nabla \phi = \frac{1}{d+1} \sum_{f \in F_n} \frac{(\phi_{f^+} - \phi_{f^-})\vec{A}_f}{d}, \tag{8}$$

where F_n are the faces incident to node n, and ϕ_{f^+} and ϕ_{f^-} are the ϕ values at the two nodes element-wise opposite face f. Since a node maps to all of its incident faces in equation (6), all incident faces F_n are involved in the gradient. And since each face maps to its two element-wise opposite nodes in equation (7), both of these nodes feed back into the gradient. One can define the value of ϕ per element using a standard averaging as $\phi_t = \sum_{i \in N_t} \phi_i/(d+1)$, where N_t are the nodes incident to element t. Defining ϕ_{t,f^+} and ϕ_{t,f^-} as the ϕ values of the two elements incident to face f, we have that $\phi_{f^+} - \phi_{f^-}$ equals $(d+1)(\phi_{t,f^+} - \phi_{t,f^-})$, since the common nodes on the face cancel. Therefore, we can rewrite equation (8) as

$$V_n \nabla \phi = \sum_{f \in F_n} \frac{(\phi_{t,f^+} - \phi_{t,f^-})\vec{A}_f}{d}. \tag{9}$$

This can be reorganized by switching the order of summation to obtain

$$V_n \nabla \phi = \sum_{t \in T_n} \phi_t \frac{-\sum_{f \in F_{t,n}} \vec{A}_f}{d}, \tag{10}$$

where T_n are the mesh elements incident to node n, and $F_{t,n}$ are the faces of element t that are incident to node n. Since the sum of $\vec{A}_f$ over the faces of each element is always zero, $-\sum_{f \in F_{t,n}} \vec{A}_f$ can be replaced with $\vec{A}_{f_{t,n}}$ where $f_{t,n}$ is the face of element t that is element-wise opposite node n. Equation (10) then becomes

$$V_n \nabla \phi = \sum_{t \in T_n} \phi_t \frac{\vec{A}_{f_{t,n}}}{d}. \tag{11}$$

If a node is on the boundary of the mesh where a Dirichlet boundary condition needs to be enforced, there is not a full one ring of incident mesh elements. To complete a full one ring of elements, new virtual mesh elements are created around the node. Since the virtual elements should not change the control volume of the node, we create them by first duplicating the node at the same location and then connecting the duplicated node with all boundary mesh faces that are incident to the original node, see Figure 9(b). Now with a full

one ring of elements, the Dirichlet boundary condition is applied by replacing ϕ_t with g_t in equation (11) for each virtual element, where g_t is the boundary condition defined at the barycenter of the virtual element. Near the MAC grid, there is not a full one ring of incident mesh elements either as shown in Figure 9(c). Here, instead of defining virtual elements as in the case of Dirichlet boundary conditions, we will connect this partial discretization of the gradient to the partial discretization of the gradient on the MAC grid mentioned after equation (5) – see Section 4.3.

If a node touches the boundary of the Cartesian computational domain, Neumann boundary conditions are applied. The volume-weighted gradient is modified via

$$V_n \nabla \phi = (\mathbf{I} - \vec{n}\vec{n}^T) \sum_{t \in T_n} \phi_t \frac{\vec{A}_{f_{t,n}}}{d} + V_n h_n \vec{n}, \tag{12}$$

where $\vec{n}$ is the normal direction of the Cartesian boundary, and h_n is the Neumann boundary condition at node n. Note that multiple projections are applied if the node is at the corner or edge of the Cartesian computational domain and thus touches multiple walls. Projections are also applied to the corresponding terms in the volume-weighted divergence operator to maintain the negated transpose relationship. For a node at the intersection of the Dirichlet boundary and the boundary of the Cartesian computational domain, both Dirichlet and Neumann boundary conditions are applied. Virtual elements are created around the node but only connecting the duplicated node to its incident boundary faces that are not on the boundary of the Cartesian computational domain. The Neumann boundary condition is then applied via the projection mentioned above.

The discretized volume-weighted Poisson equation on the Lagrangian mesh can be written as

$$-\hat{\mathbf{G}}^T \hat{\mathbf{M}}^{-1} \hat{\mathbf{G}} \hat{\boldsymbol{\phi}} = \hat{\mathbf{V}}_c \hat{\mathbf{f}} + \hat{\mathbf{G}}^T \hat{\mathbf{M}}^{-1} \hat{\mathbf{G}}_d \mathbf{g} + \hat{\mathbf{G}}_n^T \hat{\boldsymbol{\beta}} \hat{\mathbf{h}}, \tag{13}$$

where $\hat{\boldsymbol{\phi}}$ and $\hat{\mathbf{f}}$ are the vectors of ϕ and f values, $\hat{\mathbf{V}}_c$ is a diagonal matrix of control volumes for each node, $\hat{\mathbf{G}}$ is the volume-weighted gradient matrix without the terms related to virtual elements and $\hat{\mathbf{G}}_d$ are the terms related to virtual elements, $\mathbf{g}$ is the vector of Dirichlet boundary conditions, $-\hat{\mathbf{G}}^T$ is the volume-weighted divergence matrix with the terms related to Neumann boundary conditions projected off and $-\hat{\mathbf{G}}_n^T$ are the terms projected off for Neumann boundary conditions, $\hat{\boldsymbol{\beta}}$ is a diagonal matrix of β values, $\hat{\mathbf{h}}$ is the vector of Neumann boundary conditions, and $\hat{\mathbf{M}}$ is a diagonal matrix with the V_n/β values defined for each component of the gradient at the nodes. Note that $\hat{\mathbf{G}}$ can be alternatively factored into $\hat{\mathbf{G}}_e \mathbf{H}$ where $\mathbf{H}$ is an interpolation matrix that averages the nodal ϕ values to the barycenters of each mesh element to obtain ϕ_t in equation (11), and $\hat{\mathbf{G}}_e$ is the element-based gradient matrix constructed from equation (11).

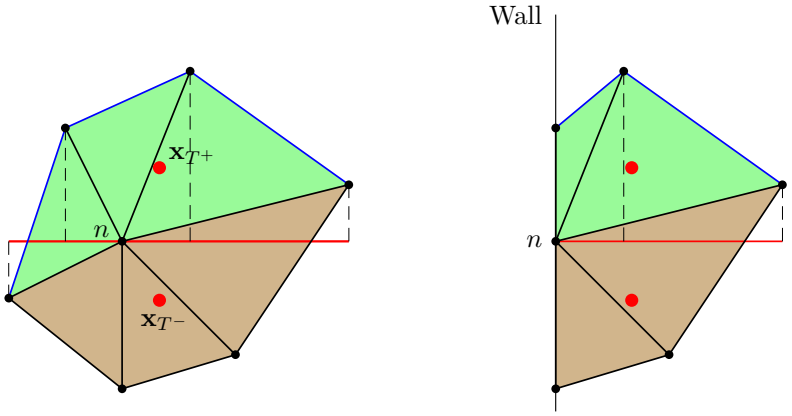
4.2.1. Lagrangian Mesh Discretization Orthogonality

Since all three components of the gradient behave similarly, we only consider the y-component. From equation (11), the y-component of the volume-weighted gradient is given by

$$(V_n \nabla \phi)^y = \sum_{t \in T_n} \phi_t \frac{A^y_{f_{t,n}}}{d}, \tag{14}$$

where $A^y_{f_{t,n}}$ is the y-component of $\vec{A}_{f_{t,n}}$. Note that $A^y_{f_{t,n}}$ equals the area of face $f_{t,n}$ projected onto the plane perpendicular to the y-axis, and ϕ_t acts on this projected region. We divide T_n into two subsets, $T^+ = \{t | A^y_{f_{t,n}} \geq 0\}$ and $T^- = \{t | A^y_{f_{t,n}} < 0\}$ shown as green and brown elements in Figure 10. The contribution of T^+ to $(V_n \nabla \phi)^y$ is

$$(V_n \nabla \phi)^y_{T^+} = \sum_{t \in T^+} \phi_t \frac{A^y_{f_{t,n}}}{d} = \phi_{T^+} A_{T^+},$$



(a) a full one ring of elements (b) near the Cartesian boundary

Figure 10: The composite face and the composite ϕ samples (red dots). Note that $A_{T^+}d$ is the entire red line, and A_{T^+} is the composite face which is half the red line.

where we have lumped all green elements into a single composite element with composite face area A_{T^+} and composite ϕ value ϕ_{T^+} given by

$$A_{T^+} = \sum_{t \in T^+} \frac{A^y_{f_{t,n}}}{d}, \qquad \phi_{T^+} = \sum_{t \in T^+} \phi_t \frac{A^y_{f_{t,n}}}{A_{T^+}d}. \tag{15}$$

ϕ_{T^+} is an affine combination of ϕ_t values, and thus its effective location $\mathbf{x}_{T^+}$ can be computed in the same fashion as equation (15) to obtain

$$\mathbf{x}_{T^+} = \sum_{t \in T^+} \mathbf{x}_t \frac{A^y_{f_{t,n}}}{A_{T^+}d}, \tag{16}$$

where $\mathbf{x}_t$ is the barycenter of element t. One can express $\mathbf{x}_t$ as

$$\mathbf{x}_t = \frac{1}{d+1}\mathbf{x}_n + \frac{d}{d+1}\mathbf{x}_{f_{t,n}},$$

where $\mathbf{x}_n$ is the location of the central node n, and $\mathbf{x}_{f_{t,n}}$ is the barycenter of face $f_{t,n}$. Substituting this into equation (16) yields

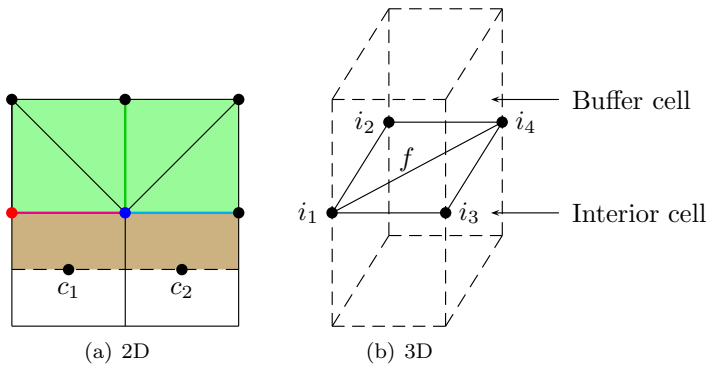
$$\mathbf{x}_{T^+} = \frac{1}{d+1}\mathbf{x}_n + \frac{d}{d+1} \sum_{t \in T^+} \mathbf{x}_{f_{t,n}} \frac{A^y_{f_{t,n}}}{A_{T^+}d}, \tag{17}$$

since $\sum_{t \in T^+} \frac{A^y_{f_{t,n}}}{A_{T^+}d} = 1$. Let $\mathbf{x}^y_{f_{t,n}}$ be the projection of $\mathbf{x}_{f_{t,n}}$ onto the plane perpendicular to the y-axis, then $\mathbf{x}^y_{f_{t,n}}$ is the centroid of $A^y_{f_{t,n}}$ and $\mathbf{x}^y_{c^+} = \sum_{t \in T^+} \mathbf{x}^y_{f_{t,n}} \frac{A^y_{f_{t,n}}}{A_{T^+}d}$ is the centroid of $A_{T^+}d$ (the red line in Figure 10). Then equation (17) can be written as

$$\mathbf{x}_{T^+} = \frac{1}{d+1}\mathbf{x}_n + \frac{d}{d+1}\mathbf{x}_{c^+}, \tag{18}$$

where $\mathbf{x}_{c^+} = \sum_{t \in T^+} \mathbf{x}_{f_{t,n}} \frac{A^y_{f_{t,n}}}{A_{T^+}d}$. And since $A_{T^+} = -A_{T^-}$ we can similarly derive

$$\mathbf{x}_{T^-} = \frac{1}{d+1}\mathbf{x}_n + \frac{d}{d+1}\mathbf{x}_{c^-}.$$



(a) 2D (b) 3D

Figure 11: Mesh structure near the boundary between the tessellated Lagrangian mesh and the MAC grid. For illustration purposes, the mesh structure in subfigure (b) is hidden except for the mesh faces lying on the intermediate face f.

Since $\mathbf{x}_{c^+}$ and $\mathbf{x}_{c^-}$ lie above and below $\mathbf{x}^y_{c^+} = \mathbf{x}^y_{c^-}$, the segment connecting them is perpendicular to the red line (plane in three spatial dimensions). Thus from the definition of $\mathbf{x}_{T^+}$ and $\mathbf{x}_{T^-}$, the segment connecting them is also perpendicular to the red line (plane) proving orthogonality.

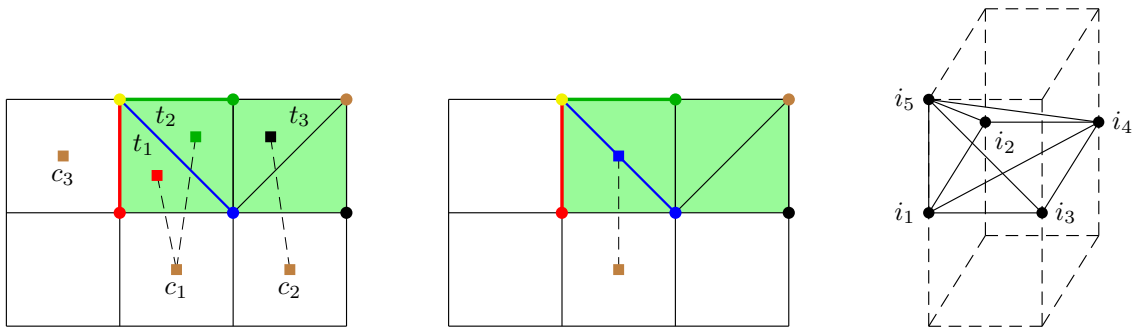
If node n is on the Dirichlet boundary, a full one ring of elements is completed by creating virtual elements, and the above discussion still applies. If node n touches the Cartesian boundary where Neumann boundary conditions are enforced, the component of the gradient along the normal direction of the Cartesian boundary is projected off. For the remaining components, as shown in Figure 10(b), the composite face lumped from the green elements is also identical to that from the brown elements, and the same argument follows.

4.3. Coupled Discretization

On the boundary between the MAC grid and the tessellated Lagrangian mesh, we define $\nabla\phi$ only on the nodes of the tessellated Lagrangian mesh and not on the faces of the MAC grid. Generally, one may define $\nabla\phi$ on both sets of degrees of freedom and connect them using continuity constraints as in [42]. However, we choose not to introduce these extra degrees of freedom and constraints. For a node incident to the MAC grid, the blue node in Figure 11(a), the ϕ values defined at the nodes of its incident mesh elements (the green elements) as well as those defined for its incident MAC grid cells (cells c_1 and c_2) form a one ring of ϕ samples. For the x component of the gradient, the ϕ values of cells c_1 and c_2 act on the MAC grid face between them and thus make no contribution to the x component of the gradient at the blue node. In fact, since both the cells adjacent to the green face are buffer cells with mesh elements, the Lagrangian mesh discretization for the x component of the gradient at the blue node proposed in Section 4.2 is sufficient. For the y-component of the gradient, since both the cells below the red and blue faces contain no mesh elements, cells c_1 and c_2 need to be considered. Because of the way the mesh is constructed in buffer cells, each incident mesh element lies entirely within a single buffer cell. Thus we lump all incident mesh elements within a single buffer cell into a single composite element. This allows the coupled discretization to be written as

$$(V_n\nabla\phi)^y = \sum_{f\in F^y_n}\left(\hat{\phi}_{f^+,n}\hat{A}_{f^+,n} - W_{f,n}\phi_{f^-}A_f\right), \tag{19}$$

where F^y_n are the relevant y-component MAC grid faces (the red and blue faces in Figure 11(a)), $\hat{\phi}_{f^+,n}$ is the composite ϕ value on the positive side of face f, $\hat{A}_{f^+,n}$ is the composite face area on the positive side of face f, ϕ_{f^-} is the MAC grid ϕ value of the cell below face f, A_f is the area of face f, and $W_{f,n}$ is a value defined for face f and node n. This distributes a fraction of the MAC grid partial gradient $-\phi_{f^-}A_f$ to node n using $W_{f,n}$ as the distribution weight. $W_{f,n}$ are defined for all boundary faces of the MAC grid and their incident nodes, and these values can be organized into a matrix $\mathbf{W}^T$. The sum of the distribution weights for each MAC grid face should be 1, i.e. columns of $\mathbf{W}^T$ should sum to 1, and this can be accomplished by



(a) before enforcing orthogonality (b) after enforcing orthogonality (c) 3D mesh structure

Figure 12: Subfigures (a) and (b) show the ϕ sample locations near the MAC grid before and after enforcing orthogonality. Subfigure (c) shows a portion of the 3D mesh structure inside a buffer cell. The cell face given by $i_1i_3i_4i_2$ separates a buffer cell (above) from an interior cell (below). For illustration purposes, only the mesh elements incident to node i_1 are shown, i.e. tetrahedra $i_1i_3i_4i_5$ and $i_1i_4i_2i_5$.

making $\mathbf{W}$ an interpolation matrix as in [42, 43]. $W_{f,n}A_f$ can be thought of as the face area on the negative side, which should equal the face area on the positive side, i.e. $\hat{A}_{f^+,n} = W_{f,n}A_f$. In two spatial dimensions $\hat{A}_{f^+,n} = A_f/2$, and thus we set $W_{f,n} = 1/2$. In three spatial dimensions, $\hat{A}_{f^+,n}$ either equals $A_f/3$ (e.g. for nodes i_1 and i_4 in Figure 11(b)) or equals $A_f/6$ (e.g. for nodes i_2 and i_3), and thus we set $W_{f,n}$ to $1/3$ or $1/6$ accordingly.

Next consider the control volume V_n for this hybrid node. The x component of the gradient only has contributions from the incident mesh elements, and thus the corresponding control volume is defined to be $1/(d+1)$ times the volume of the incident green elements as in Section 4.2. The y-component of the gradient involves both the incident mesh elements and the incident MAC grid cells. The volume fraction from the tessellated Lagrangian mesh is computed in the same fashion as for the x component. The MAC grid dual cell volume within each brown region equals $V_c/2$, and we assign this to the nodes in the same way $\mathbf{W}^T$ assigns the MAC grid face area to the nodes, i.e. the total MAC grid volume assigned to the node equals $\sum_{f\in F_n^y} W_{f,n}V_c/2$.

MAC grid cell centers now contribute to the gradients defined on the nodes of the tessellated Lagrangian mesh on adjacent MAC grid cell faces bordering buffer cells, and thus these nodes contribute to the divergences of the MAC grid cells. The contribution from one of these boundary MAC grid cell faces to the volume-weighted divergence can be expressed as $\sum_{n\in N_f}(W_{f,n}\vec{u}_n)\cdot\vec{A}_f$, which takes a weighted average of the nodal $\vec{u}$ values and dot products it with the area-weighted normal of the face. In two spatial dimensions, with $W_{f,n} = \frac{1}{2}$ it averages the $\vec{u}$ values of the two nodes, whereas in Figure 11(b) it would take one third of nodes i_1 and i_4 and one sixth of nodes i_2 and i_3. Recall that we have chosen to discretize the gradients on the boundary Lagrangian mesh nodes and not on the boundary MAC grid faces. Because of this, MAC grid degrees of freedom contribute to the gradients at the boundary nodes, but the boundary nodes do not contribute to the gradients on the MAC grid. Thus, the nodal volume-weighted divergence computed via the negated transpose of the gradient has no contribution from the MAC grid – and therefore the nodal divergences are still computed using the one ring on the tessellated Lagrangian mesh only.

4.3.1. An Orthogonal Coupled Discretization – 2D

Equation (19) requires further modification of $\hat{\phi}_{f^+,n}$ in order to obtain an orthogonal discretization. First consider the y-component of the gradient at the blue node in Figure 12(a). Similar to the discussion for Figure 11(a), the mesh elements incident to the blue node inside the left green cell (elements t_1 and t_2) are lumped into a single composite element with a composite ϕ value $\hat{\phi}_{f^+,b}$ and a composite face area $\hat{A}_{f^+,b}$. Since the red face is vertical, element t_1 does not contribute to the y-component of the gradient, leaving only the contribution from element t_2. Thus $\hat{\phi}_{f^+,b}$ is located at the centroid of element t_2 (the green square), and

$\hat{A}_{f^+,b}$ equals half of the region projected from the green face onto the plane perpendicular to the y-axis, i.e. $A_f/2$. Similar statements hold for the green cell to the right, and the composite ϕ value (the black square) is located at the centroid of element t_3. The net effect of the green and black squares with cells c_1 and c_2 averages out to be orthogonal. However, in the case of the red node with MAC grid cell c_3 to its left, there is no chance for symmetric cancellation. Therefore, we approach symmetry in a cell by cell (i.e. buffer cell by buffer cell) manner, without considering any potential symmetric cancellation between cells.

A cell composed by t_1 and t_2 is a typical buffer cell up to symmetries. As discussed above, the y-component of the gradient within the cell for the blue node is obtained by connecting the green and brown squares. For the red node, its composite ϕ value $\hat{\phi}_{f^+,r}$ is given by the centroid of element t_1 denoted by the red square, and its composite face area $\hat{A}_{f^+,r}$ is given by one half of the region projected from the blue face onto the plane perpendicular to the y-axis, i.e. $A_f/2$. Whether considering the gradient at the red node or at the blue node, we combine elements t_1 and t_2 averaging the net location of the red and green squares to obtain the blue square in Figure 12(b) resulting in an orthogonal discretization. That is in equation (19) for the blue node we replace $\hat{\phi}_{f^+,b}$ with $\frac{1}{2}(\hat{\phi}_{f^+,r} + \hat{\phi}_{f^+,b})$, and in equation (19) for the red node we similarly replace $\hat{\phi}_{f^+,r}$ with $\frac{1}{2}(\hat{\phi}_{f^+,r} + \hat{\phi}_{f^+,b})$. This is equivalent to collecting the net effect of $\hat{\phi}_{f^+,r}$ and $\hat{\phi}_{f^+,b}$ as $(\hat{\phi}_{f^+,r} + \hat{\phi}_{f^+,b})\frac{A_f}{2}$ since $\hat{A}_{f^+,r} = \hat{A}_{f^+,b} = \frac{A_f}{2}$, and distributing it back to the two nodes using $\mathbf{W}^T$ as $\frac{1}{2}(\hat{\phi}_{f^+,r} + \hat{\phi}_{f^+,b})\frac{A_f}{2}$. In summary in order to obtain an orthogonal discretization, equation (19) is modified to

$$(V_n \nabla \phi)^y = \sum_{f \in F_n^y} \left(\left(\sum_{i \in N_f} W_{f,i} \hat{\phi}_{f^+,i} \right) \hat{A}_{f^+,n} - W_{f,n} \phi_{f^-} A_f \right).$$

Defining $\hat{\phi}_{f^+} = \sum_{i \in N_f} W_{f,i} \hat{\phi}_{f^+,i}$ and noting that $\hat{A}_{f^+,n} = W_{f,n} A_f$, the above equation can be rewritten as

$$(V_n \nabla \phi)^y = \sum_{f \in F_n^y} (\hat{\phi}_{f^+} - \phi_{f^-}) W_{f,n} A_f. \tag{20}$$

Here $\hat{\phi}_{f^+}$ can be thought of as an effective ϕ value of the entire buffer cell.

The volume-weighted divergence is modified accordingly to maintain the negated transpose relationship. Since equation (20) does not change the contribution from the MAC grid to the gradients on boundary nodes, the contribution from boundary nodes to the divergence on the MAC grid is also unchanged. However, the divergence on the tessellated Lagrangian mesh is modified. Note that $\hat{\phi}_{f^+}$ is a weighted average of the ϕ values from the four nodes of the buffer cell, which means that the gradient at each node of a boundary face obtains contributions from every node of the buffer cell. Therefore the divergence at each node of the buffer cell receives contributions from each node of its boundary face. For example in Figure 12(a), the divergence at the green node would normally include contributions from the green, yellow, blue and brown nodes. However, with our new orthogonal discretization it would also include the contributions from the red and black nodes, since the green node is used to calculate the gradients at the red and black nodes.

4.3.2. An Orthogonal Coupled Discretization – 3D

In three spatial dimensions, orthogonality is also enforced independently within each buffer cell. As pointed out in [3], a cube can be decomposed into either five or six tetrahedra with each exterior face cut diagonally by two triangles. The meshing algorithm described in Section 2 randomly perturbs the positions of all relevant particles as well as the nodes of buffer cells so that no more than four particles or nodes are on the same circumsphere. It then generates a tetrahedron whenever its circumsphere contains no other particles or nodes. Since the perturbation is random, all possible decompositions of a buffer cell may appear.

Consider the y-component of the gradient at the nodes of the intermediate cell face in Figure 12(c). For each of these nodes, we lump its incident mesh elements inside the buffer cell to obtain a composite face and a composite ϕ value. Note that these tetrahedral mesh elements have three types of triangle faces. The first type of faces are incident to the node and lie on the cell faces of the buffer cell, e.g. $i_1 i_5 i_3$, $i_1 i_3 i_4$, $i_1 i_4 i_2$ and

Table 11: The order of accuracy for the 2D Poisson equation test.

Grid Cells	L^1 Error	Order	L^∞ Error	Order
256	3.35×10^{-5}	–	1.39×10^{-2}	–
512	6.84×10^{-6}	2.29	4.18×10^{-3}	1.73
1024	1.51×10^{-6}	2.18	1.19×10^{-3}	1.82
2048	3.47×10^{-7}	2.12	3.08×10^{-4}	1.95

$i_1 i_2 i_5$ for node i_1. The second type of faces are common faces between mesh elements, e.g. $i_1 i_4 i_5$. The third type of faces are those element-wise opposite the node, e.g. $i_3 i_4 i_5$ and $i_4 i_2 i_5$. Ignoring the second type, the first and third types of faces form a polyhedron. Thus, the sum of outward pointing area-weighted normals over the first type of faces equals negative the sum over the third type. Therefore, the projection of the mesh faces element-wise opposite the node (the third type) onto the plane perpendicular to the y-axis is equal to the projection of those lying on the cell faces of the buffer cell (the first type). The only non-zero contributions come from those lying on the top and bottom cell faces of the buffer cell (in this case $i_1 i_3 i_4$ and $i_1 i_4 i_2$). In general, this is either one or two triangles, and the composite face area is either $A_f/6$ or $A_f/3$ respectively.

Next consider the locations of the composite ϕ values. With regard to orthogonality, only the projections of these locations onto the plane perpendicular to the y-axis are of concern. Define $\hat{\mathbf{x}}^y_{f^+,i_1}$ to be such a projection for node i_1. The derivation of equation (18) shows that $\hat{\mathbf{x}}^y_{f^+,i_1}$ is located three-fourths of the way from node i_1 to the centroid of the region projected from all the mesh faces element-wise opposite node i_1 onto the plane perpendicular to the y-axis. As discussed above, this projected region equals the incident triangles lying on the top and bottom cell faces of the buffer cell (in this case $i_1 i_3 i_4$ and $i_1 i_4 i_2$). Therefore, $\hat{\mathbf{x}}^y_{f^+,i_1}$ is located three-fourths of the way from node i_1 to the center of the intermediate cell face. In general, the centroid of the region projected from the element-wise opposite faces is either located at the center of the intermediate cell face (for nodes i_1 and i_4) or located at the centroid of one of the two triangles on that cell face (for nodes i_2 and i_3). Based on this observation, we can also compute $\hat{\mathbf{x}}^y_{f^+,i_2}$, $\hat{\mathbf{x}}^y_{f^+,i_3}$, and $\hat{\mathbf{x}}^y_{f^+,i_4}$. Similar to the modification in two spatial dimensions, we collect the net effect from the four nodes as

$$\sum_{i=i_1}^{i_4} \hat{\phi}_{f^+,i} \hat{A}_{f^+,i} = \left(\hat{\phi}_{f^+,i_1} + \hat{\phi}_{f^+,i_4}\right) \frac{A_f}{3} + \left(\hat{\phi}_{f^+,i_2} + \hat{\phi}_{f^+,i_3}\right) \frac{A_f}{6} = \hat{\phi}_{f^+} A_f,$$

and distribute $\hat{\phi}_{f^+} A_f$ back to all four nodes using $\mathbf{W}^T$.

4.4. Numerical Results

We have implemented two tests from [22]. The first test uses a computation domain of $[-1.5, 1.5] \times [0, 3]$ in two spatial dimensions, and $\partial\Omega$ is parameterized by $(x(\theta), y(\theta))$ where $x(\theta) = (1 + .4\cos(5\theta))\cos(\theta)$ and $y(\theta) = (1 + .4\cos(5\theta))\sin(\theta)$ with $\theta \in [0, 2\pi]$, see Figure 13. An exact solution of $\phi = e^x(x^2 \sin(y) + y^2)$ and $\beta = 2 + \sin(xy)$ is used in Ω with Dirichlet boundary conditions enforced on $\partial\Omega$. The second test uses a computational domain of $[0, 1] \times [0, 1] \times [0, 1]$ in three spatial dimensions, and $\partial\Omega$ is a sphere centered at $(.5, .5, .5)$ with radius .3. An exact solution of $\phi = \sin(4\pi x)\sin(4\pi y)\sin(4\pi z)$ and $\beta = xyz$ is used in Ω with Dirichlet boundary conditions enforced on $\partial\Omega$. Tables 11 and 12 show second order accuracy. To examine the convergence behavior with Neumann boundary conditions, we modified the above tests by cutting the computational domain in half so that part of Ω touches the computational domain boundary where Neumann boundary conditions are applied. Specifically, the computational domain is $[-1.5, 1.5] \times [1.5, 3]$ in two spatial dimensions and $[0, 1] \times [.5, 1] \times [0, 1]$ in three spatial dimensions. Tables 13 and 14 show second order accuracy. Note that all tests in this section are solved using the conjugate gradient method, and the CG iteration stops when the L^∞ norm of residual is below 10^{-8}. Table 15 shows the relation between the number of CG iterations and the grid resolution for each test.

Table 12: The order of accuracy for the 3D Poisson equation test.

Grid Cells	L^1 Error	Order	L^∞ Error	Order
128	9.65×10^{-4}	–	1.20×10^{-1}	–
256	1.55×10^{-4}	2.64	3.84×10^{-2}	1.64
512	3.40×10^{-5}	2.19	1.08×10^{-2}	1.83
1024	4.35×10^{-6}	2.97	2.41×10^{-3}	2.17

Table 13: The order of accuracy for the 2D Poisson equation test with Neumann boundary conditions.

Grid Cells	L^1 Error	Order	L^∞ Error	Order
256	1.10×10^{-4}	–	1.16×10^{-2}	–
512	2.03×10^{-5}	2.44	2.72×10^{-3}	2.10
1024	4.14×10^{-6}	2.29	7.30×10^{-4}	1.90
2048	9.09×10^{-7}	2.19	1.76×10^{-4}	2.05

Table 14: The order of accuracy for the 3D Poisson equation test with Neumann boundary conditions.

Grid Cells	L^1 Error	Order	L^∞ Error	Order
128	1.08×10^{-3}	–	1.08×10^{-1}	–
256	1.74×10^{-4}	2.63	2.93×10^{-2}	1.88
512	2.88×10^{-5}	2.60	8.43×10^{-3}	1.80
1024	4.98×10^{-6}	2.53	2.09×10^{-3}	2.02

Table 15: The number of CG iterations for the Poisson equation tests.

Grid Cells	2D	3D	2D with Neumann	3D with Neumann
128	–	439	–	422
256	660	550	643	654
512	1275	1039	1189	1095
1024	2442	2014	2284	1963
2048	4693	–	4387	–

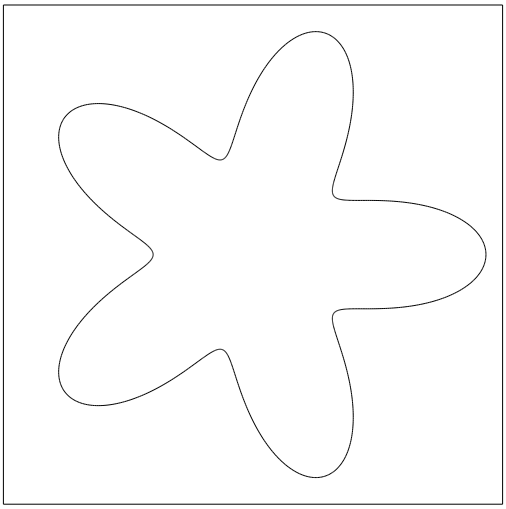


Figure 13: The interface of Ω for the 2D Poisson equation test.

5. Heat Equation

Consider the heat equation

$$\frac{\partial \phi(\vec{x},t)}{\partial t} = \nabla \cdot (\beta \nabla \phi(\vec{x},t)), \qquad \vec{x} \in \Omega \tag{21}$$

$$\phi(\vec{x},t) = g(\vec{x},t), \qquad \vec{x} \in \partial\Omega_D \tag{22}$$

$$\vec{n} \cdot \nabla \phi(\vec{x},t) = h(\vec{x},t), \qquad \vec{x} \in \partial\Omega_N \tag{23}$$

where β is the spatially constant diffusion coefficient (spatially varying viscosity is not considered). We test boundary conditions similar to what would apply to the viscosity term in the incompressible flow problems of interest, i.e. Neumann boundary conditions everywhere except on the Cartesian computational domain boundary where Dirichlet boundary conditions are applied.

We summarize the discretization of the volume-weighted Poisson equation of Section 4 via

$$-\bar{\mathbf{G}}^T \bar{\mathbf{M}}^{-1} \bar{\mathbf{G}} \bar{\boldsymbol{\phi}} = \bar{\mathbf{V}} \bar{\mathbf{f}} + \bar{\mathbf{b}}, \tag{24}$$

where $\bar{\boldsymbol{\phi}} = \begin{pmatrix} \boldsymbol{\phi} \\ \hat{\boldsymbol{\phi}} \end{pmatrix}$ is the combined vector of unknowns, $\bar{\mathbf{b}}$ is the combined vector of boundary conditions, and the other terms are defined likewise. Then the backward Euler time integration scheme is given by

$$(\bar{\mathbf{V}} + \Delta t \bar{\mathbf{G}}^T \bar{\mathbf{M}}^{-1} \bar{\mathbf{G}}) \bar{\boldsymbol{\phi}}^{n+1} = \bar{\mathbf{V}} \bar{\boldsymbol{\phi}}^n - \Delta t \bar{\mathbf{b}}^{n+1}. \tag{25}$$

Note that equation (25) is a volume-weighted form of equation (21) in order to maintain symmetry along the lines of the volume-weighted divergence $-\bar{\mathbf{G}}^T$. For second order accuracy the trapezoidal rule time integration scheme is given by

$$(\bar{\mathbf{V}} + \frac{\Delta t}{2} \bar{\mathbf{G}}^T \bar{\mathbf{M}}^{-1} \bar{\mathbf{G}}) \bar{\boldsymbol{\phi}}^{n+1} = (\bar{\mathbf{V}} - \frac{\Delta t}{2} \bar{\mathbf{G}}^T \bar{\mathbf{M}}^{-1} \bar{\mathbf{G}}) \bar{\boldsymbol{\phi}}^n - \frac{\Delta t}{2} (\bar{\mathbf{b}}^n + \bar{\mathbf{b}}^{n+1}). \tag{26}$$

Dirichlet boundary conditions are applied in the same fashion as in Section 4.2. To enforce the Neumann boundary condition at a node, we first compute the boundary normal at the node by summing over the outward pointing area-weighted normals of the boundary mesh faces incident to that node and normalizing the result. If the node is at the intersection of Dirichlet and Neumann boundaries, only those boundary faces

Table 16: The order of accuracy for the 2D heat equation test.

Grid Cells	L^1 Error	Order	L^∞ Error	Order
256	9.29×10^{-6}	–	2.82×10^{-3}	–
512	1.67×10^{-6}	2.48	8.62×10^{-4}	1.71
1024	3.41×10^{-7}	2.29	2.11×10^{-4}	2.03
2048	7.48×10^{-8}	2.19	5.93×10^{-5}	1.83

Table 17: The order of accuracy for the 3D heat equation test.

Grid Cells	L^1 Error	Order	L^∞ Error	Order
128	5.83×10^{-7}	–	1.37×10^{-4}	–
256	1.13×10^{-7}	2.37	3.79×10^{-5}	1.85
512	2.21×10^{-8}	2.35	1.02×10^{-5}	1.89

lying on the Neumann boundary are taken into account. The Neumann boundary condition is then enforced by projecting orthogonal to the boundary normal as described in Section 4.2.

We have implemented two tests from [22]. The first test uses the same computational domain as for Table 13. An exact solution of $\phi = e^{-2t}\sin(x)\sin(y)$ and $\beta = 1$ is used in Ω with Neumann boundary conditions enforced on $\partial\Omega$ except on the Cartesian computational domain boundary where Dirichlet boundary conditions are applied. The second test uses a computational domain of $[0, .5] \times [.25, .5] \times [0, .5]$ in three spatial dimensions, and $\partial\Omega$ is a semisphere centered at $(.25, .25, .25)$ with a radius of .15. An exact solution of $\phi = e^{-3t}\sin(x)\sin(y)\sin(z)$ and $\beta = 1$ is used in Ω with Neumann boundary conditions enforced on $\partial\Omega$ except on the Cartesian computational domain boundary where Dirichlet boundary conditions are applied. Both tests integrate the solution from time $t = 0$ to $t = .1$ using the trapezoidal rule time integration scheme and the time step size $\Delta t = .5\Delta x$. Tables 16 and 17 show second order accuracy. The conjugate gradient method with a Jacobi preconditioner is used in both tests. The same method is also used in the other tests in this section.

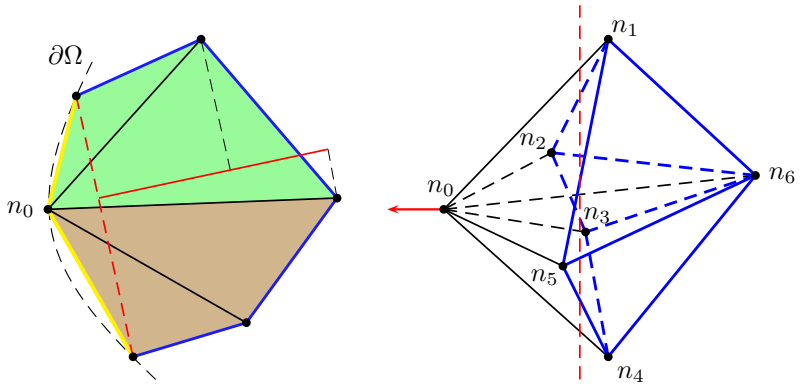
5.1. Neumann Boundary Orthogonality

As shown in Section 4.2.1, the orthogonality along a certain direction holds if the positive and negative composite faces along that direction are identical. For Neumann boundary conditions on the Cartesian boundary as shown in Figure 10(b), the positive and negative composite faces are identical because the mesh abuts a flat surface perpendicular to the boundary normal. In the case of Neumann boundary conditions on the free surface, this is not true, and orthogonality is not obtained in three spatial dimensions – *although orthogonality is obtained in two spatial dimensions.*

In two spatial dimensions as shown in Figure 14(a), the red dashed line is perpendicular to the boundary normal at n_0 because the boundary normal is computed by summing over the area-weighted normals of the yellow faces which form a closed polygon with the red dashed line. Since the gradient along the boundary normal is projected off, only the gradient perpendicular to the boundary normal needs to be examined for orthogonality (i.e. along the normal direction of the red solid line). Note that we no longer consider the gradient independently along Cartesian directions when using Neumann boundary conditions to project out

Table 18: The order of accuracy for the 2D Poisson equation test with Neumann boundary conditions on $\partial\Omega$ and Dirichlet boundary conditions on the Cartesian boundary.

Grid Cells	L^1 Error	Order	L^∞ Error	Order
256	1.28×10^{-4}	–	2.27×10^{-2}	–
512	2.01×10^{-5}	2.67	6.70×10^{-3}	1.76
1024	4.45×10^{-6}	2.18	1.53×10^{-3}	2.13
2048	1.22×10^{-6}	1.87	3.32×10^{-4}	2.20



(a) 2D (b) 3D

Figure 14: Mesh structure near a node on the free boundary boundary. Subfigure (b) shows all the tetrahedra incident to node n_0 (the five tetrahedra connecting n_0 with each blue triangle). Nodes n_0 to n_5 lie on $\partial\Omega$, and node n_6 is inside Ω. Note that nodes n_1 to n_5 are not coplanar. The red arrow shows the boundary normal at node n_0, and the red dashed line represents a plane with normal n_0 (drawn as a line since it is parallel to the view direction).

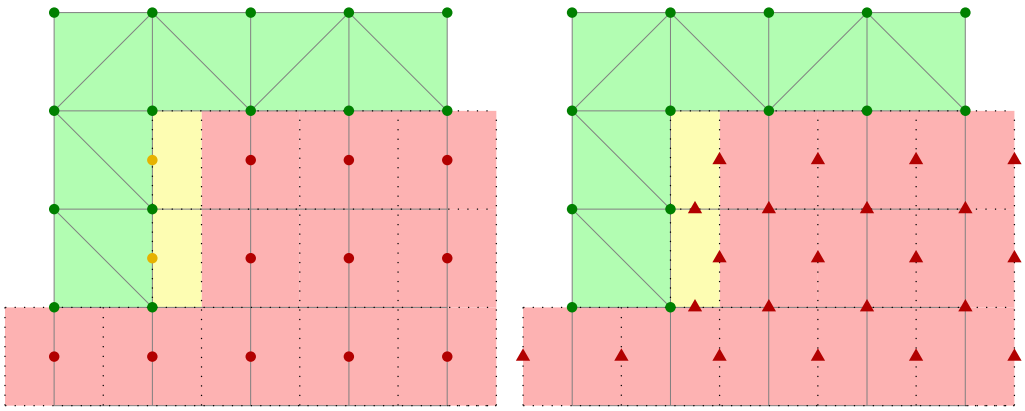
Table 19: The order of accuracy for the 3D Poisson equation test with Neumann boundary conditions on $\partial\Omega$ and Dirichlet boundary conditions on the Cartesian boundary.

Grid Cells	L^1 Error	Order	L^∞ Error	Order
128	1.07×10^{-3}	–	1.03×10^{-1}	–
256	1.79×10^{-4}	2.58	3.15×10^{-2}	1.70
512	3.57×10^{-5}	2.33	7.32×10^{-3}	2.11

a direction. The half one ring of element-wise opposite mesh faces for node n_0 (the blue faces) can be completed by the red dashed line forming a closed polygon similar to the case in Figure 10(b), so the blue faces of the green elements and those of the brown elements are projected onto the same region on the red solid line, i.e. the positive and negative composite faces are identical. Thus the orthogonality holds in two spatial dimensions.

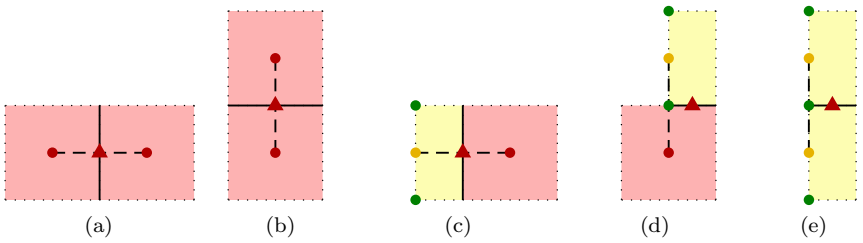
In three spatial dimensions as shown in Figure 14(b), the blue triangles are the mesh faces element-wise opposite node n_0, and the elements incident to node n_0 are those connecting the blue triangles with n_0. The red arrow is the boundary normal at node n_0, and the red dashed line represents a plane with normal n_0 (drawn as a line since it is parallel to the view direction of the figure). To enforce the Neumann boundary condition, the gradient along the red arrow is projected off, leaving the gradients parallel to the plane given by the red dashed line. Since nodes n_1 to n_5 are not coplanar (which is common), there is a discrepancy between the positive and negative composite faces, i.e. the discretization is not orthogonal. Fortunately, this discrepancy converges to zero under mesh refinement if $\partial\Omega$ is smooth, since the boundary mesh faces incident to n_0 converge to a flat surface similar to the case in Figure 10(b) where orthogonality holds.

To test the order of accuracy of the spatial discretization with Neumann boundary conditions on the free surface, we modified the Poisson equation tests from Tables 13 and 14. Instead of Dirichlet boundary conditions, Neumann boundary conditions are enforced on $\partial\Omega$. In addition, Dirichlet boundary conditions are applied on the Cartesian computational domain boundary. Table 18 shows second order accuracy in two spatial dimensions as expected. Table 19 shows second order accuracy in three spatial dimensions, which indicates the tendency toward orthogonality under mesh refinement.



(a) ϕ (b) $\nabla\phi$

Figure 15: The ϕ degrees of freedom are defined at the green, yellow and red dots shown in Subfigure (a). The red boxes delineated by the dotted lines are the control volumes associated with the red dots, while the yellow boxes are the control volumes associated with the yellow dots. $\nabla\phi$ is computed at the green dots as well as at the red triangles shown in Subfigure (b).



(a) (b) (c) (d) (e)

Figure 16: The volume-weighted gradients are defined at the common boundaries between MAC grid control volumes and are computed by differencing the associated red or yellow dots weighted by the area of their common face illustrated by the black line segments. In Subfigures 16(c), 16(d) and 16(e), the ϕ values at the yellow dots are interpolated from their neighboring nodes of the tessellated Lagrangian mesh (the green dots) using the $\mathbf{W}$ matrix.

5.2. MAC Grid Discretization

In order to address a spatially constant viscosity term in the Navier-Stokes equations, we need to store ϕ on the MAC grid faces instead of at the MAC grid cell centers. Without loss of generality, we consider the x-component MAC grid faces denoted by the red and yellow dots in Figure 15(a). On the tessellated Lagrangian mesh, ϕ is still stored at the nodes. We define two types of control volumes on the MAC grid: the dual cells centered at each red dot, and the half dual cells adjacent to each yellow dot.

5.2.1. Volume-Weighted Gradient

On the MAC grid, we define the volume-weighted gradients of ϕ at the common boundaries between MAC grid control volumes denoted by the red triangles in Figure 15(b). Different cases of discretization are illustrated in Figure 16. The gradient between two red dots (Figures 16(a) and 16(b)) is defined in standard fashion as explained in Section 4.1. The gradient in the x-direction between a red and a yellow dot (Figure 16(c)) is also defined in standard fashion except that the ϕ value at the yellow dot is interpolated from its neighboring nodes of the tessellated Lagrangian mesh using the $\mathbf{W}$ matrix defined in Section 4.3. The gradient in the y-direction between a red and a yellow dot (Figure 16(d)) or between two yellow dots (Figure 16(e)) is defined similarly but with only one half the area of a regular MAC grid face. On the tessellated Lagrangian mesh, the volume-weighted gradients are still defined at the nodes, and the Lagrangian mesh discretization as explained in Sections 4.2, 4.3.1 and 4.3.2 remains unchanged. However, at the nodes

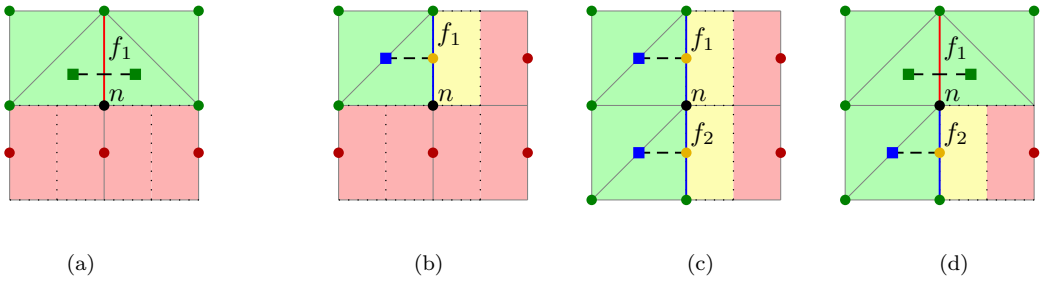


Figure 17: The volume-weighted gradient in the x-direction at a node incident to the MAC grid (the black dot in each subfigure). The red line segments are x-component MAC grid faces between two buffer cells, and the blue line segments are x-component MAC grid faces between a buffer cell and an interior MAC grid cell. The ϕ values at the green and blue squares as well as the ϕ values at the yellow dots are interpolated from neighboring ϕ samples, and the gradient at each black dot is computed by differencing these interpolated values across each red and/or blue face.

incident to the MAC grid, this discretization only computes a partial gradient. In order to compute a full gradient at those nodes, we proceed as follows.

First, we consider the gradient in the x-direction at the black nodes in Figure 17 and handle each subfigure respectively as follows. In the case of Figure 17(a), the discretization as in Section 4.2 is sufficient to compute the gradient in the x-direction as explained in Section 4.3. Within each buffer cell, the mesh elements incident to the black node can be lumped into a single composite element to obtain a composite ϕ value located at the green squares. For details, see the discussion before equation (19). In this configuration, the gradient in the x-direction at node n does not depend on the MAC grid degrees of freedom. In the case of Figure 17(b), the Lagrangian mesh discretization as in Sections 4.3.1 and 4.3.2 is used to interpolate the ϕ values from the nodes of the buffer cell to the location of the blue square. For details, see the derivation towards equation (20) and compare this blue square to the blue square in Figure 12(b). To the right of the blue square, the yellow and red dots both lie on the same line passing through the blue square perpendicular to the blue face f_1, i.e. differencing the blue square with either the yellow or red dot would both result in an orthogonal discretization. However, it is complicated to merge the yellow control volume with the red control volume when considering the volume-weighted divergence, and since the gradient and divergence are related by the transpose operator, the yellow dot and its control volume are used for our discretization. We compute the gradient in the x-direction at the black node as $(\phi_{f_1^+} - \phi_{f_1^-})W_{f_1,n}A_{f_1}$ where ϕ_{f^+} and ϕ_{f^-} are the ϕ values at the yellow dot and the blue square respectively. This discretization can be extended to the case of Figure 17(c) by similarly defining a blue square for each buffer cell and then computing the gradient in the x-direction at the black node as $\sum_{f=f_1}^{f_2}(\phi_{f^+} - \phi_{f^-})W_{f,n}A_f$. In the case of Figure 17(d), the two green squares are defined as in Figure 17(a), and the blue square is defined as in Figure 17(b). The gradient in the x-direction at the black node can then be written as $\sum_{f=f_1}^{f_2}(\phi_{f^+} - \phi_{f^-})W_{f,n}A_f$ where the terms are defined as in Figures 17(a) and 17(b).

Next, consider the gradient in the y-direction at the black nodes shown in Figure 18 (the z-direction in three spatial dimensions is handled similarly). In the case of Figure 18(a), the discretization in Section 4.2 is sufficient to compute the gradient in the y-direction similar to Figure 17(a). In the case of Figure 18(b), the blue square is defined similarly as in Figure 17(b). We average the ϕ values from the two neighboring red dots to the brown square located at the center of the interior cell below face f_1. The gradient in the y-direction at the black node can then be written as $(\phi_{f_1^+} - \phi_{f_1^-})W_{f_1,n}A_{f_1}$ where $\phi_{f_1^+}$ and $\phi_{f_1^-}$ are the ϕ values at the blue and brown squares respectively. In the case of Figure 18(c), we simply combine two separate discretizations along the lines of Figure 18(b) just as in Figure 17(c). In the case of Figure 18(d), we combine the discretizations from Figures 18(a) and 18(b) except that the brown square is computed by averaging a yellow and red dot.

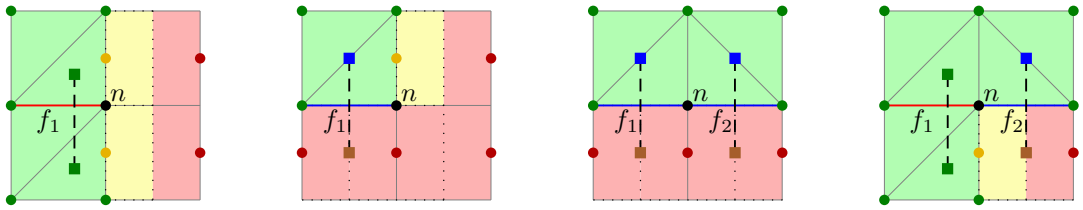


Figure 18: The volume-weighted gradient in the y-direction at a node incident to the MAC grid (the black dot in each subfigure). The red line segments are y-component MAC grid faces between two buffer cells, and the blue line segments are y-component MAC grid faces between a buffer cell and an interior MAC grid cell. The ϕ values at the green, blue and brown squares are interpolated from neighboring ϕ samples, and the gradient at each black dot is computed by differencing these interpolated values across each red and/or blue face.

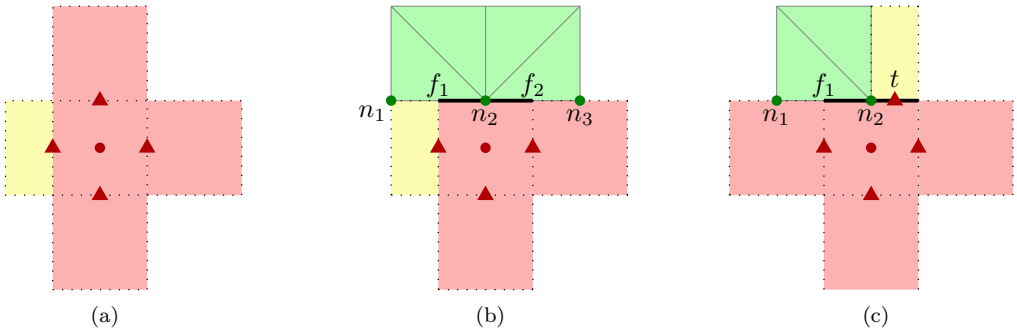


Figure 19: The volume-weighted divergence at a red dot is computed from the neighboring $\vec{u}$ samples denoted by the red triangles (Subfigure (a)) or denoted by both the red triangles and the green dots (Subfigures (b) and (c)).

5.2.2. Volume-Weighted Divergence

The volume-weighted divergence of a quantity $\vec{u}$ (e.g. $\vec{u} = \beta\nabla\phi$) is defined via the negated transpose of the gradient. On the tessellated Lagrangian mesh, since the nodal ϕ values contribute to the nodal gradients using the Lagrangian mesh discretization as explained in Sections 4.2, 4.3.1 and 4.3.2, the nodal $\vec{u}$ values also contribute to the nodal divergences as explained in those sections. On the MAC grid, the volume-weighted divergences are computed at the red and yellow dots colocated with the MAC grid ϕ samples in Figure 15(a). As shown in Figures 16(c), 16(d) and 16(e), the nodal ϕ values of the tessellated Lagrangian mesh contribute to the gradients on the MAC grid via the definition of the yellow dots interpolated from the green dots using $\mathbf{W}$. This means that the divergences at those green dots should be modified based on the $\vec{u}$ values from the MAC grid locations designated by the red triangles in those subfigures. We accomplish this in a two-step illustrative construction which first defines divergence on the yellow dots and then map that divergence back to the appropriate green parents using $\mathbf{W}^T$. Conversely, the red and yellow degrees of freedom on the MAC grid contribute to the gradients on the tessellated Lagrangian mesh as illustrated in Figures 17 and 18, and thus the divergence on the MAC grid depends on the tessellated Lagrangian mesh as dictated by the transpose operator.

Consider the divergence at the red dot shown in Figure 19(a). Since the red dot contributes to the gradients at the surrounding faces in standard fashion, the divergence at the red dot is computed using the $\vec{u}$ values at those faces also in standard fashion. In the case of Figure 19(b) the ϕ value at the red dot contributes to the gradients at nodes n_1, n_2 and n_3 with weights $-W_{f_1,n_1}A_f/2$, $-(W_{f_1,n_2}+W_{f_2,n_2})A_f/2$ and $-W_{f_2,n_3}A_f/2$ respectively as illustrated in Figure 18. Via the negated transpose of the gradient, the same negated weights $W_{f_1,n_1}A_f/2$, $(W_{f_1,n_2}+W_{f_2,n_2})A_f/2$ and $W_{f_2,n_3}A_f/2$ are used to map from the $\vec{u}$ values at

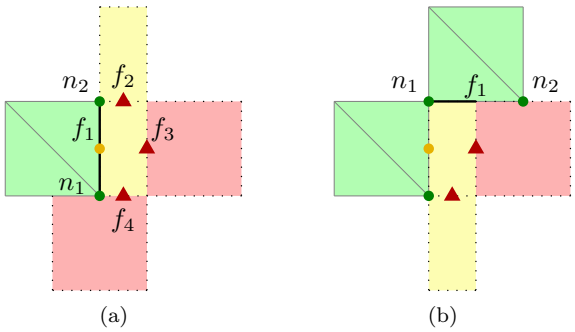


Figure 20: The volume-weighted divergence at a yellow dot is computed from its neighboring $\vec{u}$ samples denoted by the green dots and the red triangles.

nodes n_1, n_2 and n_3 to the divergence at the red dot. Note that the sum of these weights equals A_f which is the area of the black face and consistent with the standard discretization. In the case of Figure 19(c), half of the black face is handled in a standard fashion using the $\vec{u}$ value at the red triangle t, and the other half proceeds as in Figure 19(b). The ϕ value at the red dot contributes to the gradients at nodes n_1 and n_2 with weights $-W_{f_1,n_1}A_f/2$ and $-W_{f_1,n_2}A_f/2$ as seen in Figure 18(b). Thus via the negated transpose of the gradient, the negated weights $W_{f_1,n_1}A_f/2$ and $W_{f_1,n_2}A_f/2$ are used to map from the $\vec{u}$ values at nodes n_1 and n_2 to the divergence at the red dot. Similar to the previous case, the sum of these weights equals half the area of the black face.

Next, consider the divergences at the yellow dots shown in Figure 20. In the case of Figure 20(a), the ϕ value at the yellow dot contributes to the gradient on face f_3 in standard fashion, and thus the $\vec{u}$ value on face f_3 contributes to the divergence at the yellow dot in standard fashion as well. Faces f_2 and f_4 are similar except that only half the area of a regular MAC grid face is used. As seen in Figure 17(b), the ϕ value at the yellow dot contributes to the gradients at nodes n_1 and n_2 with weights $W_{f_1,n_1}A_{f_1}$ and $W_{f_1,n_2}A_{f_1}$ respectively. Thus via the negated transpose of the gradient, the $\vec{u}$ values at nodes n_1 and n_2 contribute to the divergence at the yellow dot with the same negated weights $-W_{f_1,n_1}A_{f_1}$ and $-W_{f_1,n_2}A_{f_1}$ which sum to the area of face f_1. In the case of Figure 20(b), three of the faces are handled as in Figure 20(a), and only the top face differs. There considering Figure 18(d), we see that the ϕ value at the yellow dot contributes to the gradients at nodes n_1 and n_2 with weights $-W_{f_1,n_1}A_{f_1}/2$ and $-W_{f_1,n_2}A_{f_1}/2$. Thus via the negated transpose of the gradient, the $\vec{u}$ values at nodes n_1 and n_2 contribute to the divergence at the yellow dot with the same negated weights $W_{f_1,n_1}A_{f_1}/2$ and $W_{f_1,n_2}A_{f_1}/2$ which sum to half the area of face f_1.

5.2.3. Control Volumes of the Gradients

We have defined the volume-weighted gradient and the volume-weighted divergence which can be expressed as matrices $\tilde{\mathbf{G}}$ and $-\tilde{\mathbf{G}}^T$ respectively. Our discretization also requires defining the diagonal matrix $\tilde{\mathbf{V}}$ which consists of the control volumes of the gradients, so that the volume-weighted Laplacian can be defined as $-\tilde{\mathbf{G}}^T(\beta\tilde{\mathbf{V}}^{-1})\tilde{\mathbf{G}}$. Here β is a scalar value because it is spatially constant, and $\beta\tilde{\mathbf{V}}^{-1}$ typically represents one over the mass. On the MAC grid, the control volumes of the gradients are illustrated by the blue shaded regions in Figure 21. Note that each subfigure in Figure 21 corresponds to a subfigure in Figure 16. For the gradients in the x-direction, the control volumes defined in Figures 21(a) and 21(c) cover the entire MAC grid region as shown in Figure 22(a). Moreover, as seen in Figure 17, since the ϕ values at the yellow dots depend on the nodal ϕ values, none of those nodal gradients involve the MAC grid degrees of freedom, and therefore the control volumes of those nodal gradients only have volume fractions from the tessellated Lagrangian mesh, i.e. their control volumes are computed in the same fashion as in Section 4.2. Note that in the x-direction the tessellated Lagrangian mesh and the MAC grid are coupled by the gradients in Figure 21(c) via the definition of the yellow dots.

For the gradients in the y-direction, the control volumes defined in Figures 21(b), 21(d) and 21(e) cover

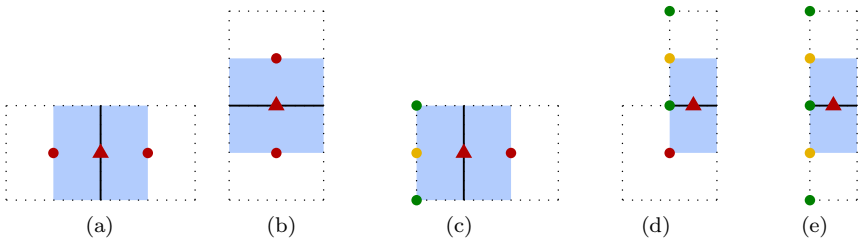
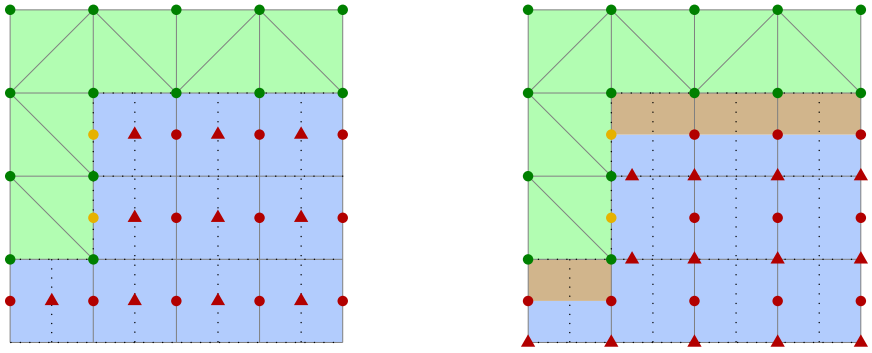


Figure 21: Control volume of the gradient on the MAC grid (the blue shaded region in each subfigure).



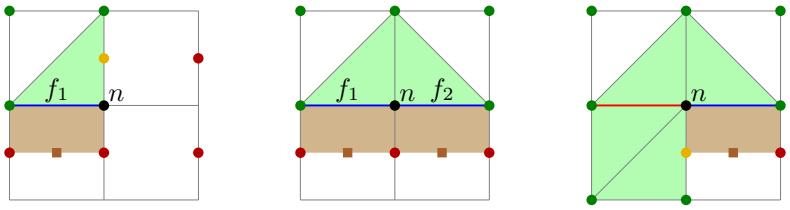
(a) x-direction (b) y-direction

Figure 22: Control volumes of the gradients on the entire hybrid structure. The green shaded region is covered by the tessellated Lagrangian mesh, and the blue shaded region consists of control volumes of the MAC grid gradients. For the gradients in the x-direction (Subfigure (a)) the blue region covers the entire MAC grid, but for the gradients in the y-direction (Subfigure (b)) the brown shaded region is not covered by the MAC grid control volumes.

the entire MAC grid region except the brown shaded regions shown in Figure 22(b). These brown regions are lumped to the nodal control volumes of the tessellated Lagrangian mesh as shown in Figure 23. In the case of Figure 23(a) which corresponds to the discretization shown in Figure 18(b), the y-directional gradient at the black node has contributions from both the tessellated Lagrangian mesh and the red dots, and therefore its control volume also has volume fractions from both the tessellated Lagrangian mesh and the MAC grid. The red dots contribute to the nodal gradients via the definition of the brown square which then behaves in the same fashion as the MAC grid cell centers behave in Figure 11(a). Thus, the volume fraction from the MAC grid is computed in a similar fashion as in Figure 11(a), i.e. using $\mathbf{W}^T$ to assign the volume of half the dual cell around face f_1 (the brown shaded region) to the black node n. This yields $W_{f_1,n}V_c/2$ where V_c is the volume of a regular MAC grid cell. In the case of Figure 23(b) which corresponds to the discretization shown in Figure 18(c), the control volume is defined similarly as in the previous case except that now there are two brown shaded regions that assign volume to the black node yielding $\sum_{f=f_1}^{f_2} W_{f,n}V_c/2$. In the case of Figure 23(c) which corresponds to the discretization shown in Figure 18(d), although the brown square is averaged from a yellow and a red dot instead of from two red dots, it still contributes to the nodal gradients in the same fashion. Therefore, the brown shaded region also assigns volume to the black node in a similar fashion as in Figure 23(a).

5.2.4. Numerical Results

We have reimplemented the heat equation tests from Tables 16 and 17 to test the order of accuracy on the hybrid structure with MAC grid cells replaced by MAC grid faces. Each orthogonal set of MAC grid faces is tested independently using the same configuration with the trapezoidal rule time integration scheme.



(a) (b) (c)

Figure 23: The control volume associated with the gradient in the y-direction at the black node in each subfigure equals one third the volume of the green shaded region plus half the volume of the brown shaded region.

Table 20: The order of accuracy for the 2D heat equation test on the x-component faces.

Grid Cells	L^1 Error	Order	L^∞ Error	Order
256	1.30×10^{-5}	–	2.72×10^{-3}	–
512	2.41×10^{-6}	2.43	8.50×10^{-4}	1.68
1024	4.99×10^{-7}	2.27	2.11×10^{-4}	2.01
2048	1.13×10^{-7}	2.15	5.60×10^{-5}	1.91

Tables 20 and 21 show second order accuracy on the x-component faces. The results on the y-component and z-component faces are similar.

6. Incompressible Flow

We consider the incompressible Navier-Stokes equations as follows

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \vec{u} + \vec{g}, \tag{27}$$

$$\nabla \cdot \vec{u} = 0, \tag{28}$$

where ρ is the density, $\vec{u}$ is the velocity, p is the pressure, ν is the kinematic viscosity, and $\vec{g}$ is gravity. The entire computational domain is defined using a MAC grid, and the part of the domain occupied by fluid is discretized using a hybrid particle grid structure as described in Section 2. Velocities are stored on particles in a colocated fashion and are also stored on MAC grid faces as is typical for MAC grid discretizations.

Equations (27) and (28) are solved with a splitting method [12]. First, we solve for advection,

$$\vec{u}^* = \vec{u}^n - \Delta t(\vec{u}^n \cdot \nabla)\vec{u}^n,$$

using the method from Section 3 – specifically forward Euler on particles and second order accurate SL-MacCormack on the MAC grid. Then we apply gravity to both the particles and the MAC grid

$$\vec{u}^{**} = \vec{u}^* + \Delta t \vec{g},$$

Table 21: The order of accuracy for the 3D heat equation test on the x-component faces.

Grid Cells	L^1 Error	Order	L^∞ Error	Order
128	5.22×10^{-7}	–	4.50×10^{-5}	–
256	9.92×10^{-8}	2.40	1.21×10^{-5}	1.89
512	1.89×10^{-8}	2.39	2.99×10^{-6}	2.02

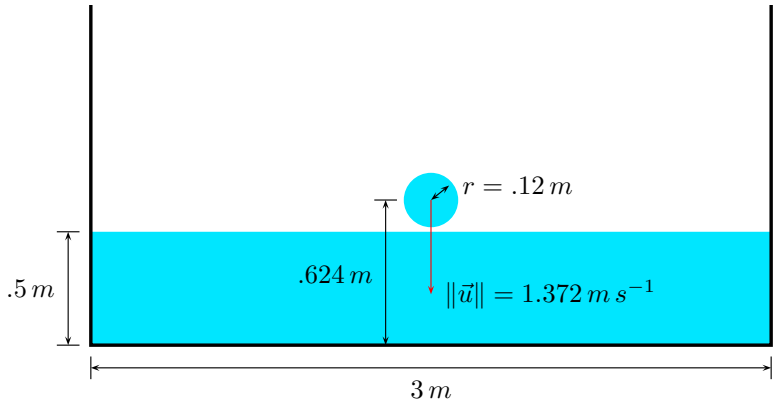


Figure 24: Configuration of 2D droplet impact test.

and subsequently apply viscosity

$$\vec{u}^{***} = \vec{u}^{**} + \Delta t(\nu \nabla^2 \vec{u}^{***}).$$

This equation is solved independently for each Cartesian component of $\vec{u}$ using a first order accurate time discretization (equation (25)) and a second order accurate spatial discretization as explained in Section 5.2. Note that we use backward Euler time integration as opposed to the trapezoidal rule in order to avoid potential numerical oscillations caused by unresolved temporal modes. On the Cartesian domain boundaries, slip boundary conditions are applied by specifying Dirichlet boundary conditions for the normal component of velocity and zero Neumann boundary conditions for the tangential components of velocity. If viscous interactions with domain boundaries are important, we can switch to no-slip boundary conditions. On the free surface, zero Neumann boundary conditions are applied to all Cartesian components of $\vec{u}$. Finally, we solve for pressure as follows

$$\nabla \cdot (\frac{1}{\rho} \nabla p) = \frac{1}{\Delta t} \nabla \cdot \vec{u}^{***}.$$

This equation is discretized as explained in Section 4 yielding the following analog of equation (24)

$$-\bar{\mathbf{G}}^T \bar{\mathbf{M}}^{-1} \bar{\mathbf{G}} \bar{\mathbf{p}} = -\bar{\mathbf{G}}^T \bar{\mathbf{u}}^{***}, \tag{29}$$

where $\bar{\mathbf{M}}$ is the mass matrix, $\bar{\mathbf{p}}$ is the vector of Δt-weighted pressures, and $\bar{\mathbf{u}}^{***}$ is the vector of $\vec{u}^{***}$ values. On Cartesian domain boundaries, zero Neumann boundary conditions are applied. On the free surface, zero Dirichlet boundary conditions are applied. Note that since only zero values are used for boundary conditions, the term related to boundary conditions (analog of the $\bar{\mathbf{b}}$ term in equation (24)) vanishes in equation (29). After solving for pressure, the velocities are updated as follows

$$\bar{\mathbf{u}}^{n+1} = \bar{\mathbf{u}}^{***} - \bar{\mathbf{M}}^{-1} \bar{\mathbf{G}} \bar{\mathbf{p}}.$$

This updates velocities on the nodes of the tessellated Lagrangian mesh as well as velocities in the interior region of the MAC grid. The velocities elsewhere are updated using interpolation or extrapolation as explained in Section 3.

6.1. Numerical Results

Consider a droplet impact problem shown in Figure 24. The computational domain is $[0\,m, 3\,m] \times [0\,m, 1.5\,m]$ which is partially filled with liquid where the free surface is located at $y = .5\,m$. A circular liquid droplet of radius $r = .12\,m$ is centered at $[1.5\,m, .624\,m]$ and has an initial velocity $[0\,m\,s^{-1}, -1.372\,m\,s^{-1}]$. The liquid has parameters $\rho = 1000\,kg\,m^{-2}$ and $\mu = .1\,kg\,s^{-1}$. No surface tension is used in this test.

Figure 26 shows the free surface at $t = .4\,s$, $t = .8\,s$ and $t = 1.2\,s$ for four grids of resolutions 300×150, 600×300, 1200×600 and 2400×1200 respectively. Note the thin features that our solver is able to resolve. Although we do not expect convergence under grid refinement since the flow is turbulent, we do observe that the splashing liquid sheets roughly converge to the same location under grid refinement. In order to more clearly illustrate the convergence behavior of our solver, we increase the viscosity of the liquid to $\mu = 10\,kg\,s^{-1}$. Figure 27 shows the free surface at $t = .2\,s$, $t = .5\,s$ and $t = .8\,s$.

We also simulated a dam breaking experiment corresponding to Figure 2(b) from [28]. The configuration is shown in Figure 25. The computational domain is $[0\,m, 1.28\,m] \times [0\,m, .16\,m]$. A rectangular reservoir is placed at the left with a range of $[0\,m, .38\,m] \times [0\,m, .15m]$, and a wet bed is placed at the right with a range of $[.38\,m, 1.28\,m] \times [0\,m, .018\,m]$. The liquid has parameters $\rho = 1000\,kg\,m^{-2}$ and $\mu = .001\,kg\,s^{-1}$. No surface tension is used in this test. Figure 28 shows the free surface at $t = .1\,s$, $t = .2\,s$, $t = .3\,s$, $t = .4\,s$, $t = .5\,s$ and $t = .6\,s$ for three grids of resolutions 80×640, 160×1280 and 320×2560 respectively. Note the small scale details that our solver is able to resolve and accurately track over time. Our results successfully capture the initial mushroom shape on the wave front seen in Figure 28(a) which was first reported in [50] and also observed in [28]. Our results also capture multiple wave breaking events in the later stages which was reported in [28].

The above two tests have also been extended to three spatial dimensions. For the droplet impact test, the computational domain is $[0\,m, 3\,m] \times [0\,m, 2\,m] \times [0\,m, 3\,m]$ and the free surface is located at $y = .5\,m$. A spherical liquid droplet of radius $r = .12\,m$ is centered at $[1.5\,m, .624\,m, 1.5\,m]$ with an initial velocity $[0\,m\,s^{-1}, -5.372\,m\,s^{-1}, 0\,m\,s^{-1}]$. The liquid has parameters $\rho = 1000\,kg\,m^{-3}$ and $\mu = .1\,kg\,m^{-1}\,s^{-1}$. Figure 29 shows the results at $t = 0\,s$, $t = .2\,s$, $t = .7\,s$ and $t = 1.2\,s$ using a grid of resolution $150 \times 100 \times 150$. For the dam breaking test, the computational domain is $[0\,m, 1.28\,m] \times [0\,m, .16\,m] \times [0\,m, .16\,m]$. The reservoir and wet bed are configured in the same way as shown in Figure 25 except that they are extended along the z-axis from $z = 0\,m$ to $z = .16\,m$. Figure 30 shows the results at $t = 0\,s$, $t = .1\,s$, $t = .2\,s$, $t = .3\,s$, $t = .4\,s$, $t = .5\,s$ and $t = .6\,s$ using a grid of resolution $64 \times 512 \times 64$.

7. Surface Tension

We consider incompressible Navier-Stokes equations with surface tension as follows

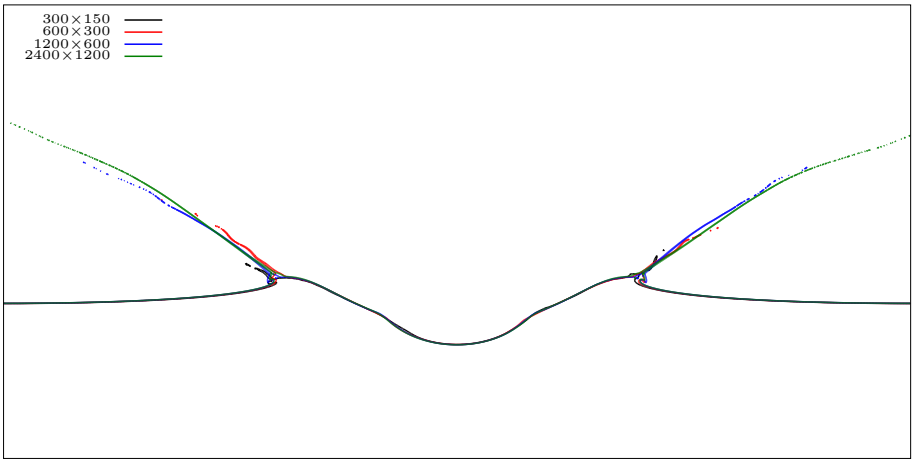
$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} = -\frac{1}{\rho}\nabla p + \frac{1}{\rho}\vec{f} + \vec{g}, \tag{30}$$

$$\nabla \cdot \vec{u} = 0, \tag{31}$$

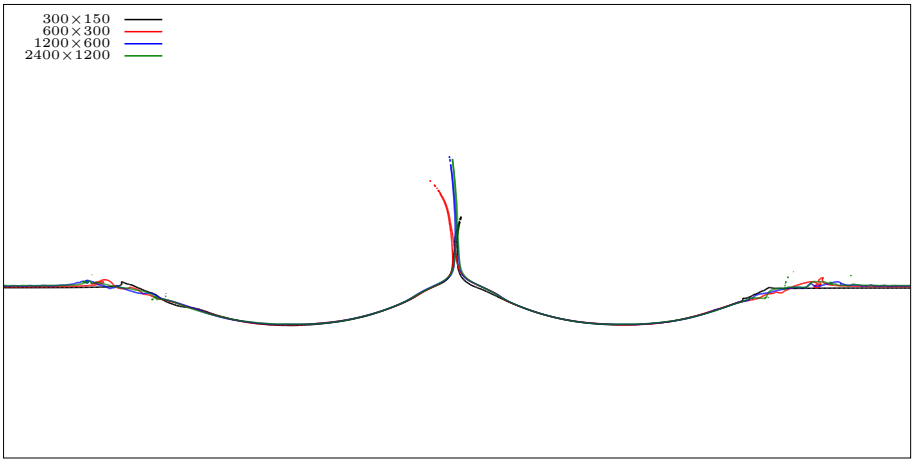
where $\vec{f}$ is the surface tension force density. For simplicity, the viscous term is ignored during the derivation in this section, but adding this force back is straightforward.

We take a step-by-step constructive approach to the discretization for illustrative purposes. We start with a fully Lagrangian representation of the fluid in Section 7.1, in light of the fact that an explicit representation of the fluid geometry gives precise feedback to an implicit solver for surface tension. In Section 7.2, a semi-implicit scheme is devised motivated by the method in [43], but it does not approach a steady state solution when using large time steps. To solve this issue, a fully implicit scheme is developed in Section 7.3. This scheme successfully achieves a steady state solution with large time step sizes, but it fails to conserve volume.

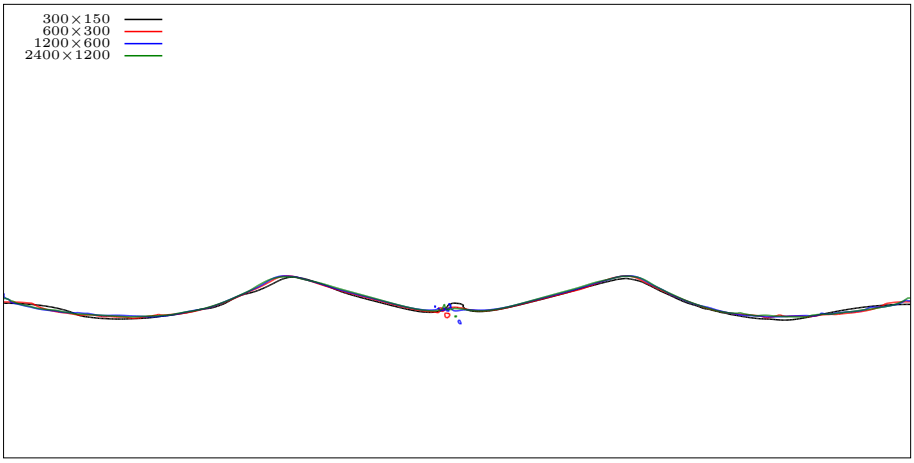
Figure 25: Configuration of 2D dam breaking test.



(a) $t = .4\,s$

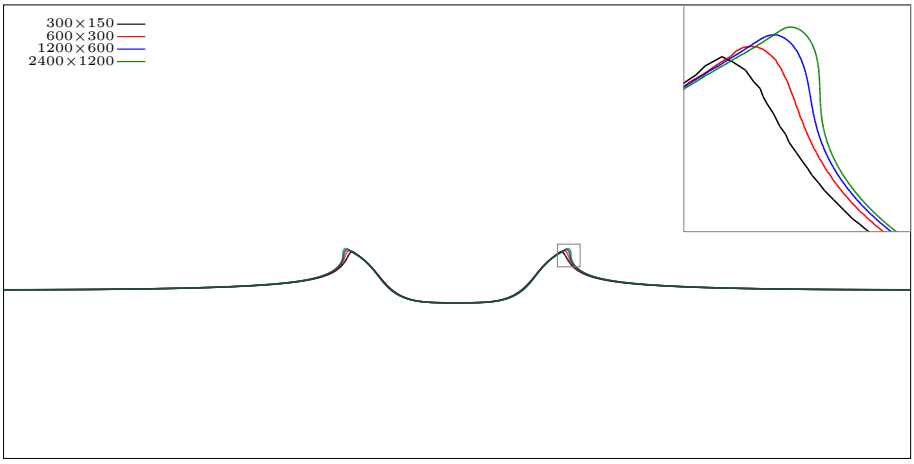


(b) $t = .8\,s$

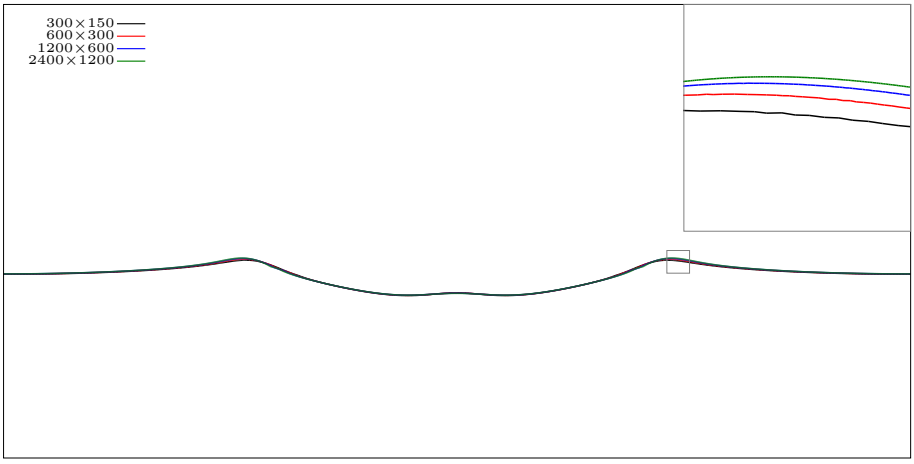


(c) $t = 1.2\,s$

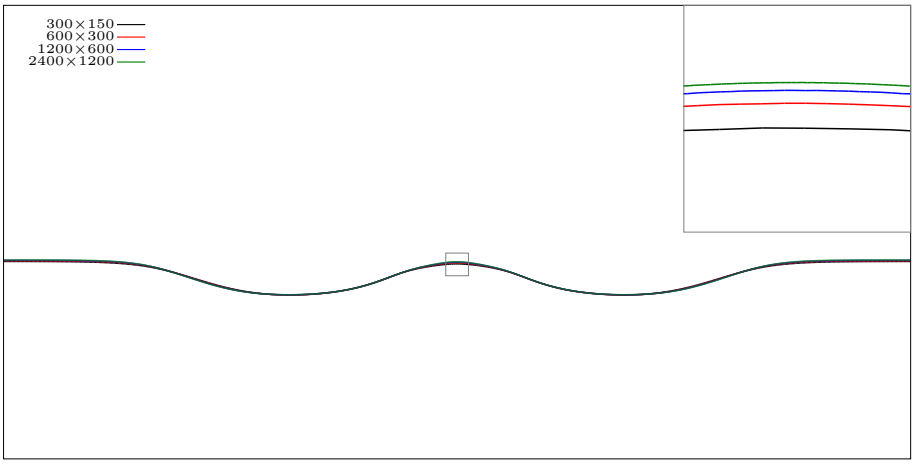
Figure 26: The free surface for the 2D droplet impact test with $\mu = .1\,kg\,s^{-1}$. Four different grids of resolutions are shown.



(a) $t = .2\,s$



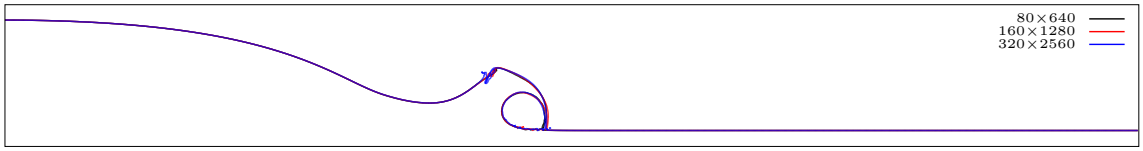
(b) $t = .5\,s$



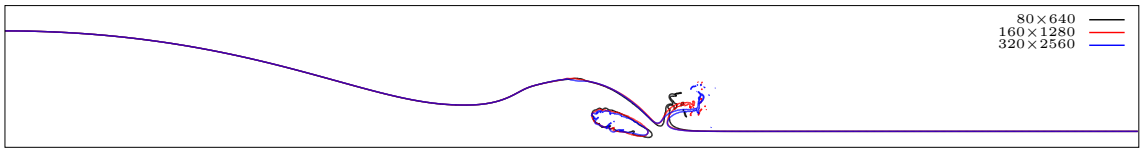
(c) $t = .8\,s$

Figure 27: The free surface for the 2D droplet impact test with $\mu = 10\,kg\,s^{-1}$. Four different grids of resolutions are shown.

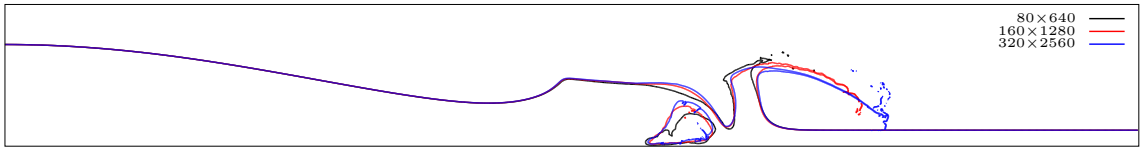
(a) $t = .1\,s$



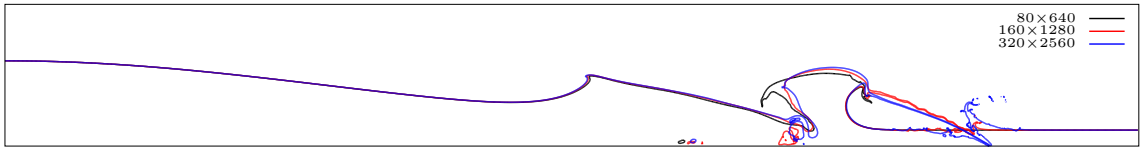
(b) $t = .2\,s$



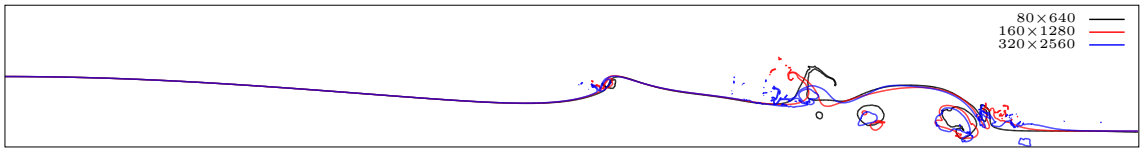
(c) $t = .3\,s$



(d) $t = .4\,s$

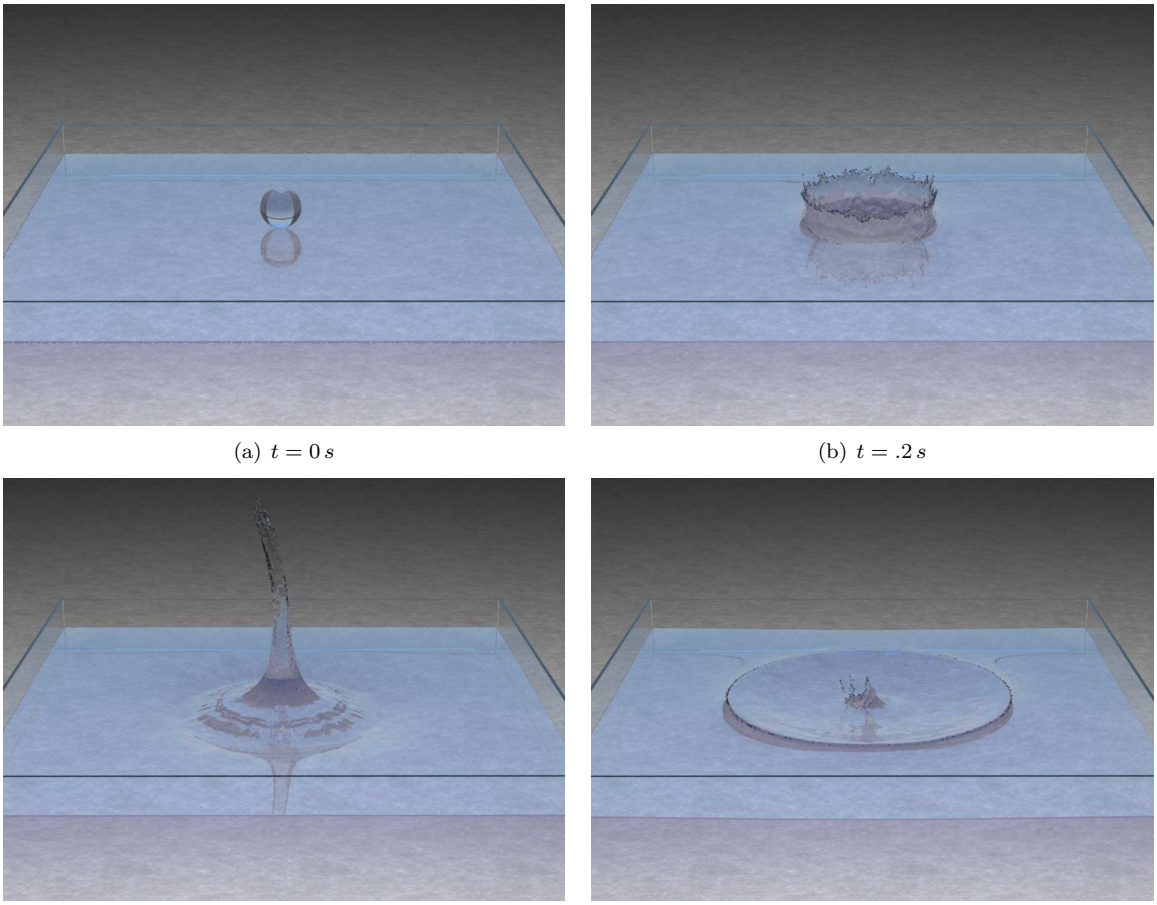


(e) $t = .5\,s$



(f) $t = .6\,s$

Figure 28: The free surface for the 2D dam breaking test with $\rho = 1000\,kg\,m^{-2}$ and $\mu = .001\,kg\,s^{-1}$. Three different grids of resolutions are shown.



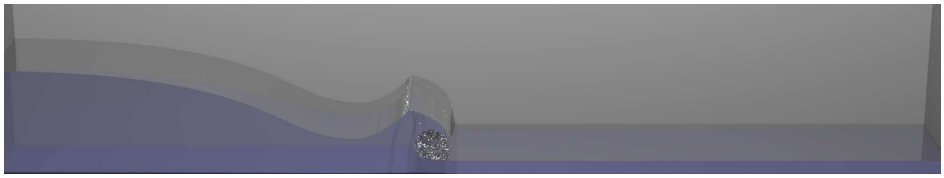
(a) $t = 0\,s$ (b) $t = .2\,s$

(c) $t = .7\,s$ (d) $t = 1.2\,s$

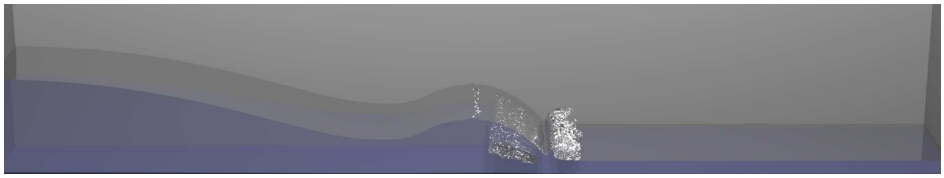
Figure 29: 3D droplet impact test with $\rho = 1000\,kg\,m^{-3}$ and $\mu = .1\,kg\,m^{-1}\,s^{-1}$.

(a) $t = 0\,s$

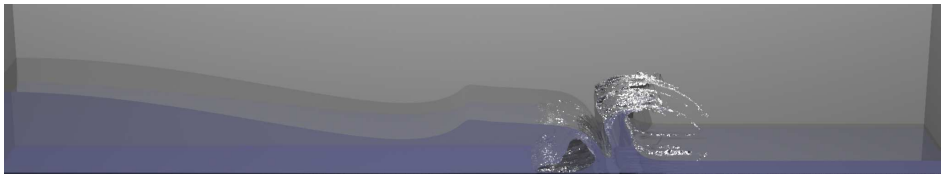
(b) $t = .1\,s$



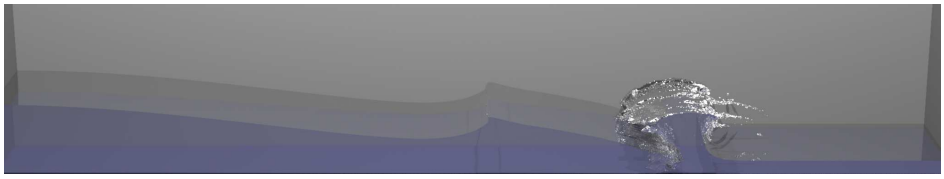
(c) $t = .2\,s$



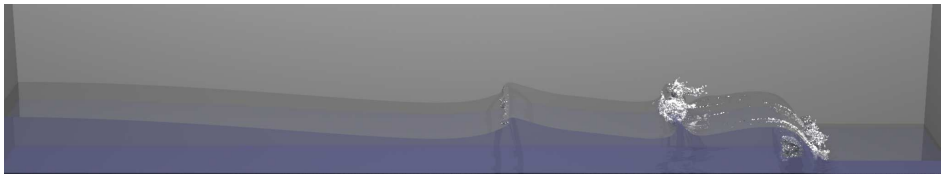
(d) $t = .3\,s$



(e) $t = .4\,s$



(f) $t = .5\,s$



(g) $t = .6\,s$

Figure 30: 3D dam breaking test with $\rho = 1000\,kg\,m^{-3}$ and $\mu = .001\,kg\,m^{-1}\,s^{-1}$.

We then improve upon this scheme in Section 7.4 explicitly conserving volume by replacing the divergence-free condition with a constant-volume condition. The final scheme achieves a steady state solution and conserves volume well. For the sake of efficiency, this scheme is then extended to the hybrid particle grid structure in Section 7.5. In addition, we also show how one would apply these ideas to standard front tracking and level set methods in Section 7.6.

7.1. Lagrangian Surface Tension Force

We discretize the surface tension force on a Lagrangian surface mesh since it is convenient to treat the force implicitly with an explicit representation of the interface. In two spatial dimensions, the surface tension force is discretized on a mesh of line segments using the method in [43], which is briefly described as in Section 7.1.1. In three spatial dimensions, the surface tension force is discretized on a mesh of triangles using the method described in Section 7.1.2.

7.1.1. Discretization in 2D

The free surface is approximated by a mesh S of line segments, and the surface tension force is computed on S in a per-segment fashion as follows. Let s be a line segment on S connecting two nodes i and j. The force applied at nodes i and j due to segment s is then

$$\vec{f}_{s,i} = -\vec{f}_{s,j} = \sigma \vec{t}_s, \tag{32}$$

where σ is the surface tension coefficient, and $\vec{t}_s = \frac{\vec{x}_j - \vec{x}_i}{\|\vec{x}_j - \vec{x}_i\|}$ is the tangential direction of the segment s. Equation (32) is summed over all segments to obtain the explicit part of the surface tension force.

In order to treat the surface tension force implicitly, the force derivatives are computed as follows

$$\frac{\partial \vec{f}_{s,i}}{\partial \vec{x}_i} = \frac{\partial \vec{f}_{s,j}}{\partial \vec{x}_j} = -\frac{\partial \vec{f}_{s,j}}{\partial \vec{x}_i} = -\frac{\partial \vec{f}_{s,i}}{\partial \vec{x}_j} = -\frac{\sigma}{l_s} \vec{n}_s \vec{n}_s^T, \tag{33}$$

where l_s is the length of the segment s, and n_s is the direction normal to the segment. Composing these derivatives into a matrix yields

$$\mathbf{D}_s = -\frac{\sigma}{l_s} \begin{pmatrix} \vec{n}_s \vec{n}_s^T & -\vec{n}_s \vec{n}_s^T \\ -\vec{n}_s \vec{n}_s^T & \vec{n}_s \vec{n}_s^T \end{pmatrix}. \tag{34}$$

The matrix $\mathbf{D}_s$ is symmetric negative semi-definite. It can be factorized as $\mathbf{D}_s = -\mathbf{C}_s^T \mathbf{C}_s$ where

$$\mathbf{C}_s = \sqrt{\frac{\sigma}{l_s}} \begin{pmatrix} \vec{n}_s^T & -\vec{n}_s^T \end{pmatrix}. \tag{35}$$

This factorization will be useful when coupling surface tension with pressure, where the technique from [42] will be used to form an SPD system. The $\mathbf{D}_s$ and $\mathbf{C}_s$ matrices are summed over all segments on S to form the full matrices $\mathbf{D}$ and $\mathbf{C}$. Since all $\mathbf{D}_s$ matrices are symmetric negative semi-definite, their sum $\mathbf{D}$ is as well. Note that we can factorize $\mathbf{D} = -\mathbf{C}^T\mathbf{C}$ due to the independent nature of the individual $\mathbf{C}_s$ matrices.

$\mathbf{D}_s$ projects out the tangential velocity components, and the resulting force acts only in the normal direction. We refer to this model as the normal-directional implicit model. It may be desirable to have $\mathbf{D}_s$ act in the tangential direction as well. If one treats l_s as a constant in equation (32), the derivatives in equation (34) become

$$\hat{\mathbf{D}}_s = -\frac{\sigma}{l_s} \begin{pmatrix} \mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & \mathbf{I} \end{pmatrix},$$

where $\hat{\mathbf{D}}_s$ is also symmetric negative semi-definite, and equation (35) becomes

$$\hat{\mathbf{C}}_s = \sqrt{\frac{\sigma}{l_s}} \begin{pmatrix} \mathbf{I} & -\mathbf{I} \end{pmatrix}.$$

We refer to this as the all-directional implicit model.

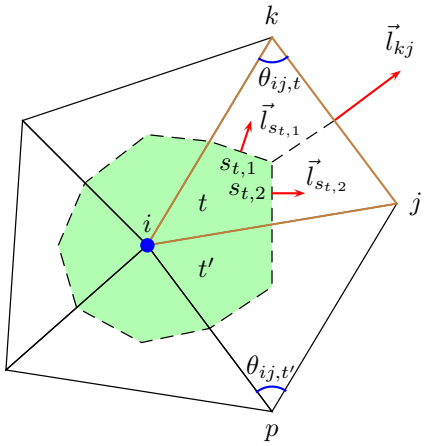


Figure 31: Discretization of the surface tension force in 3D.

7.1.2. Discretization in 3D

The free surface is approximated by a triangular mesh T. Let i be a node on T (the blue dot in Figure 31). We define the control area of node i (denoted by A_i) to be one third of the area of its incident triangles (the green shaded region), where each incident triangle has been divided into three equal-area pieces by cutting along the segments connecting the centroid and the midpoints of the three edges. The net force on node i is computed by integrating surface tension along the boundary of A_i as

$$\vec{f}_i = \int_{\partial A_i} \sigma \vec{m}\, dL,$$

where $\vec{m}$ is the outward pointing unit binormal that is perpendicular to ∂A_i and tangent to the surface. Since ∂A_i is a set of boundary segments, this integral is transformed into a summation over the boundary segments as

$$\vec{f}_i = \sum_s \sigma \vec{l}_s, \tag{36}$$

where $\vec{l}_s$ is the length-weighted binormal of the boundary segment s. Noting that every incident triangle contains two boundary segments, equation (36) can be further transformed into a summation over the incident triangles as

$$\vec{f}_i = \sum_t \sigma(\vec{l}_{s_{t,1}} + \vec{l}_{s_{t,2}}) = \sum_t \vec{f}_{t,i}. \tag{37}$$

Consider the brown triangle in Figure 31. Segments $s_{t,1}$ and $s_{t,2}$ form a planar polygon with half of the edges ji and ik (the green shaded region inside the brown triangle), and thus $\vec{l}_{s_1} + \vec{l}_{s_2} = -\frac{1}{2}(\vec{l}_{ji} + \vec{l}_{ik})$ where $\vec{l}_{ji}$ and $\vec{l}_{ik}$ are the length-weighted binormals of edges ji and ik. In addition, since edges ji, ik and kj form a triangle, $-\frac{1}{2}(\vec{l}_{ji} + \vec{l}_{ik}) = \frac{1}{2}\vec{l}_{kj}$. Therefore, equation (37) can be rewritten as

$$\vec{f}_{t,i} = \frac{\sigma}{2}\vec{l}_{kj}. \tag{38}$$

Similarly, the brown triangle also contributes to the forces on nodes j and k as

$$\vec{f}_{t,j} = \frac{\sigma}{2}\vec{l}_{ik}, \qquad \vec{f}_{t,k} = \frac{\sigma}{2}\vec{l}_{ji}. \tag{39}$$

In order to treat the force implicitly, we compute the force derivatives using this per-triangle form of the

force to obtain

$$\frac{\partial(\vec{f}_{t,i},\vec{f}_{t,j},\vec{f}_{t,k})}{\partial(\vec{x}_i,\vec{x}_j,\vec{x}_k)} = \begin{pmatrix} -\frac{\sigma\vec{x}_{kj}^T\vec{x}_{kj}}{4A_t}\vec{n}_t\vec{n}_t^T & -\frac{\sigma\vec{x}_{kj}^T\vec{x}_{ik}}{4A_t}\vec{n}_t\vec{n}_t^T + \frac{\sigma}{2}[\vec{n}_t]_\times & -\frac{\sigma\vec{x}_{kj}^T\vec{x}_{ji}}{4A_t}\vec{n}_t\vec{n}_t^T - \frac{\sigma}{2}[\vec{n}_t]_\times \\ -\frac{\sigma\vec{x}_{ik}^T\vec{x}_{kj}}{4A_t}\vec{n}_t\vec{n}_t^T - \frac{\sigma}{2}[\vec{n}_t]_\times & -\frac{\sigma\vec{x}_{ik}^T\vec{x}_{ik}}{4A_t}\vec{n}_t\vec{n}_t^T & -\frac{\sigma\vec{x}_{ik}^T\vec{x}_{ji}}{4A_t}\vec{n}_t\vec{n}_t^T + \frac{\sigma}{2}[\vec{n}_t]_\times \\ -\frac{\sigma\vec{x}_{ji}^T\vec{x}_{kj}}{4A_t}\vec{n}_t\vec{n}_t^T + \frac{\sigma}{2}[\vec{n}_t]_\times & -\frac{\sigma\vec{x}_{ji}^T\vec{x}_{ik}}{4A_t}\vec{n}_t\vec{n}_t^T - \frac{\sigma}{2}[\vec{n}_t]_\times & -\frac{\sigma\vec{x}_{ji}^T\vec{x}_{ji}}{4A_t}\vec{n}_t\vec{n}_t^T \end{pmatrix},$$

where $\vec{x}_{ab} = \vec{x}_a - \vec{x}_b$, A_t is the area of triangle t, $\vec{n}_t$ is the unit normal of triangle t, and $[\cdot]_\times$ is the cross product matrix of a vector. This matrix is indefinite due to the $[\vec{n}_t]_\times$ terms. To avoid this problem, we drop all $[\vec{n}_t]_\times$ terms to obtain a symmetric negative semi-definite approximation $\mathbf{D}_t$. This approximated matrix can be factorized to obtain

$$\mathbf{C}_t = \sqrt{\frac{\sigma}{4A_t}} \begin{pmatrix} \vec{x}_{kj}\vec{n}_t^T & \vec{x}_{ik}\vec{n}_t^T & \vec{x}_{ji}\vec{n}_t^T \end{pmatrix}.$$

The all-directional implicit model is derived as follows. Since $\vec{l}_{kj} = \vec{x}_{kj} \times \vec{n}_t$ and $\vec{n}_t = \frac{1}{2A_t}(\vec{x}_{ik} \times \vec{x}_{ji})$, equation (38) can be rewritten as

$$\begin{aligned} \vec{f}_{t,i} &= \frac{\sigma}{2}\vec{x}_{kj} \times \vec{n}_t = \frac{\sigma}{4A_t}\vec{x}_{kj} \times (\vec{x}_{ik} \times \vec{x}_{ji}) \\ &= \frac{\sigma}{4A_t}\left((\vec{x}_{kj} \cdot \vec{x}_{ji})\vec{x}_{ik} - (\vec{x}_{kj} \cdot \vec{x}_{ik})\vec{x}_{ji}\right) \\ &= \frac{\sigma}{4A_t}\left((\vec{x}_{kj} \cdot \vec{x}_{ji})\vec{x}_{ik} + (\vec{x}_{kj} \cdot \vec{x}_{ik})\vec{x}_{ij}\right). \end{aligned} \tag{40}$$

Since $(\vec{x}_{kj} \cdot \vec{x}_{ji})/2A_t = -\cot\theta_{ik,t}$ and $(\vec{x}_{kj} \cdot \vec{x}_{ik})/2A_t = -\cot\theta_{ij,t}$ where $\theta_{ik,t}$ and $\theta_{ij,t}$ are the interior angles of triangle t that are element-wise opposite edges ik and ij respectively, equation (40) can be written as

$$\vec{f}_{t,i} = -\frac{\sigma}{2}\left(\vec{x}_{ik}\cot\theta_{ik,t} + \vec{x}_{ij}\cot\theta_{ij,t}\right). \tag{41}$$

Noting that edge ij is adjacent to both triangles t and t' (see Figure 31), we also write the force due to triangle t' as

$$\vec{f}_{t',i} = -\frac{\sigma}{2}\left(\vec{x}_{ip}\cot\theta_{ip,t'} + \vec{x}_{ij}\cot\theta_{ij,t'}\right). \tag{42}$$

Collecting the $\vec{x}_{ij}$ terms from equations (41) and (42) yields

$$\vec{f}_{ij,i} = -\frac{\sigma}{2}\vec{x}_{ij}(\cot\theta_{ij,t} + \cot\theta_{ij,t'}), \tag{43}$$

where $\vec{f}_{ij,i}$ is the force on node i due to edge ij. Similarly, the force on node j due to edge ij is

$$\vec{f}_{ij,j} = \frac{\sigma}{2}\vec{x}_{ij}(\cot\theta_{ij,t} + \cot\theta_{ij,t'}) = -\vec{f}_{ij,i}. \tag{44}$$

Equations (43) and (44) provide a per-edge form of the force, and they can be expressed in the matrix form as

$$\begin{pmatrix} \vec{f}_{ij,i} \\ \vec{f}_{ij,j} \end{pmatrix} = -\frac{\sigma(\cot\theta_{ij,t} + \cot\theta_{ij,t'})}{2}\begin{pmatrix} \mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & \mathbf{I} \end{pmatrix}\begin{pmatrix} \vec{x}_i \\ \vec{x}_j \end{pmatrix}.$$

Treating the cotangent terms as constants, we obtain

$$\hat{\mathbf{D}}_t = -\frac{\sigma(\cot\theta_{ij,t} + \cot\theta_{ij,t'})}{2}\begin{pmatrix} \mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & \mathbf{I} \end{pmatrix},$$

as an approximation to the force derivatives for the all-directional implicit model. Note that $\hat{\mathbf{D}}_t$ may be symmetric positive semi-definite or symmetric negative semi-definite depending on the sign of $(\cot\theta_{ij,t} + \cot\theta_{ij,t'})$. Thus, summing $\hat{\mathbf{D}}_t$ over all edges may result in an indefinite matrix which is undesirable. As proved in [7], by using an edge flipping algorithm any triangular mesh can be transformed into a special type of mesh where the condition $(\cot\theta_{ij,t} + \cot\theta_{ij,t'}) \geq 0$ is always true. In practice, we simply clamp the value of $(\cot\theta_{ij,t} + \cot\theta_{ij,t'})$ to be non-negative. We found that this simple scheme works well on our surface mesh, since our surface mesh is relatively regular. The clamped $\hat{\mathbf{D}}_t$ is symmetric negative semi-definite, and it can be factorized to obtain

$$\hat{\mathbf{C}}_t = \sqrt{\frac{\sigma(\cot\theta_{ij,t} + \cot\theta_{ij,t'})}{2}} \begin{pmatrix} \mathbf{I} & -\mathbf{I} \end{pmatrix}.$$

7.2. Semi-implicit Scheme with a Lagrangian Mesh

In this section, the particles are connected into a volumetric mesh using the meshing algorithm in Section 2 for the entire fluid region, i.e. there is no interior MAC grid region. The surface tension force is computed on the surface of this mesh using the method from Section 7.1. In order to treat the surface tension force implicitly, a linearized prediction is used

$$\hat{\mathbf{f}}^{n+1} = \hat{\mathbf{f}}(\hat{\mathbf{x}}^{n+1}) = \hat{\mathbf{f}}(\hat{\mathbf{x}}^n + \Delta t\hat{\mathbf{u}}^{n+1}) \approx \hat{\mathbf{f}}^n + \Delta t\mathbf{D}\hat{\mathbf{u}}^{n+1}, \tag{45}$$

where $\hat{\mathbf{x}}^{n+1}$ and $\hat{\mathbf{u}}^{n+1}$ are the particle positions and velocities at time t^{n+1}, and $\mathbf{D}$ is the implicit matrix constructed as in Section 7.1. A splitting method [12] is used to solve equations (30) and (31). First, an intermediate velocity $\hat{\mathbf{u}}^*$ is obtained by advecting particles forward using the forward Euler method. Then, the explicit part of the surface tension force and gravity are applied to obtain another intermediate velocity

$$\hat{\mathbf{u}}^{**} = \hat{\mathbf{u}}^* + \Delta t\hat{\mathbf{M}}^{-1}\hat{\mathbf{f}}^n + \Delta t\hat{\mathbf{g}},$$

where $\hat{\mathbf{M}}$ is the mass matrix and the mass of each particle is defined using the density-weighted control volume.

Finally, the implicit part of the surface tension force is coupled with the pressure to obtain

$$\hat{\mathbf{u}}^{n+1} = \hat{\mathbf{u}}^{**} - \hat{\mathbf{M}}^{-1}\hat{\mathbf{G}}\hat{\mathbf{p}} + \Delta t^2\hat{\mathbf{M}}^{-1}\mathbf{D}\hat{\mathbf{u}}^{n+1}, \tag{46}$$

$$\hat{\mathbf{G}}^T\hat{\mathbf{u}}^{n+1} = 0, \tag{47}$$

where $\hat{\mathbf{G}}$ is the volume-weighted gradient matrix, and $\hat{\mathbf{p}}$ is the vector of Δt-weighted pressures. Equations (46) and (47) are transformed into an SPD system using the technique in [42]. First, $\mathbf{D}$ is factorized into $-\mathbf{C}^T\mathbf{C}$ as described in Section 7.1 allowing equation (46) to be rewritten as

$$\hat{\mathbf{u}}^{n+1} = \hat{\mathbf{u}}^{**} - \hat{\mathbf{M}}^{-1}\hat{\mathbf{G}}\hat{\mathbf{p}} + \Delta t^2\hat{\mathbf{M}}^{-1}\mathbf{C}^T\mathbf{C}\hat{\mathbf{u}}^{n+1}.$$

Rewriting this equation using block matrices yields

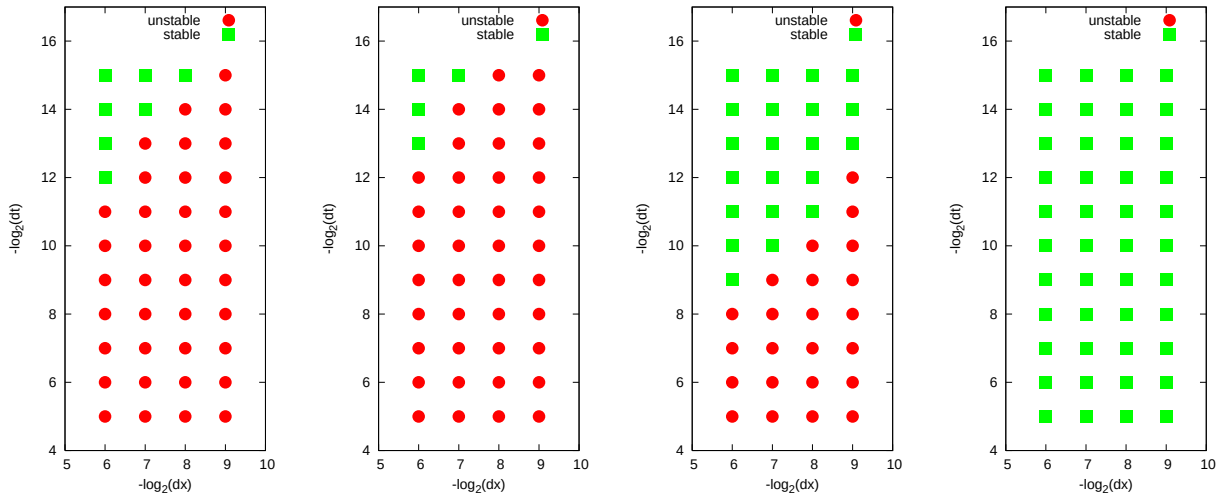
$$\hat{\mathbf{u}}^{n+1} = \hat{\mathbf{u}}^{**} - \hat{\mathbf{M}}^{-1}\begin{pmatrix} \hat{\mathbf{G}}^T \\ \Delta t\mathbf{C} \end{pmatrix}^T \begin{pmatrix} \hat{\mathbf{p}} \\ \Delta t\mathbf{C}\hat{\mathbf{u}}^{n+1} \end{pmatrix},$$

which can be written concisely as

$$\hat{\mathbf{u}}^{n+1} = \hat{\mathbf{u}}^{**} - \hat{\mathbf{M}}^{-1}\mathbf{K}^T\hat{\mathbf{z}}, \tag{48}$$

where

$$\mathbf{K} = \begin{pmatrix} \hat{\mathbf{G}}^T \\ \Delta t\mathbf{C} \end{pmatrix}, \quad \hat{\mathbf{z}} = \begin{pmatrix} \hat{\mathbf{p}} \\ \Delta t\mathbf{C}\hat{\mathbf{u}}^{n+1} \end{pmatrix}.$$



(a) level set (b) Lagrangian, explicit (c) Lagrangian, normal-directional (d) Lagrangian, all-directional

Figure 32: Stability tests on an oscillating ellipse. The x axis indicates the grid cell size (Subfigure (a)) or the average mesh element size (Subfigures (b), (c) and (d)), with more refined simulations towards the right. The y axis indicates the size of Δt, with smaller Δt towards the top. A green square indicates a stable simulation, and a red circle indicates an unstable simulation.

Left multiplying equation (48) by $\mathbf{K}$ and rearranging terms yields

$$\mathbf{K}\hat{\mathbf{u}}^{n+1} + \mathbf{K}\hat{\mathbf{M}}^{-1}\mathbf{K}^T\hat{\mathbf{z}} = \mathbf{K}\hat{\mathbf{u}}^{**}. \tag{49}$$

Noting that

$$\mathbf{K}\hat{\mathbf{u}}^{n+1} = \begin{pmatrix} \hat{\mathbf{G}}^T\hat{\mathbf{u}}^{n+1} \\ \Delta t\mathbf{C}\hat{\mathbf{u}}^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \Delta t\mathbf{C}\hat{\mathbf{u}}^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{p}} \\ \Delta t\mathbf{C}\hat{\mathbf{u}}^{n+1} \end{pmatrix} = \mathbf{P}\hat{\mathbf{z}},$$

where

$$\mathbf{P} = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix},$$

equation (49) can then be rewritten as

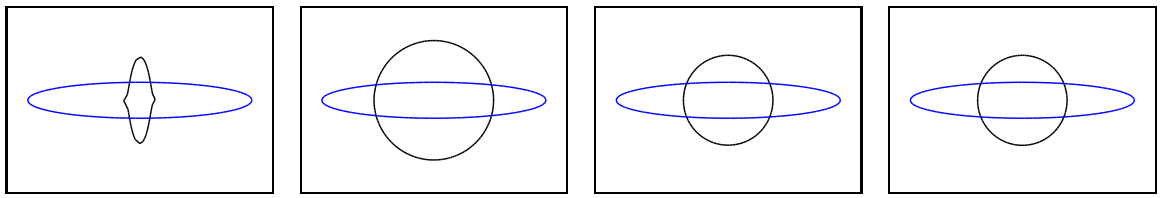
$$(\mathbf{P} + \mathbf{K}\hat{\mathbf{M}}^{-1}\mathbf{K}^T)\hat{\mathbf{z}} = \mathbf{K}\hat{\mathbf{u}}^{**}. \tag{50}$$

Equation (50) is an SPD system, and it is solved using the conjugate gradient method to obtain $\hat{\mathbf{z}}$, which is subsequently substituted into equation (48) to obtain the new velocity $\hat{\mathbf{u}}^{n+1}$. Note that if the viscous term is present, it can be coupled into equation (50) as shown in [42].

7.2.1. Stability Tests

An oscillating ellipse problem is used to test stability. The computational domain is $[-.005\,m, .005\,m] \times [-.005\,m, .005\,m]$. An elliptic liquid droplet is centered at the origin with an interface described by $(\frac{x}{r})^2 + (\frac{y}{r/\alpha})^2 = 1$ where $r = .0045\,m$ is the radius along the longer axis, and r/α is the radius along the shorter axis where $\alpha \geq 1$ controls the ratio between the longer and shorter axes. In all our tests, $\alpha = 2.5$ is used. The liquid has parameters $\rho = 1000\,kg\,m^{-2}$, $\mu = 1.138 \times 10^{-3}\,kg\,s^{-1}$ and $\sigma = .0728\,kg\,m\,s^{-2}$. No gravity is used. The droplet has zero initial velocity and is only driven by the surface tension force.

The semi-implicit scheme with the normal-directional model and that with the all-directional model are both tested. For comparison, two explicit schemes are also tested. The first explicit scheme uses the fully



(a) all-directional semi-implicit scheme (b) fully implicit scheme (c) fully implicit volume conserving scheme (d) fully implicit volume conserving scheme with the hybrid particle grid structure

Figure 33: The free surface of the oscillating ellipse test. The blue line shows the initial free surface, and the black line shows the free surface after a single rather large time step $\Delta t = .25\,s$. Subfigure (a) uses the all-directional semi-implicit scheme with a Lagrangian mesh, where the droplet does not achieve a steady state solution. Subfigure (b) uses the fully implicit scheme with a Lagrangian mesh, where the droplet achieves a steady state solution but gains 69.64% of its initial volume. Subfigure (c) uses the fully implicit volume conserving scheme with a Lagrangian mesh, where the droplet achieves a steady state solution and the relative volume error is only .07%. Subfigure (d) uses the fully implicit volume conserving scheme with the hybrid particle grid structure, where the droplet achieves a steady state solution and the relative volume error is only .088%.

Lagrangian representation described in this section, and it treats the surface tension force explicitly by simply setting the implicit matrix $\mathbf{D} = \mathbf{0}$. The second explicit scheme uses a level set based approach from [30] where the surface tension force is applied explicitly as a pressure jump.

Figure 32 shows the stability results with various mesh resolutions and time step sizes. A simulation is noted unstable if the free surface reaches the domain boundary indicating a severely distorted geometry. The two explicit schemes have similar stability properties. The semi-implicit scheme with the normal-directional model significantly improves stability but still suffers from instability with finer mesh resolutions and larger time step sizes. The semi-implicit scheme with the all-directional model remains stable on all mesh resolutions and time step sizes. However, this scheme does not approach a steady state solution when using large time steps. See Figure 33(a) where a simulation with a rather large time step of $\Delta t = .25\,s$ is run. This is because a linearized model is used to predict the surface tension force, which diverges from the actual force as Δt grows.

7.3. Fully Implicit Scheme with a Lagrangian Mesh

We start with the Lagrangian form of equation (30)

$$\frac{D\vec{u}}{Dt} = -\frac{1}{\rho}\nabla p + \frac{1}{\rho}\vec{f} + \vec{g}. \tag{51}$$

The temporal discretization of equations (51) and (31) using the backward Euler method is

$$\begin{aligned} \vec{u}^{n+1} - \vec{u}^{n} &= -\frac{\Delta t}{\rho}\nabla p^{n+1} + \frac{\Delta t}{\rho}\vec{f}^{n+1} + \Delta t\vec{g}, \\ \nabla \cdot \vec{u}^{n+1} &= 0. \end{aligned}$$

The spatial discretization is the same as in Section 7.2 but with all forces and matrices evaluated at the time t^{n+1}, which results in

$$\hat{\mathbf{M}}^{n+1}(\hat{\mathbf{u}}^{n+1} - \hat{\mathbf{u}}^{n}) + \hat{\mathbf{G}}^{n+1}\hat{\mathbf{p}}^{n+1} - \Delta t\hat{\mathbf{f}}^{n+1} - \Delta t\hat{\mathbf{M}}^{n+1}\hat{\mathbf{g}} = 0, \tag{52}$$

$$(\hat{\mathbf{G}}^{n+1})^{T}\hat{\mathbf{u}}^{n+1} = 0, \tag{53}$$

which can be compactly written as $\mathbf{F}(\mathbf{w}) = 0$ where

$$\mathbf{w} = \begin{pmatrix} \hat{\mathbf{u}}^{n+1} \\ \hat{\mathbf{p}}^{n+1} \end{pmatrix}.$$

An iterative approach is used to solve this nonlinear system. We begin with an initial guess $\hat{\mathbf{u}}_0^{n+1} = 0$, $\hat{\mathbf{p}}_0^{n+1} = 0$ and $\hat{\mathbf{x}}_0^{n+1} = \hat{\mathbf{x}}^n$ where the subscript indicates the iteration index. We drop the superscript $n+1$ in the remainder of this section where no confusion will result. The Newton-Raphson method is used

$$\mathbf{J}_k \Delta \mathbf{w} = -\mathbf{F}(\mathbf{w}_k), \tag{54}$$

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \Delta \mathbf{w}, \tag{55}$$

where $\mathbf{J}$ is the Jacobian of $\mathbf{F}$. The exact Jacobian is difficult to compute since it requires taking derivatives of $\hat{\mathbf{M}}$ and $\hat{\mathbf{G}}$. To avoid this difficulty, an approximated Jacobian is computed by treating $\hat{\mathbf{M}}$ and $\hat{\mathbf{G}}$ as constants for each iteration. Note that if the viscous term is present, its matrix will also be treated as a constant in this approximated Jacobian. To compute the Jacobian of $-\Delta t \hat{\mathbf{f}}^{n+1}$, we use the approximation $\hat{\mathbf{f}}^{n+1} \approx \hat{\mathbf{f}}^n + \Delta t \mathbf{D} \hat{\mathbf{u}}^{n+1}$. The derivative with respect to $\hat{\mathbf{u}}^{n+1}$ is then computed as $-\Delta t^2 \mathbf{D}$, and the derivative with respect to $\hat{\mathbf{p}}^{n+1}$ is zero. We use the all-directional implicit model to construct the implicit matrix $\mathbf{D}$, because the extra damping effects provided by this model are desirable for stabilizing the Newton-Raphson iterations. The approximated Jacobian can then be written as

$$\mathbf{J}_k = \begin{pmatrix} \hat{\mathbf{M}}_k - \Delta t^2 \mathbf{D}_k & \hat{\mathbf{G}}_k \\ \hat{\mathbf{G}}_k^T & \mathbf{0} \end{pmatrix},$$

where we stress that $\mathbf{J}_k$ is updated with new values of $\hat{\mathbf{M}}_k$, $\hat{\mathbf{G}}_k$ and $\mathbf{D}_k$ at each iteration.

To solve equations (54) and (55), they are first transformed into an SPD system following a derivation similar to that in Section 7.2. Substituting the definitions of $\mathbf{J}$ and $\mathbf{F}$ into equation (54) yields

$$\hat{\mathbf{M}}_k \Delta \hat{\mathbf{u}} + \hat{\mathbf{G}}_k \Delta \hat{\mathbf{p}} - \Delta t^2 \mathbf{D}_k \Delta \hat{\mathbf{u}} = -\hat{\mathbf{M}}_k (\hat{\mathbf{u}}_k - \hat{\mathbf{u}}^n) - \hat{\mathbf{G}}_k \hat{\mathbf{p}}_k + \Delta t \hat{\mathbf{f}}_k + \Delta t \hat{\mathbf{M}}_k \hat{\mathbf{g}}, \tag{56}$$

$$\hat{\mathbf{G}}_k^T \Delta \hat{\mathbf{u}} = -\hat{\mathbf{G}}_k^T \hat{\mathbf{u}}_k, \tag{57}$$

and substituting the definition of $\mathbf{w}$ into equation (55) yields

$$\hat{\mathbf{u}}_{k+1} = \hat{\mathbf{u}}_k + \Delta \hat{\mathbf{u}}, \tag{58}$$

$$\hat{\mathbf{p}}_{k+1} = \hat{\mathbf{p}}_k + \Delta \hat{\mathbf{p}}. \tag{59}$$

Equation (56) is combined with equation (59) to eliminate $\Delta \hat{\mathbf{p}}$ as

$$\hat{\mathbf{M}}_k \Delta \hat{\mathbf{u}} - \Delta t^2 \mathbf{D}_k \Delta \hat{\mathbf{u}} = -\hat{\mathbf{M}}_k (\hat{\mathbf{u}}_k - \hat{\mathbf{u}}^n) - \hat{\mathbf{G}}_k \hat{\mathbf{p}}_{k+1} + \Delta t \hat{\mathbf{f}}_k + \Delta t \hat{\mathbf{M}}_k \hat{\mathbf{g}}. \tag{60}$$

Left multiplying by $\hat{\mathbf{M}}_k^{-1}$ on both sides of equation (60) and rearranging terms yields

$$\Delta \hat{\mathbf{u}} = \hat{\mathbf{u}}^* - \hat{\mathbf{M}}_k^{-1} \hat{\mathbf{G}}_k \hat{\mathbf{p}}_{k+1} + \Delta t^2 \hat{\mathbf{M}}_k^{-1} \mathbf{D}_k \Delta \hat{\mathbf{u}}, \tag{61}$$

where

$$\hat{\mathbf{u}}^* = \hat{\mathbf{u}}^n - \hat{\mathbf{u}}_k + \Delta t \hat{\mathbf{M}}_k^{-1} \hat{\mathbf{f}}_k + \Delta t \hat{\mathbf{g}}. \tag{62}$$

Substituting the factorization $\mathbf{D}_k = -\mathbf{C}_k^T \mathbf{C}_k$ into equation (61) yields

$$\Delta \hat{\mathbf{u}} = \hat{\mathbf{u}}^* - \hat{\mathbf{M}}_k^{-1} \hat{\mathbf{G}}_k \hat{\mathbf{p}}_{k+1} - \Delta t^2 \hat{\mathbf{M}}_k^{-1} \mathbf{C}_k^T \mathbf{C}_k \Delta \hat{\mathbf{u}}.$$

Rewriting this using block matrices yields

$$\Delta \hat{\mathbf{u}} = \hat{\mathbf{u}}^* - \hat{\mathbf{M}}_k^{-1} \begin{pmatrix} \hat{\mathbf{G}}_k^T \\ \Delta t \mathbf{C}_k \end{pmatrix}^T \begin{pmatrix} \hat{\mathbf{p}}_{k+1} \\ \mathbf{C}_k \Delta \hat{\mathbf{u}} \end{pmatrix},$$

which can be written compactly as

$$\Delta \hat{\mathbf{u}} = \hat{\mathbf{u}}^* - \hat{\mathbf{M}}_k^{-1} \mathbf{K}_k^T \hat{\mathbf{z}}, \tag{63}$$

where

$$\mathbf{K}_k = \begin{pmatrix} \hat{\mathbf{G}}_k^T \\ \Delta t \hat{\mathbf{C}}_k \end{pmatrix}, \quad \hat{\mathbf{z}} = \begin{pmatrix} \hat{\mathbf{p}}_{k+1} \\ \Delta t \mathbf{C}_k \Delta \hat{\mathbf{u}} \end{pmatrix}. \tag{64}$$

Left multiplying by $\mathbf{K}_k$ on both sides of equation (63) and combining it with equation (57) produces an SPD system

$$(\mathbf{P} + \mathbf{K}_k \hat{\mathbf{M}}_k^{-1} \mathbf{K}_k^T)\hat{\mathbf{z}} = \mathbf{K}_k \hat{\mathbf{u}}^* + \begin{pmatrix} \mathbf{G}_k^T \hat{\mathbf{u}}_k \\ \mathbf{0} \end{pmatrix}. \tag{65}$$

Note that if the viscous term is present, it can be coupled into equation (65) as shown in [42]. Equation (65) is solved using the conjugate gradient method to obtain $\hat{\mathbf{z}}$, which is subsequently substituted into equation (63) to obtain $\Delta\hat{\mathbf{u}}$. The new prediction of the velocity $\hat{\mathbf{u}}_{k+1}$ is then updated using equation (58).

At this point, we have obtained $\hat{\mathbf{u}}_{k+1}$ at each particle location $\hat{\mathbf{x}}_k$. The next step is to move particles from their current location $\hat{\mathbf{x}}_k$ to their new predicted locations $\hat{\mathbf{x}}_{k+1}$ carrying over $\hat{\mathbf{u}}_{k+1}$ and $\hat{\mathbf{u}}^n$ with them. The predicted locations at two consecutive iterations k and $k+1$ are respectively

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}^n + \Delta t \hat{\mathbf{u}}_k, \tag{66}$$

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}^n + \Delta t \hat{\mathbf{u}}_{k+1}. \tag{67}$$

Subtracting these two equations yields

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{x}}_k + \Delta t \Delta \hat{\mathbf{u}}. \tag{68}$$

With a large update $\Delta t \Delta \hat{\mathbf{u}}$, some mesh elements may be inverted after solving equation (68). This results in an invalid discretization. To avoid this problem, $\Delta\hat{\mathbf{u}}$ is scaled down by half if any mesh elements are inverted, and this process is repeated until no mesh elements are inverted. Equation (58) is updated accordingly. After moving particles to $\hat{\mathbf{x}}_{k+1}$, we remesh to maintain a good quality mesh, and recompute $\hat{\mathbf{M}}$, $\hat{\mathbf{G}}$ and $\mathbf{D}$ completing one iteration of our nonlinear solver.

The oscillating ellipse from Section 7.2.1 is used to test stability. The proposed scheme achieves a steady state solution after one time step and remains stable for all tested selections of Δx and Δt in Figure 32. However, the droplet gains 69.64% of its original volume as shown in Figure 33(b).

7.4. Fully Implicit Volume Conserving Scheme with a Lagrangian Mesh

Starting with the integral form of the continuity equation and dividing by the constant density results in the typical incompressible flow equation for conservation of volume

$$\frac{DV}{Dt} = -\int_V \nabla \cdot \vec{u}\, dV. \tag{69}$$

Letting V be the particle control volume and writing equation (69) for each particle results in

$$\frac{D\hat{\mathbf{v}}}{Dt} = \hat{\mathbf{G}}^T \hat{\mathbf{u}}, \tag{70}$$

where $\hat{\mathbf{v}}$ is the vector of particle control volumes defined on the Lagrangian mesh and $-\hat{\mathbf{G}}^T$ is the volume-weighted divergence operator. The temporal discretization of equation (70) using the backward Euler method is

$$\frac{1}{\Delta t}(\hat{\mathbf{v}}^{n+1} - \hat{\mathbf{v}}^n) = (\hat{\mathbf{G}}^{n+1})^T \hat{\mathbf{u}}^{n+1}. \tag{71}$$

Comparing this with the discretization in Section 7.3, we see that one can replace equation (53) with a discrete constraint on the volume, after which equation (65) becomes

$$(\mathbf{P} + \mathbf{K}_k \hat{\mathbf{M}}_k^{-1} \mathbf{K}_k^T)\hat{\mathbf{z}} = \mathbf{K}_k \hat{\mathbf{u}}^* + \begin{pmatrix} \frac{1}{\Delta t}(\hat{\mathbf{v}}_k - \hat{\mathbf{v}}^n) \\ \mathbf{0} \end{pmatrix}. \tag{72}$$

The nonlinear solver then proceeds as follows. We start with an initial guess $\hat{\mathbf{u}}_0 = 0$, $\hat{\mathbf{p}}_0 = 0$, $\hat{\mathbf{x}}_0 = \hat{\mathbf{x}}^n$ and $\hat{\mathbf{v}}_0 = \hat{\mathbf{v}}^n$. At each iteration, the current control volume $\hat{\mathbf{v}}_k$ is computed using $\hat{\mathbf{x}}_k$ and the mesh topology. We then solve equations (72) and (63) to obtain $\hat{\mathbf{p}}_{k+1}$ and $\Delta\hat{\mathbf{u}}$. Note that the current volume error is fed back into the system at every iteration. $\Delta\hat{\mathbf{u}}$ is still scaled to maintain a valid mesh as discussed in Section 7.3, which is then used in equations (58) and (68) to obtain $\hat{\mathbf{u}}_{k+1}$ and $\hat{\mathbf{x}}_{k+1}$. Finally, the mesh is regenerated from $\hat{\mathbf{x}}_{k+1}$ to maintain a good quality mesh.

An important consideration when evaluating the right hand side of equation (72) is that the remeshing done after each iteration changes the particle control volumes $\hat{\mathbf{v}}$. One can address this using a Lagrangian plus remap method (see e.g. [6] and the references therein), but this is overkill for incompressible flow especially since we only use the volume temporarily for stability in our iterative approach to surface tension. Thus, we take a simple local diffusion approach instead.

To identify the volume errors caused by topology changes, we rewind the new mesh at the $(k+1)$-th iteration back to the $\hat{\mathbf{x}}_k$ locations and denote this as mesh T^{new} noting that it differs from the mesh T^{old} at the k-th iteration only in topology. For each particle i, we compute its control volumes from the mesh geometry on both mesh T^{old} and T^{new} to obtain volumes V_i^{old} and V_i^{new}. Particle i takes place in the diffusion process only if $|V_i^{\text{new}} - V_i^{\text{old}}|$ is greater than a tolerance ϵ, and once $|V_i^{\text{new}} - V_i^{\text{old}}| < \epsilon$ we remove that particle from the diffusion process. Iterations are done in a Gauss-Seidel fashion as follows. For each particle i, we compute the total combined volume for it and all of its logically connected neighbors in its one ring on mesh T^{new} and denote this as V_C^{new}. V_C^{old} is calculated in the same exact fashion using V_i^{old}, stressing that the one ring according to mesh T^{new} is used not the one ring according to T^{old}. The diffusion redistributes the total volume V_C^{old} to the particle i and its one ring neighbors j based on the ratio $V_j^{\text{new}}/V_C^{\text{new}}$, i.e. V_j^{old} is assigned the value $V_C^{\text{old}}(V_j^{\text{new}}/V_C^{\text{new}})$. Note that V_j^{new} never changes as we iterate, V_C^{new} changes only when a particle has been removed from the diffusion process, and V_C^{old} is updated based on the new values of V_j^{old}. This diffusion process typically takes around 10 to 20 iterations.

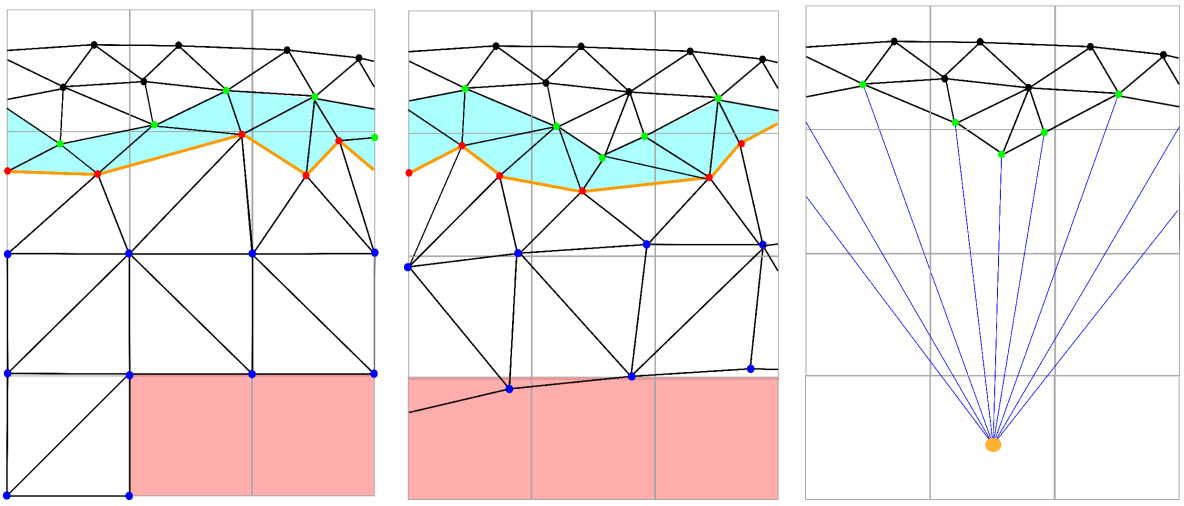
The real goal here is to update $\hat{\mathbf{v}}^n$ in equation (72) based on the changes of mesh topology, and the change between $\hat{\mathbf{v}}^{\text{new}}$ and $\hat{\mathbf{v}}^{\text{old}}$ is *only* used as a guide for how that should be accomplished. $\hat{\mathbf{v}}^n$ is computed at the beginning of the iterative process based on mesh topology, and then modified every time we proceed from iteration k to iteration $k+1$ using remeshing. It is modified in parallel with the diffusion process described above where every time V_C^{old} is redistributed to V_j^{old} based on the weights $V_j^{\text{new}}/V_C^{\text{new}}$ we likewise redistribute V_C^n to V_j^n based on the same weights. The iterative procedure, stopping criteria, etc. are all as previously described, we just update $\hat{\mathbf{v}}^n$ in parallel. At the end of the diffusion process, only $\hat{\mathbf{v}}^n$ is needed by the next iteration in equation (72), and both $\hat{\mathbf{v}}^{\text{old}}$ and $\hat{\mathbf{v}}^{\text{new}}$ are discarded.

Remeshing occasionally requires new particles, and we add those to the new mesh in the same way as in Section 2. These new particles are rewound in the same fashion as the others when computing the mesh T^{new}. Their control volumes on mesh T^{new} are properly defined, but they lack of values for V^{old} and V^n. We compute these as follows. If a new particle was inserted at the midpoint of two old particles, the sum of the old control volumes of the two old particles is evenly divided into three parts and assigned to each of the three particles. If two old particles were merged into a new particle, the sum of their old control volumes is assigned to the new particle. After this initialization, the iterative diffusion proceeds as usual.

The oscillating ellipse from Section 7.2.1 is used to test stability. The proposed scheme achieves a steady state solution after one time step and remains stable for all tested selections of Δx and Δt in Figure 32. In addition, the relative volume error is only $-.07\%$ as shown in Figure 33(c).

7.5. Fully Implicit Volume Conserving Scheme with a Hybrid Particle Grid Structure

The spatial discretization follows that in Section 6, and all vectors and matrices are redefined accordingly. $\hat{\mathbf{u}}$ now consists of velocities on both particles and MAC grid faces, $\hat{\mathbf{p}}$ consists of pressures on both particles and MAC grid cell centers, and $\hat{\mathbf{v}}$ consists of a hybrid set of particle control volumes and MAC grid cell control volumes. The volume-weighted divergence $-\hat{\mathbf{G}}^T$ and the volume-weighted gradient $\hat{\mathbf{G}}$ are now defined on the hybrid structure as in Section 4. The surface tension force and its implicit matrix $\mathbf{D}$ are still computed using the method in Section 7.1. The nonlinear solver takes the same steps as in Section 7.4 but with the following changes.



(a) The hybrid structure T^{old} at the old iteration.

(b) The hybrid structure T^{new} rewound from the new iteration.

(c) The auxiliary mesh T^{aux} for diffusion.

Figure 34: Topology changes of the hybrid structure between two iterations.

Equation (68) is changed to an advection operator on the hybrid particle grid structure. Recall that when solving equation (68), $\hat{\mathbf{u}}_{k+1}$ and $\hat{\mathbf{u}}^n$ are carried over from $\hat{\mathbf{x}}_k$ to $\hat{\mathbf{x}}_{k+1}$ along with the particles. So with the hybrid particle grid structure, they are advected from the hybrid structure at the k-th iteration to the hybrid structure at the $(k+1)$-th iteration using the advection method from Section 3, where $\Delta\hat{\mathbf{u}}$ is used as the background velocity. Repeatedly using $\Delta\hat{\mathbf{u}}$ for advection increases interpolation errors on the MAC grid, and although this could be avoided we found it unnecessary since the MAC grid cells are not so close to the interface where the surface tension forces are prevalent.

The volume conservation scheme requires modification in order to handle the Eulerian degrees of freedom, since the interior MAC grid cells and the grid nodes of buffer cells are fixed in space and cannot be explicitly tracked in a Lagrangian fashion. To avoid this problem, we lump the entire interior MAC grid volume (the red cells in Figure 34(a)), the grid nodes of the buffer cells (the blue dots) and the one ring of Lagrangian mesh around the buffer cells (the red dots) into a single control volume V_P where P indicates a composite set of degrees of freedom. Note that V_P contains the entire region inside the yellow lines in Figure 34(a) as well as the portion of the blue triangles that are lumped into the red nodes in the standard fashion. At the beginning of the time step, $\hat{\mathbf{v}}^n$ is computed as usual noting that the MAC grid cell degrees of freedom have their control volumes defined as in Section 4.1, and then V_P^n is computed. At each iteration, $\hat{\mathbf{v}}_k$ is computed for all degrees of freedom (including those in P) using the current hybrid structure. Then for each degree of freedom $i \in P$ we modify $\hat{\mathbf{v}}^n$ via $V_P^n([\hat{\mathbf{v}}_k]_i / \sum_{j\in P}[\hat{\mathbf{v}}_k]_j)$ where $[\hat{\mathbf{v}}_k]_i$ is the i-th entry of $\hat{\mathbf{v}}_k$. This redistributes the volume of V_P^n to the degrees of freedom in P according to their current volume fractions in the k-th iteration. $\hat{\mathbf{v}}_k$ and the modified $\hat{\mathbf{v}}^n$ are then used on the right hand side of equation (72) to correct volume errors.

In order to handle the volume errors due to topology changes of the hybrid particle grid structure, we modify the diffusion process from Section 7.4 as follows. We rewind the new tessellated Lagrangian mesh at the $(k+1)$-th iteration back to the $\hat{\mathbf{x}}_k$ locations and denote this as T^{new} (Figure 34(b)) noting that it not only differs from the old hybrid structure T^{old} in topology but may also have a different set P. To distinguish these two different sets of P we denote them as P^{old} and P^{new}, and we bear this difference in mind when initializing control volumes. On T^{old} and T^{new}, first V_i^{old}, V_P^{old}, V_i^{new} and V_P^{new} are computed as usual. Then if a particle was untessellated on T^{old} and becomes tessellated on T^{new}, we estimate its V^{old} value as $V_P^{\text{old}}(V_i^{\text{new}}/V_P^{\text{new}})$, and subtract this value from V_P^{old} if the particle is not in P^{new}. If a particle was tessellated on T^{old} and becomes untessellated on T^{new}, its V^{old} value is added to V_P^{old} if it was not already

in P^{old}. If a particle is in P^{new} but was not in P^{old}, its V^{old} value is added to V_P^{old}. If a particle was in P^{old} but is not in P^{new}, its V^{old} value is subtracted from V_P^{old}. $\hat{\mathbf{v}}^n$ and V_P^n are adjusted in the same fashion as $\hat{\mathbf{v}}^{\text{old}}$ and V_P^{old}. An auxiliary mesh T^{aux} is created from T^{new} for the diffusion process (see Figure 34(c)), where P^{new} is represented by a single degree of freedom (the yellow dot). The yellow dot is connected to the green particles so that the green particles become the one ring neighbors of the yellow dot recalling that this connectivity information is used in the diffusion process to find a local set of neighbors for volume redistribution.

The oscillating ellipse from Section 7.2.1 is used to test stability. The proposed scheme achieves a steady state solution after one time step and remains stable for all tested selections of Δx and Δt in Figure 32. In addition, the relative volume error is $-.088\%$ as shown in Figure 33(d).

7.6. Fully Implicit Volume Conserving Scheme with a MAC Grid

The spatial discretization follows that in [43] with velocities and pressures defined on a regular MAC grid. Control volumes, the volume-weighted gradient and the volume-weighted divergence are defined using a cut-cell discretization modified from [43] as described in Section 7.6.1 below. The free surface is represented either by a standard front tracking method [59, 56] or by the particle level set method [20] where a mesh of line segments is extracted from the level set using the third order contouring method from [43]. Both approaches maintain an explicit representation of the surface mesh, and the surface tension force and its implicit matrix $\mathbf{D}$ are computed on this surface mesh using the method from Section 7.1. The nonlinear solver takes the same steps as in Section 7.4 but with the following modifications.

Since the surface tension force and the pressure force are now discretized on different degrees of freedom, we couple them by interpolating velocities from the MAC grid to the surface mesh using an interpolation matrix $\mathbf{H}$ and by distributing forces from the surface mesh back to the MAC grid using $\mathbf{H}^T$. Equation (56) is modified accordingly to use this coupling scheme as

$$\hat{\mathbf{M}}_k\Delta\hat{\mathbf{u}} + \hat{\mathbf{G}}_k\Delta\hat{\mathbf{p}} - \Delta t^2\mathbf{H}_k^T\mathbf{D}_k\mathbf{H}_k\Delta\hat{\mathbf{u}} = -\hat{\mathbf{M}}_k(\hat{\mathbf{u}}_k - \hat{\mathbf{u}}^n) - \hat{\mathbf{G}}_k\hat{\mathbf{p}}_k + \Delta t\mathbf{H}_k^T\hat{\mathbf{f}}_k + \Delta t\hat{\mathbf{M}}_k\hat{\mathbf{g}}.$$

Propagating this change modifies equations (62) and (64) to

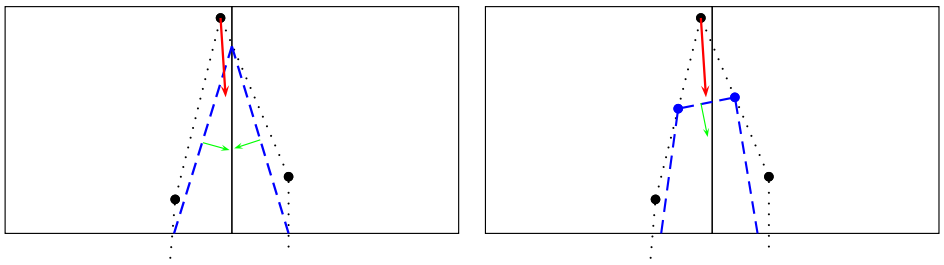
$$\hat{\mathbf{u}}^* = \hat{\mathbf{u}}^n - \hat{\mathbf{u}}_k + \Delta t\hat{\mathbf{M}}_k^{-1}\mathbf{H}_k^T\hat{\mathbf{f}}_k + \Delta t\hat{\mathbf{g}}, \quad \mathbf{K}_k = \begin{pmatrix} \hat{\mathbf{G}}_k^T \\ \Delta t\mathbf{C}_k\mathbf{H}_k \end{pmatrix}, \quad \hat{\mathbf{z}} = \begin{pmatrix} \hat{\mathbf{p}}_{k+1} \\ \Delta t\mathbf{C}_k\mathbf{H}_k\Delta\hat{\mathbf{u}} \end{pmatrix},$$

which are then used when solving equations (72) and (63).

Equation (68) is changed to an advection operator on the MAC grid. Similar to the discussion in Section 7.5, velocities $\hat{\mathbf{u}}_{k+1}$ and $\hat{\mathbf{u}}^n$ are advected from the k-th iteration to the $(k+1)$-th iteration with $\Delta\hat{\mathbf{u}}$ as the background velocity where third order accurate Hamilton-Jacobi ENO [39] and third order accurate TVD Runge Kutta [48] are used. For the front tracking method, the surface mesh is advected using the forward Euler method. For the particle level set method, the marker particles and the level set are separately advected using a third order accurate TVD Runge-Kutta method [30, 48], and the spatial derivatives are calculated with a fifth order accurate Hamilton-Jacobi WENO scheme [29].

At the beginning of each time step, we compute the total volume of the fluid V^n enclosed by the surface mesh. At each iteration, we also compute the total volume V. One can think of V^n and V as V_P^n and V_P except that now P represents all degrees of freedom. $\hat{\mathbf{v}}^n$ and $\hat{\mathbf{v}}_k$ are obtained by redistributing V^n and V to the MAC grid cells using the weights $V_c/(\sum_c V_c)$ where V_c is the cell volume, i.e. V_c is 0, Δx^2 or the cut cell volume. Note that this global volume conservation scheme is similar in spirit to [52, 53].

The update $\Delta\hat{\mathbf{u}}$ is scaled differently in order to stabilize the nonlinear iteration. As discussed in [43], the $\mathbf{H}^T$ matrix is constructed so that it only transfers the normal components of the surface tension force to the MAC grid in order to reduce parasitic currents. And via its transpose, the $\mathbf{H}$ matrix only interpolates the normal components of the velocity to the surface mesh. This prevents the damping of the tangential components of the velocity making the all-directional implicit model behave like a normal-directional model and thus become less stable. Thus, to stabilize the iterations we scale down the update $\Delta\hat{\mathbf{u}}$ using a CFL condition $\|\Delta t\Delta\hat{\mathbf{u}}\| \leq \alpha\Delta x$ where α is the CFL number.



(a) Original cut-cell discretization (b) Modified cut-cell discretization

Figure 35: Cut-cell discretization. The black dotted lines are the surface mesh, and the blue dashed lines are the faces constructed for each cut cell. The red arrow is the surface tension force, and the green arrows are the normal directions of the faces.

7.6.1. Modifications to [43]

We modify the cut-cell discretization from [43] in order to handle sharp features on the surface mesh. In [43], a face is constructed for each cut cell by connecting the two intersection points cut by the surface mesh (the blue dashed lines in Figure 35(a)), and then the surface tension force (the red arrow) is projected perpendicular to these faces using $\mathbf{H}^T$. However, the face normals may significantly differ from the force direction for sharp features, and thus a majority of the force can be lost during projection. To avoid this problem, we instead construct a face for each node of the surface mesh by connecting the two midpoints of its two incident segments (the blue dashed lines in Figure 35(b)), so that the face normal is now parallel to the area-weighted normal of its corresponding node giving a better approximation to the force direction.

We modify the method proposed in [43] to better conserve momentum. First, we compute the total projected force by summing over all faces that are coupled with surface tension as

$$\vec{f}_{total} = \sum_c f_c \vec{n}_c,$$

where f_c is the projected value on the coupling face c, and $\vec{n}_c$ is the outward pointing normal. In order to enforce a zero total force, we apply a correction force g_c on each coupling face so that

$$\sum_c (f_c + g_c)\vec{n}_c = 0.$$

Rearranging terms yields

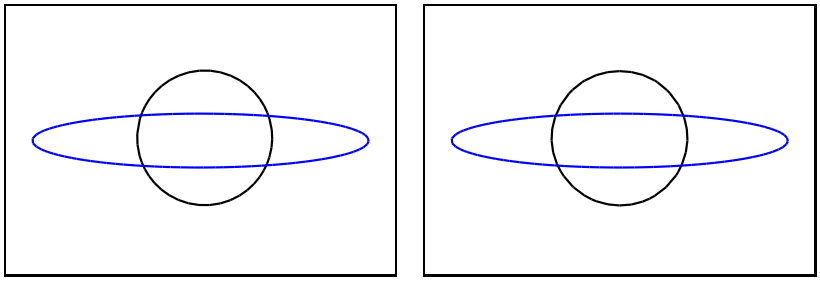
$$\sum_c g_c \vec{n}_c = \sum_c -f_c \vec{n}_c = -\vec{f}_{total}.$$

This is an underdetermined system which can be compactly written as

$$\mathbf{N}\mathbf{g} = -\vec{f}_{total}, \tag{73}$$

where the columns of $\mathbf{N}$ are the $\vec{n}_c$ vectors, and $\mathbf{g}$ is a vector of g_c. To obtain the minimum-norm solution, the QR factorization is used to obtain $\mathbf{N}^T = \mathbf{Q}\mathbf{R}$. Since $\mathbf{N}^T$ only has two columns, this is done by a single iteration of the Gram-Schmidt process. Then equation (73) is solved as

$$\mathbf{R}^T\mathbf{Q}^T\mathbf{g} = -\vec{f}_{total},$$
$$\mathbf{g} = -\mathbf{Q}\mathbf{R}^{-T}\vec{f}_{total}.$$



(a) front tracking (b) particle level set

Figure 36: The free surface of the oscillating ellipse test with $\Delta t = .25\,s$. The fully implicit volume conserving scheme on the MAC grid is used. The blue line shows the initial free surface, and the black line shows the free surface after a single time step. Subfigure (a) shows the result using the front tracking method, and Subfigure (b) shows the result using the particle level set method.

The g_c values can then be added to the coupling faces to exactly conserve momentum. However, we only use this technique on the explicit part of the surface tension force. The implicit part of the force is tightly coupled with the pressure, and changing it either breaks the symmetry of the system or breaks the balance between the surface tension force and the pressure force. Fortunately, the explicit part of the force is usually much stronger than the implicit part, and it accounts for the majority of momentum loss.

7.6.2. Stability Tests

The oscillating ellipse from Section 7.2.1 is used to test stability. The proposed scheme with both standard front tracking and the particle level set method achieves a steady state solution after one time step and remains stable for all tested selections of Δx and Δt in Figure 32. In addition, the relative volume error with the standard front tracking method is .023% as shown in Figure 36(a), and that with the level set method is $-.042\%$ as shown in Figure 36(b).

7.7. Numerical Results

Consider an oscillating circle test from [43] with a computational domain $[-2\,m, 2\,m] \times [-2\,m, 2\,m]$. The initial droplet interface is a deformed circle centered at the origin whose perturbed radius is described by $R(\theta) = a + \epsilon\cos(n\theta)$ where θ runs between 0 and 2π. The linearized analytic solution of the radius in the absence of viscosity (see [33]) is $R(\theta, t) = a + \epsilon\cos(n\theta)\cos(\omega t)$ where $\omega^2 = \frac{\sigma(n^2-1)n}{\rho a^3}$. We use the parameters $a = 1\,m$, $\epsilon = .05\,m$ and $n = 2$. The liquid has parameters $\rho = 1\,kg\,m^{-2}$ and $\sigma = 1\,kg\,s^{-2}$. The fully implicit volume conserving schemes from Sections 7.4, 7.5 and 7.6 are tested. Table 22 shows the relative error between succeeding resolutions for the computed location of $R(0, t)$, where the L^1 error is averaged from 100 sample points in time with an interval .01 s and the L^∞ error is the maximum of those. In order to show that the simulation reasonably captures the correct amplitude and frequency of the oscillating circle, Figure 37 compares the simulated $R(0, t)$ locations with the analytic solution.

Consider a rising bubble example modified from [43] with a computational domain $[0\,m, 1\,m] \times [0\,m, 2\,m]$ filled with liquid. A circular vacuum bubble is given an initial radius of $.2\,m$ and is centered at $[.5\,m, .5\,m]$. The liquid has parameters $\rho = 20\,kg\,m^{-2}$, $\mu = 1\,kg\,s^{-1}$ and $\sigma = .0728\,kg\,s^{-2}$. The semi-implicit scheme from [43], the semi-implicit scheme from Section 7.2 and the fully implicit volume conserving schemes from Sections 7.4 and 7.5 are tested. The scheme from [43] remains stable up to $\Delta t = .03\,s$ for both particle level set and front tracking methods, the scheme from Section 7.2 remains stable up to $\Delta t = .07\,s$, and the schemes from Sections 7.4 and 7.5 both remain stable up to $\Delta t = .25\,s$. With $\Delta t > .25\,s$ the nonlinear solvers from Sections 7.4 and 7.5 fail to converge, possibly due to the fact that an approximate Jacobian is used.

To test the ability of our solver to handle complex scenarios, we implement contact angle handling. When the surface mesh is in contact with the boundary of the computational domain, we do not use the mesh elements whose nodes are all on the boundary of the computational domain to compute the surface

Table 22: The order of accuracy for the oscillating circle test using the fully implicit volume conserving scheme with various spatial discretizations.

Lagrangian mesh				
Δx	L^1 Error	Order	L^∞ Error	Order
.064	–	–	–	–
.032	6.26×10^{-5}	–	6.45×10^{-4}	–
.016	3.53×10^{-5}	.83	3.59×10^{-4}	.85
.008	1.78×10^{-5}	.98	1.72×10^{-4}	1.06
Hybrid particle grid structure				
Δx	L^1 Error	Order	L^∞ Error	Order
.064	–	–	–	–
.032	6.44×10^{-5}	–	6.84×10^{-4}	–
.016	3.98×10^{-5}	.69	3.99×10^{-4}	.78
.008	1.98×10^{-5}	1.01	1.87×10^{-4}	1.09
Front tracking				
Δx	L^1 Error	Order	L^∞ Error	Order
.064	–	–	–	–
.032	1.02×10^{-3}	–	2.31×10^{-3}	–
.016	5.40×10^{-4}	.91	1.12×10^{-3}	1.05
.008	2.41×10^{-4}	1.17	5.32×10^{-4}	1.07
Particle level set				
Δx	L^1 Error	Order	L^∞ Error	Order
.064	–	–	–	–
.032	1.73×10^{-3}	–	3.81×10^{-3}	–
.016	8.61×10^{-4}	1.00	1.92×10^{-3}	.99
.008	4.56×10^{-4}	.92	9.43×10^{-4}	1.03

tension force, and instead enforce a contact angle boundary condition as follows. In two spatial dimensions, the fluid surface at the contact line is virtually extended into the wall along a direction $\vec{t_c}$ (the green line in Figure 38(a)) so that the angle between the extended surface and the wall equals the contact angle θ. This extended surface then applies a force to the particle on the contact line as $\vec{f_c} = \sigma \vec{t_c}$. In three spatial dimensions, the contact line consists of a set of line segments, and from each segment s the fluid surface is virtually extended into the wall along a direction $\vec{t_c}$ perpendicular to the segment s (the green plane in Figure 38(b)) so that the angle between the extended surface and the wall equals the contact angle θ. This extended surface then applies a force to segment s as $\vec{f_c} = \sigma l_s \vec{t_c}$, where l_s is the length of the segment s. Half of this force is then applied to each of the two particles connected by segment s. We project the total force at each particle on the contact line perpendicular to the wall normal. For the implicit forces, these projections are composed into a matrix $\mathbf{P}$ and then used to modify the implicit matrix $\mathbf{D}$ into $\mathbf{PDP}$. Note that the second $\mathbf{P}$ is used to maintain symmetry since $\mathbf{P} = \mathbf{P}^T$, and it does not change the behavior of the implicit matrix since it merely projects off the zero normal velocity component on the wall. The $\mathbf{C}$ matrix is accordingly modified to obtain $\mathbf{CP}$.

Consider a droplet impact example modified from Section 6. The computational domain is $[0\,m, .2\,m] \times [0\,m, .15\,m]$ which is partially filled with liquid where the free surface is located at $y = .05\,m$. A circular liquid droplet of radius $r = .012\,m$ is centered at $[.1\,m, .0624\,m]$ and has an initial velocity $[0\,m\,s^{-1}, -8\,m\,s^{-1}]$. The liquid has parameters $\rho = 1000\,kg\,m^{-2}$ and $\sigma = 1\,kg\,s^{-2}$. The contact angle is set to be 90° on all walls. Figure 39 shows the free surface using the fully implicit volume conserving scheme from Section 7.5 with a grid of resolution 400×300. Note the small details our solver is able to resolve and track stably over time.

We extended the oscillating ellipse problem from Section 7.2.1 to three spatial dimensions. The computational domain is $[-.005\,m, .005\,m] \times [-.005\,m, .005\,m] \times [-.005\,m, .005\,m]$. A liquid droplet is initially

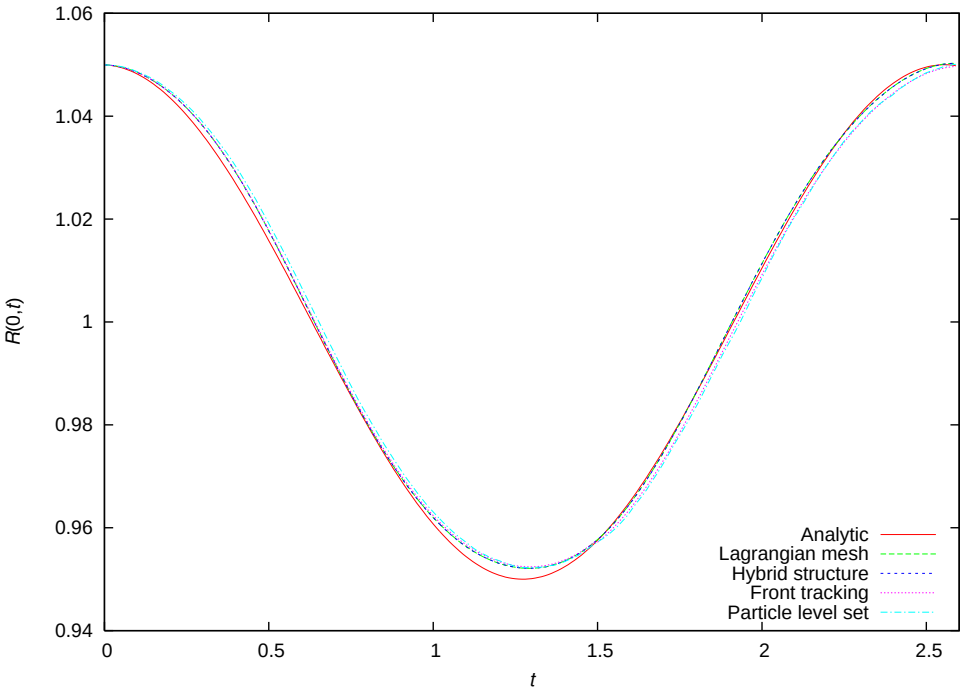


Figure 37: The simulated $R(0,t)$ locations compared with the analytic solution for the oscillating circle test. $\epsilon = .05\,m$ and a spatial resolution of $\Delta x = .008\,m$ are used in all schemes.

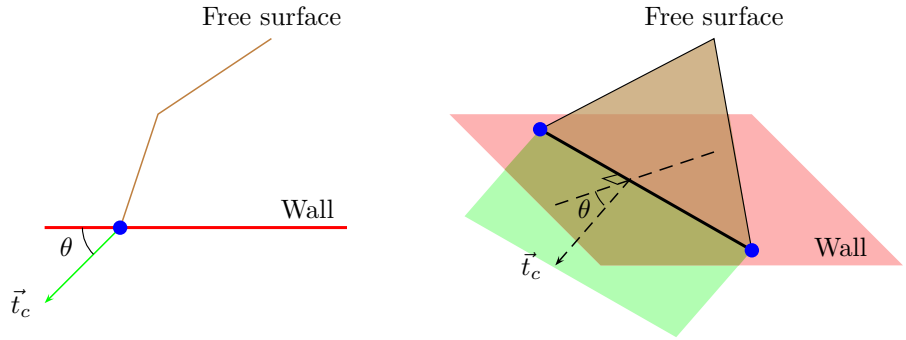
Table 23: The breakup time for the 3D Rayleigh capillary instability test.

Grid Cells	Breakup Time	Difference	Order
32	$1.14 \times 10^{-5}\,s$	–	–
64	$1.11 \times 10^{-5}\,s$	$3 \times 10^{-7}\,s$	–
128	$1.10 \times 10^{-5}\,s$	$1 \times 10^{-7}\,s$	1.58

centered at the origin with an ellipsoid interface described by $(\frac{x}{r})^2 + (\frac{y}{r/\alpha})^2 + (\frac{z}{r/\alpha})^2 = 1$ where $r = .0045\,m$ is the radius along the x axis, and r/α is the radius along the y and z axes where $\alpha = 2.5$. The liquid has parameters $\rho = 1000\,kg\,m^{-3}$, $\mu = 1.138 \times 10^{-3}\,kg\,m^{-1}\,s^{-1}$ and $\sigma = .0728\,kg\,s^{-2}$. The fully implicit volume conserving scheme from Section 7.5 is used with a grid of resolution $32 \times 32 \times 32$. As shown in Figure 40, the droplet achieves a steady state solution after a very large time step $\Delta t = .25\,s$, and the relative volume error is only .014%.

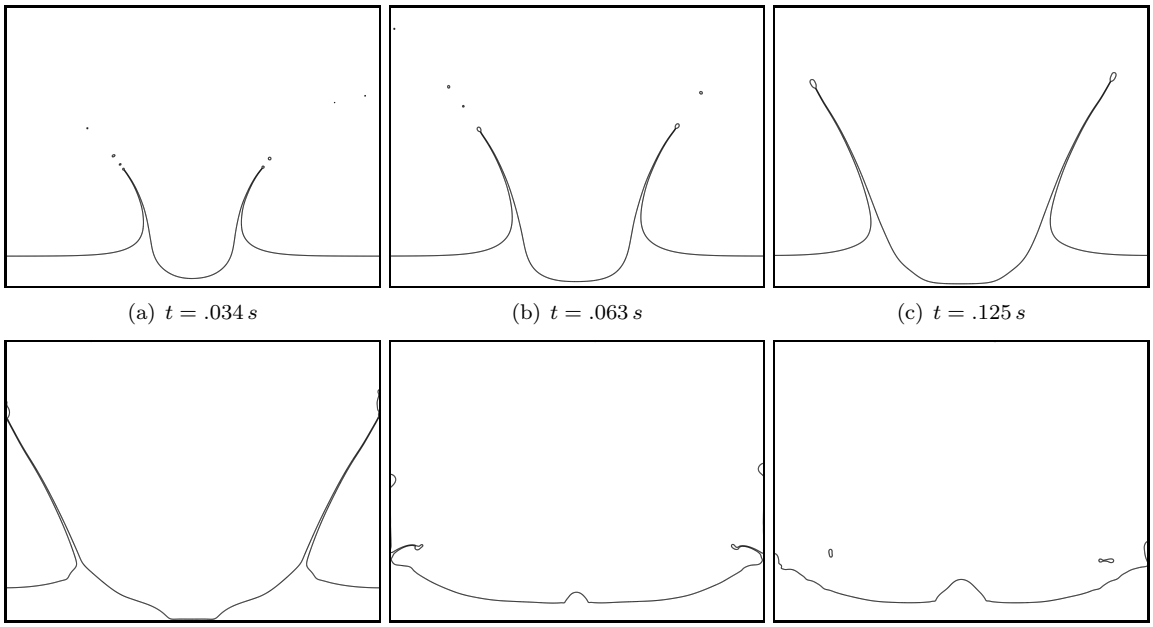
Consider the Rayleigh capillary instability test taken from [54]. The computational domain is $[-1.5 \times 10^{-5}\,m, 1.5 \times 10^{-5}\,m] \times [-1.5 \times 10^{-5}\,m, 1.5 \times 10^{-5}\,m] \times [0\,m, 3 \times 10^{-5}\,m]$. A slightly perturbed cylindrical column of liquid has its initial interface described as $R(z) = R_0 + \epsilon\cos(2\pi z/\lambda)$ where $R_0 = 6.52 \times 10^{-6}\,m$, $\epsilon = 1.3 \times 10^{-6}\,m$ and $\lambda = 6 \times 10^{-5}\,m$. The liquid has parameters $\rho = 10^3\,kg\,m^{-3}$, $\mu = 1.138 \times 10^{-3}\,kg\,m^{-1}\,s^{-1}$ and $\sigma = .0728\,kg\,m\,s^{-2}$. No gravity is used. The liquid has zero initial velocity and is driven to break up into droplets by surface tension. The hybrid particle grid structure and the fully implicit volume conserving scheme from Section 7.5 are used with three grids of resolutions 32^3, 64^3 and 128^3. Figure 41 shows the results with the grid of resolution 128^3, and Table 23 shows the convergence of the breakup time under grid refinement.

In order to test our volume conserving scheme in complex scenarios, we modified the above Rayleigh capillary instability test by extending the computational domain to $[-1.5 \times 10^{-5}\,m, 1.5 \times 10^{-5}\,m] \times [-1.5 \times 10^{-5}\,m, 1.5 \times 10^{-5}\,m] \times [0\,m, 24 \times 10^{-5}\,m]$ so that the liquid column is lengthened and will break up into



(a) 2D (b) 3D

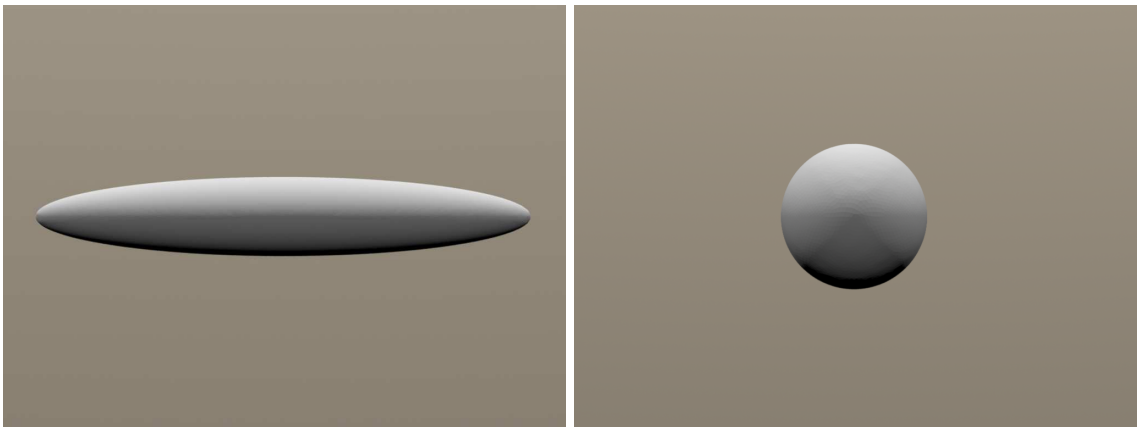
Figure 38: Contact angle boundary condition. The fluid surface at the contact line is virtually extended into the wall (the green line in Subfigure (a) or the green plane in Subfigure (b)) so that the angle between the extended surface and the wall equals the contact angle θ.



(a) $t = .034\,s$ (b) $t = .063\,s$ (c) $t = .125\,s$

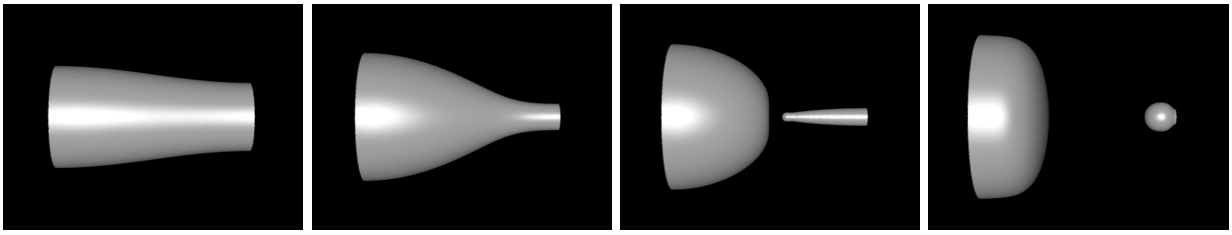
(d) $t = .188\,s$ (e) $t = .313\,s$ (f) $t = .375\,s$

Figure 39: The free surface for the droplet impact test with surface tension.



(a) $t = 0\,s$ (b) $t = .25\,s$

Figure 40: The 3D oscillating ellipsoid test with $\Delta t = .25\,s$. The fully implicit volume conserving scheme on the hybrid particle MAC grid structure is used. Subfigure (a) shows the initial geometry. Subfigure (b) shows the geometry after one time step where the relative volume error is .014%.



(a) $t = 0\,s$ (b) $t = 9.3 \times 10^{-6}\,s$ (c) $t = 1.1 \times 10^{-5}\,s$ (d) $t = 1.2 \times 10^{-5}\,s$

Figure 41: The 3D Rayleigh capillary instability test.

many droplets with various sizes. The same formula for describing the initial interface and the same physical parameters are used. The hybrid particle grid structure and the fully implicit volume conserving scheme from Section 7.5 are used with a grid of resolution $32 \times 32 \times 256$. As seen in Figure 42, the liquid column breaks into five big droplets and four small droplets. From the initial formation of all droplets (Figure 42(c)) to the end of the simulation (Figure 42(d)), each droplet only changes at most .1% of its volume. Note that the five big droplets each contain a separate interior MAC grid region, but our volume conserving scheme only conserves the total volume of all MAC grid regions. This potentially allows big droplets exchange volumes and thus fail to conserve their volumes individually. Whereas it is straightforward to track and correct the volume of each individual MAC grid region, it is not clear how to partition the volume when droplets are pinching due to the complication of remeshing. However, in this example and other complex examples in this section, this problem is negligible since the ratio between the volume error and the total volume of a large bulk of liquid is very small.

Consider a droplet dripping example with a computational domain $[0\,m, .1\,m] \times [0\,m, .15\,m] \times [0\,m, .1\,m]$. The droplet is initially located at the center of the ceiling and has a hemispherical shape with a radius $.02\,m$. The liquid has parameters $\rho = 1000\,kg\,m^{-2}$ and $\sigma = 1.25\,kg\,s^{-2}$. No viscosity is used. The contact angle on the ceiling is set to be 45°, and the contact angle on the ground is set to be 135°. Figure 43 shows the free surface using the hybrid particle grid structure and the fully implicit volume conserving scheme from Section 7.5 with a grid of resolution $64 \times 96 \times 64$. As seen in the figure, the droplet forms a thread of liquid when dripping down from the ceiling, which then breaks up and eventually forms two droplets achieving their equilibrium states on the ceiling and on the ground respectively with the correct contact angles.

We modified the droplet impact test in three spatial dimensions from Section 6. The computational domain is $[0\,m, .3\,m] \times [0\,m, .2\,m] \times [0\,m, .3\,m]$ and the free surface of a pool of liquid is located at $y = .05\,m$. A spherical liquid droplet of radius $r = .012\,m$ is centered at $[.15\,m, .0624\,m, .15\,m]$ with an initial velocity $[0\,m\,s^{-1}, -10\,m\,s^{-1}, 0\,m\,s^{-1}]$. The liquid has parameters $\rho = 1000\,kg\,m^{-3}$ and $\sigma = .0728\,kg\,s^{-2}$. The fully implicit volume conserving scheme from Section 7.5 is used with a grid of resolution $150 \times 100 \times 150$. Figures 44 and 45 show the results.

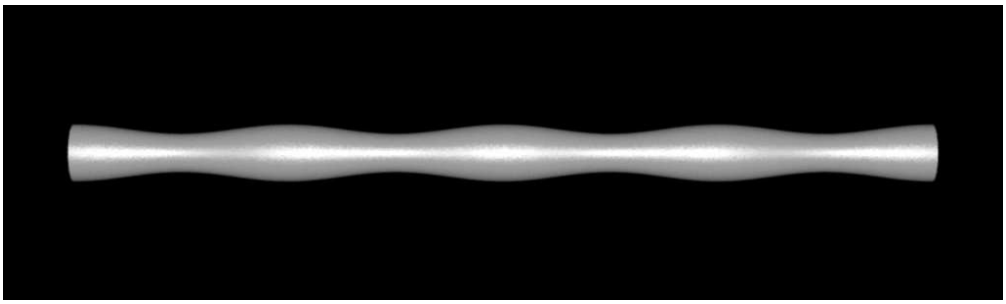
Remark: Note that the update $\Delta\hat{\mathbf{u}}$ is subject to a restriction limiting the sizes of nonlinear iterations. However, this restriction is only $O(\Delta x)$ which is less strict than the $O(\Delta x^{3/2})$ restriction due to surface tension. Moreover, unlike the oscillating ellipsoid example as shown in Figure 40 where the high-curvature features of the droplet need to span a considerable number of grid cells before converging to a steady state solution, in many real world problems of interest such as that shown in Figures 44 and 45 the highest curvature features typically have sizes on the order of a grid cell, and as a result only a couple of nonlinear iterations are actually needed.

8. Summary

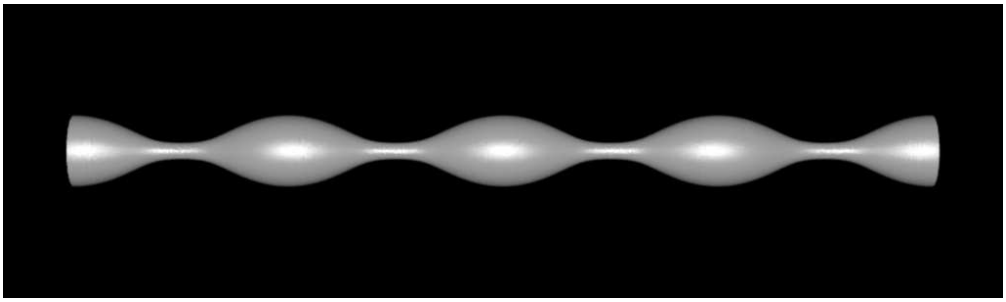
We designed a new spatial discretization for simulating the incompressible Navier-Stokes equations with a free surface by hybridizing a Lagrangian mesh with the MAC grid so that the details near the interface are well resolved by particles while the majority of the interior fluid region is efficiently handled by the MAC grid. In order to accomplish this, we developed a meshing algorithm that not only correctly captures the fluid geometry but also conforms to the MAC grid in the interior. Since meshing is limited to a narrow band near the fluid interface, it only takes a small portion of the total computational time especially with large grid resolutions.

We developed both first and second order accurate advection schemes on this hybrid structure. Particles are moved forward using the forward Euler method or the second order accurate Runge-Kutta method. The quantities on the MAC grid are advected using the semi-Lagrangian method or the semi-Lagrangian style MacCormack method. In order to enable the semi-Lagrangian back tracing needed by both of these methods, a band of ghost cells are filled with valid data either by interpolating from the Lagrangian mesh or by extrapolating from the MAC grid region inside the fluid.

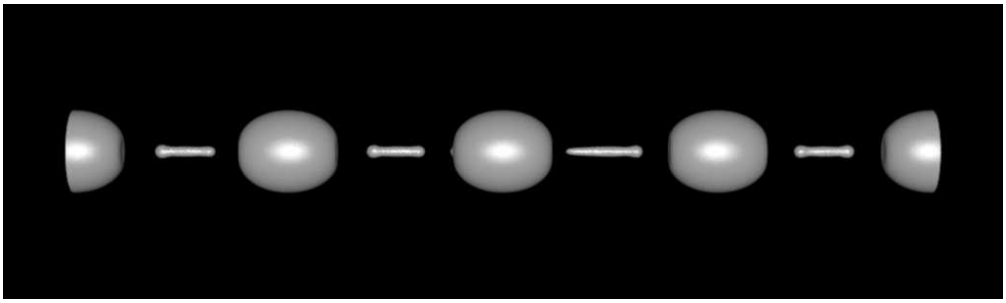
We developed a monolithic second order accurate Poisson equation solver by coupling the Lagrangian mesh with the MAC grid in an orthogonal fashion. The node-based finite volume method from [8, 27] is used



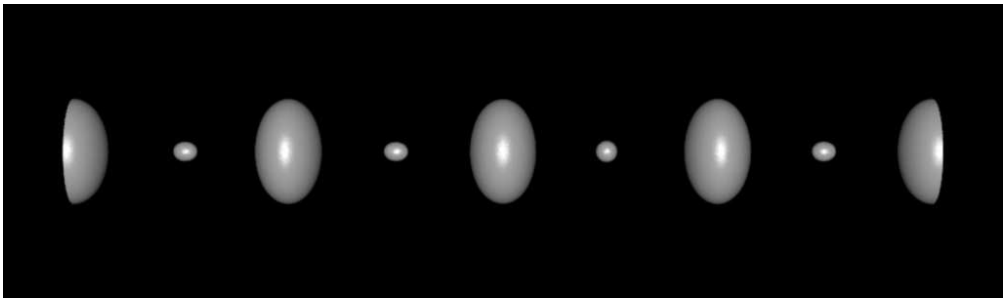
(a) $t = 0\,s$



(b) $t = .95 \times 10^{-5}\,s$

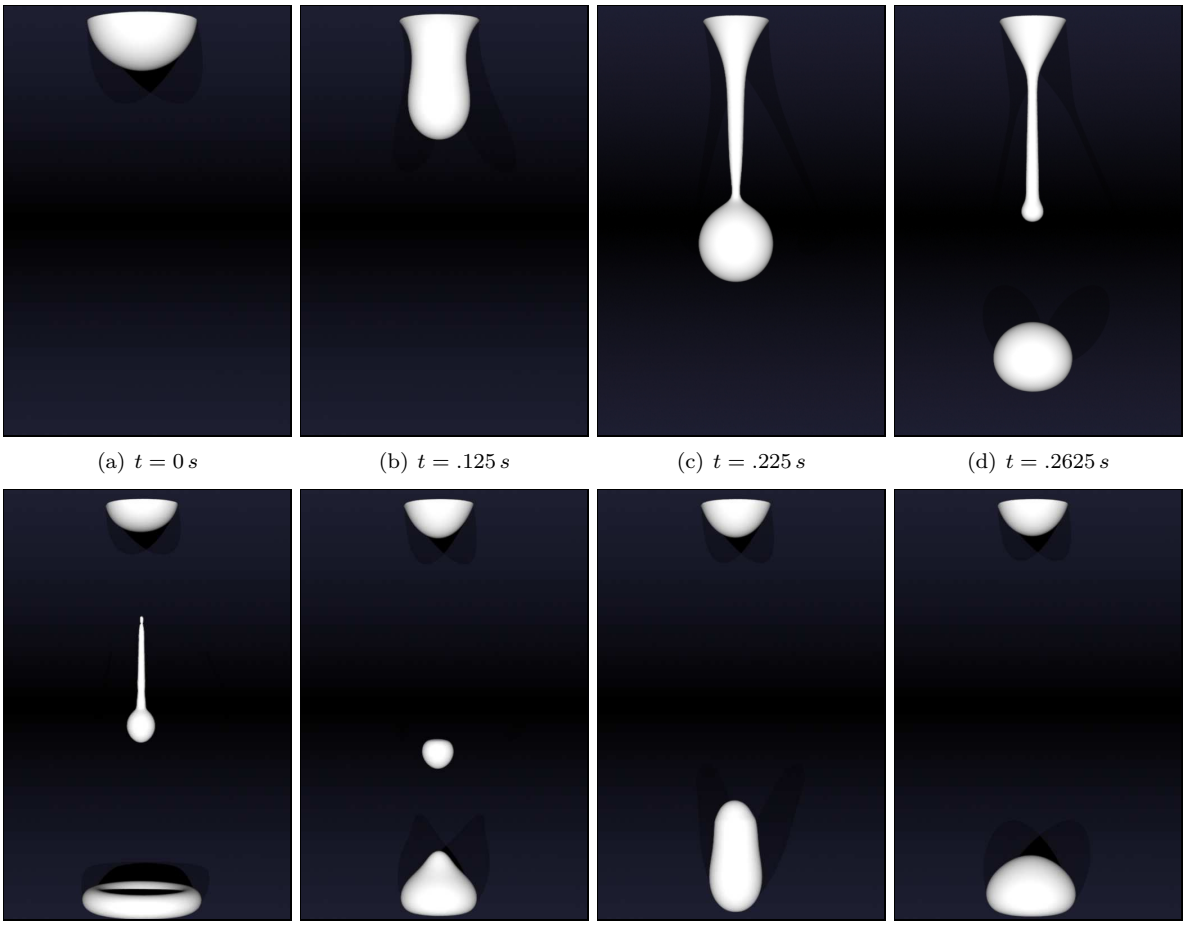


(c) $t = 1.16 \times 10^{-5}\,s$



(d) $t = 1.32 \times 10^{-5}\,s$

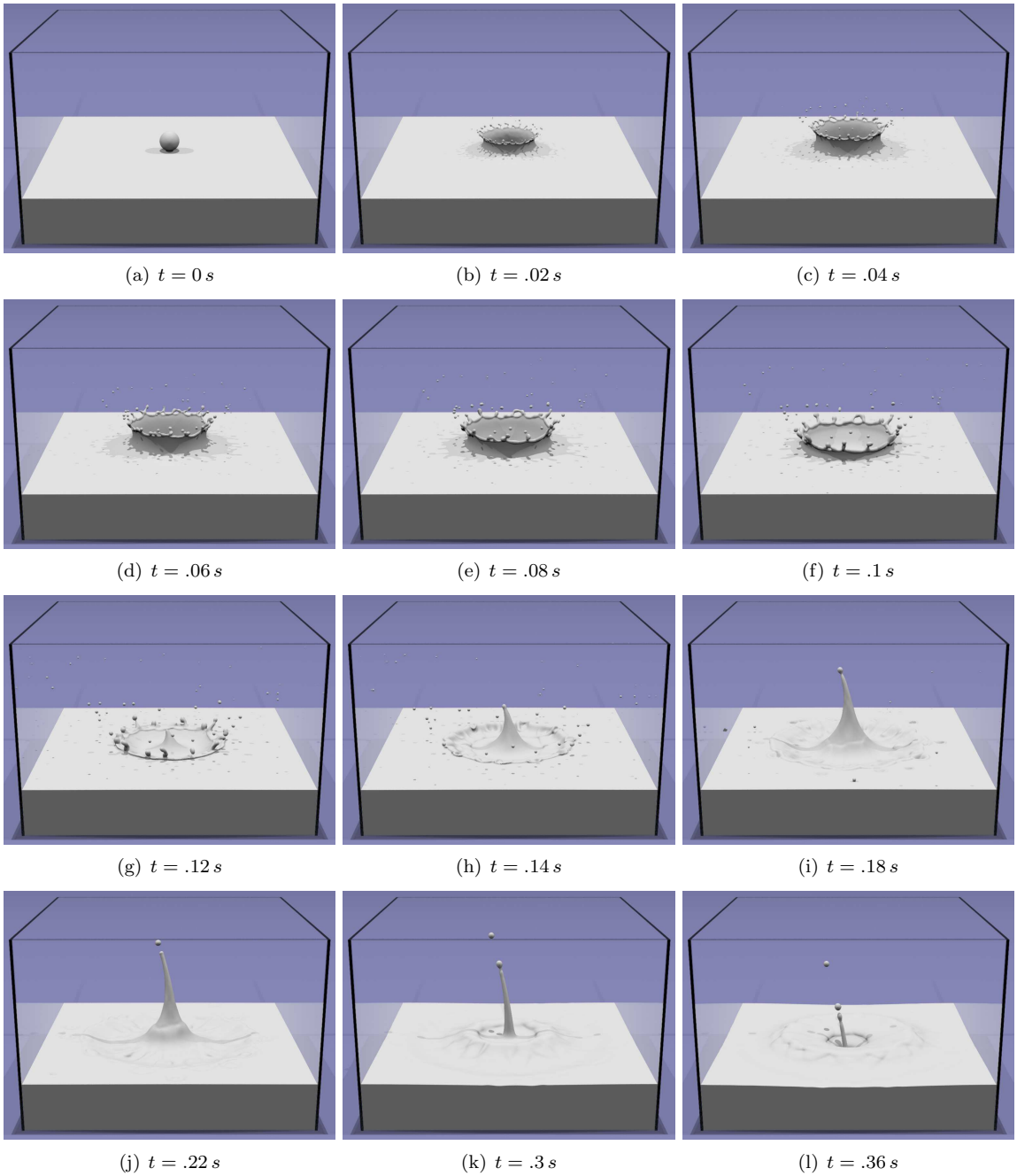
Figure 42: The 3D test of a liquid column breaking up into multiple droplets.



(a) $t = 0\,s$ (b) $t = .125\,s$ (c) $t = .225\,s$ (d) $t = .2625\,s$

(e) $t = .3075\,s$ (f) $t = .325\,s$ (g) $t = .35\,s$ (h) $t = 1\,s$

Figure 43: The 3D droplet dripping test.



(a) $t = 0\,s$ (b) $t = .02\,s$ (c) $t = .04\,s$

(d) $t = .06\,s$ (e) $t = .08\,s$ (f) $t = .1\,s$

(g) $t = .12\,s$ (h) $t = .14\,s$ (i) $t = .18\,s$

(j) $t = .22\,s$ (k) $t = .3\,s$ (l) $t = .36\,s$

Figure 44: The 3D droplet impact test with surface tension.

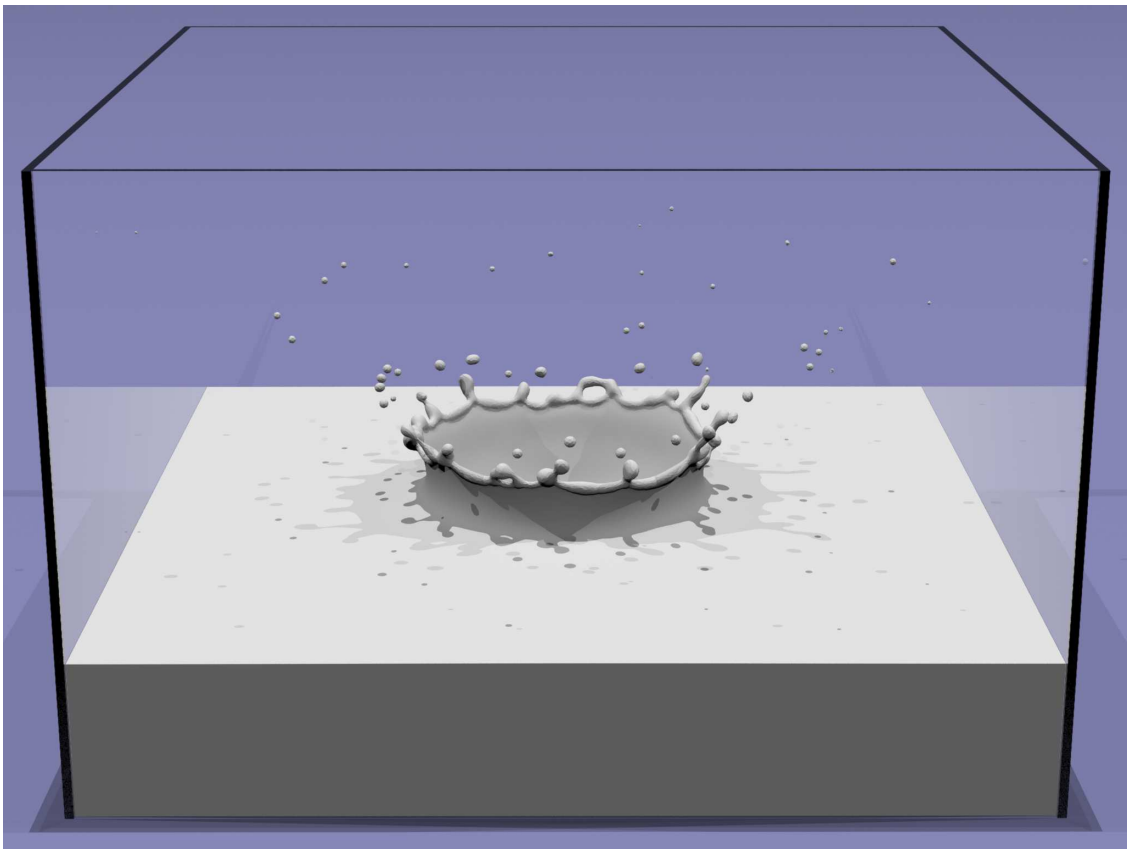


Figure 45: The 3D droplet impact test with surface tension at time $t = .08\,s$.

on the Lagrangian mesh, while the typical MAC grid discretization is used on the MAC grid. We then proved that the Lagrangian mesh discretization is orthogonal, and based on the findings we devised a novel method to couple the Lagrangian mesh discretization and the MAC grid discretization in an orthogonal fashion. Numerical tests in both two and three spatial dimensions demonstrate that this coupled discretization is second order accurate. We also extended this Poisson equation solver to solve the heat equation on cell centers and modified it to solve for viscous forces on staggered MAC grid faces.

With the above building blocks, we used a splitting method [12] to solve the Navier-Stokes equations without surface tension. The droplet impact test and the dam break test in both two and three spatial dimensions were used to demonstrate the ability of our method to resolve details in complex free surface flows.

We also explored stable methods for simulating surface tension using a step-by-step constructive approach on a fully Lagrangian representation. Our first implementation was a semi-implicit scheme where both the normal-directional and all-directional implicit models were tested. The all-directional model showed significant improvement on stability over the normal-directional model, but it suffers from inaccuracy when taking large time steps. To address this issue, we developed a fully implicit scheme by evaluating all matrices at the end of the time step resulting in a nonlinear system. An iterative nonlinear solver is used to solve this nonlinear system where an approximate Jacobian matrix is computed to linearize the system at each iteration. This scheme achieves a steady state solution with large time steps, but it suffers from severe volume error. To solve this problem, we replaced the divergence-free condition with a constant-volume condition resulting in a fully implicit volume conserving scheme that can achieve a steady state solution with large time steps while conserving volume. We then extended this scheme to our hybrid particle grid structure. In addition, we also showed that the same ideas can be applied to the standard front tracking method and the particle level set method. A number of numerical tests were used to examine the ability of our method for surface tension.

9. Limitations and Future Work

There are numerous avenues for future work. In this paper, we have used marker particles throughout the entire fluid region for simplicity, and a further optimization would be to use particles only within a band near the free surface. In Section 2, the Lagrangian mesh is generated against the boundary of the computation domain, but a more efficient solution would allow MAC grid cells to touch the boundary and only mesh the region near the free surface. In Section 4, the coupled Poisson equation system is solved using the conjugate gradient method without a preconditioner. In Section 5, the spatial discretization in three spatial dimensions is not orthogonal when enforcing a Neumann boundary condition on the free surface boundary. Although the numerical results show second order accuracy due to the tendency toward orthogonality under grid refinement, a fully orthogonal discretization would be preferable. In Section 7, the fully implicit schemes are stable but computationally expensive because they require solving a linear system for all degrees of freedom at every iteration. Motivated by the method from [1], we would like to explore simplified models that replace large fluid regions by single degrees of freedom in order to speed up the nonlinear solver. Section 7.5 explores the fully implicit method with the hybrid mesh/grid structure but conserves the volume of the interior MAC grid regions in a global fashion. Whereas it is straightforward to track and correct the volume of each individual MAC grid region, it is not clear how to partition the volume when bulks of liquid are pinching due to the complication of remeshing. A local volume conservation scheme would be preferable for future work. In addition, Section 7.6 explores the fully implicit method with standard front tracking and level set methods but uses global volume conservation. Local volume conservation would also be preferable here.

10. Acknowledgment

Research supported in part by ONR N00014-13-1-0346, ONR N00014-09-1-0101, ONR N-00014-11-1-0027, ONR N00014-11-1-0707, ARL AHPCRC W911NF-07-0027, and the Intel Science and Technology Center for Visual Computing. Computing resources were provided in part by ONR N00014-05-1-0479.

Bibliography

[1] Mridul Aanjaneya, Saket Patkar, and Ronald Fedkiw. A monolithic mass tracking formulation for bubbles in incompressible flow. Journal of Computational Physics, 247:17–61, 2013.

[2] D. Adalsteinsson and J. Sethian. The fast construction of extension velocities in level set methods. J. Comput. Phys., 148:2–22, 1999.

[3] Guy Albertelli and Roger A. Crawfis. Efficient subdivision of finite-element datasets into consistent tetrahedra. In Proceedings of the 8th conference on Visualization '97, VIS '97, pages 213–219, Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[4] N. Amenta and M. Bern. Surface reconstruction by voronoi filtering. Discrete and Computational Geometry, 22:481–504, 1999.

[5] Tariq D. Aslam. A partial differential equation approach to multidimensional extrapolation. Journal of Computational Physics, 193(1):349 – 355, 2004.

[6] Markus Berndt, JéRôMe Breil, StéPhane Galera, Milan Kucharik, Pierre-Henri Maire, and Mikhail Shashkov. Two-step hybrid conservative remapping for multimaterial arbitrary Lagrangian-Eulerian methods. J. Comput. Phys., 230(17):6664–6687, July 2011.

[7] A.I. Bobenko and B.A. Springborn. A discrete laplace-beltrami operator for simplicial surfaces. Discrete Comput Geom, 38:740–756, 2007.

[8] J. Bonet and A. Burton. A simple average nodal pressure tetrahedral element for incompressible and nearly incompressible dynamic explicit applications. Comm. Num. Meth. Eng., 14:437–449, 1998.

[9] A. Bowyer. Computing dirichlet tessellations. The Computer Journal, 24(2):162–166, 1981.

[10] J. U. Brackbill and H. M. Ruppel. Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. J. Comput. Phys., 65:314–343, 1986.

[11] D. Chopp. Some improvements of the fast marching method. SIAM J. Sci. Comput., 223:230–244, 2001.

[12] A.J. Chorin. A numerical method for solving incompressible viscous flow problems. J. Comput. Phys., 2(1):12–26, 1967.

[13] F. Colin, R. Egli, and F. Lin. Computing a null divergence velocity field using smoothed particle hydrodynamics. J. Comput. Phys., 217:680–692, 2006.

[14] S. Cummins and M. Rudman. An SPH projection method. J. Comput. Phys., 152(2):584–607, 1999.

[15] Olivier Devillers and Monique Teillaud. Perturbations for delaunay and weighted delaunay 3d triangulations. Computational Geometry, 44(3):160 – 168, 2011.

[16] T.K. Dey. Curve and Surface Reconstruction: Algorithms with Mathematical Analysis. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2011.

[17] T. Dupont and Y. Liu. Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function. J. Comput. Phys., 190/1:311–324, 2003.

[18] T. Dupont and Y. Liu. Back and forth error compensation and correction methods for semi-Lagrangian schemes with application to level set interface computations. Math. Comp., 76(258):647–668, 2007.

[19] Essex Edwards and Robert Bridson. A high-order accurate particle-in-cell method. International Journal for Numerical Methods in Engineering, 90(9):1073–1088, 2012.

[20] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. J. Comput. Phys., 183:83–116, 2002.

[21] Peter Fleischmann and Siegfried Selberherr. Three-dimensional delaunay mesh generation using a modified advancing front approach. In In Proceedings of the 6th International Meshing Roundtable, pages 267–278. John Wiley & Sons, 1997.

[22] F. Gibou, R. Fedkiw, L.-T. Cheng, and M. Kang. A second-order-accurate symmetric discretization of the Poisson equation on irregular domains. J. Comput. Phys., 176:205–227, 2002.

[23] F. H. Harlow. The particle-in-cell computing method for fluid dynamics. Methods in Computational Physics, 3:319, 1963.

[24] JI Hochstein and TL Williams. An implicit surface tension model. In AIAA, Aerospace Sciences Meeting and Exhibit, 34 th, Reno, NV, 1996.

[25] X.Y. Hu and N.A. Adams. An incompressible multi-phase sph method. J. of Comput. Phys., 227:264–278, 2007.

[26] S. Hysing. A new implicit surface tension implementation for interfacial flows. International Journal for Numerical Methods in Fluids, 51(6):659–672, 2006.

[27] G. Irving, C. Schroeder, and R. Fedkiw. Volume conserving finite element simulations of deformable models. ACM Trans. Graph. (SIGGRAPH Proc.), 26(3):13.1–13.6, 2007.

[28] Imre Jánosi, Dominique Jan, K.Gábor Szabó, and Tamás Tél. Turbulent drag reduction in dam-break flows. Experiments in Fluids, 37(2):219–229, 2004.

[29] G.-S. Jiang and D. Peng. Weighted ENO schemes for Hamilton-Jacobi equations. SIAM J. Sci. Comput., 21:2126–2143, 2000.

[30] M. Kang, R. Fedkiw, and X.-D. Liu. A boundary condition capturing method for multiphase incompressible flow. J. Sci. Comput., 15:323–360, 2000.

[31] B.-M. Kim, Y. Liu, I. Llamas, and J. Rossignac. Flowfixer: Using BFECC for fluid simulation. In Eurographics Workshop on Natural Phenomena 2005, 2005.

[32] B.-M. Kim, Y. Liu, I. Llamas, and J. Rossignac. Advections with significantly reduced dissipation and diffusion. IEEE Trans. on Vis. and Comput. Graph., 13(1):135–144, 2007.

[33] H. Lamb. Hydrodynamics. Cambridge Univ Pr, 1997.

[34] E.-S. Lee, C. Moulinec, R. Xu, D. Violeau, D. Laurence, and P. Stansby. Comparisons of weakly compressible and truly incompressible algorithms for the SPH mesh free particle method. Journal of Computational Physics, 227(18):8417 – 8436, 2008.

[35] P. Liovic, M. Francois, M. Rudman, and R. Manasseh. Efficient simulation of surface tension-dominated flows through enhanced interface geometry interrogation. J. of Comput. Phys., 2010.

[36] J. Liu, S. Koshizuka, and Y. Oka. A hybrid particle-mesh method for viscous, incompressible, multiphase flows. J. Comput. Phys., 202(1):65–93, 2005.

[37] F. Losasso, R. Fedkiw, and S. Osher. Spatially adaptive techniques for level set methods and incompressible flow. Computers and Fluids, 35:995–1010, 2006.

[38] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw. Two-way coupled SPH and particle level set fluid simulation. IEEE TVCG, 14(4):797–804, 2008.

[39] S. Osher and C.-W. Shu. High order essentially non-oscillatory schemes for Hamilton-Jacobi equations. SIAM J. Num. Anal., 28:902–921, 1991.

[40] J. Pozorski and A. Wawrenczuk. SPH computation of incompressible viscous flows. Journal of Theoretical and Applied Mechanics, 40:917–938, 2002.

[41] M. Raessi, M. Bussmann, and J. Mostaghimi. A semi-implicit finite volume implementation of the CSF method for treating surface tension in interfacial flows. International Journal for Numerical Methods in Fluids, 59(10):1093–1110, 2009.

[42] A. Robinson-Mosher, C. Schroeder, and R. Fedkiw. A symmetric positive definite formulation for monolithic fluid structure interaction. J. Comput. Phys., 230(4):1547–1566, 2011.

[43] C. Schroeder, W. Zheng, and R. Fedkiw. Implicit surface tension formulation with a Lagrangian surface mesh on an Eulerian simulation grid. J. Comput. Phys., 231:2092–2115, 2012.

[44] A. Selle, R. Fedkiw, B. Kim, Y. Liu, and J. Rossignac. An Unconditionally Stable MacCormack Method. J. Sci. Comp., 35(2):350–371, 2008.

[45] J. Sethian. Fast marching methods. SIAM Review, 41:199–235, 1999.

[46] Jonathan Richard Shewchuk. Robust Adaptive Floating-Point Geometric Predicates. In Proceedings of the Twelfth Annual Symposium on Computational Geometry, pages 141–150. Association for Computing Machinery, May 1996.

[47] Jonathan Richard Shewchuk. General-dimensional constrained delaunay and constrained regular triangulations, I: Combinatorial properties. Discrete Comput. Geom., 39(1):580–637, March 2008.

[48] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock capturing schemes. J. Comput. Phys., 77:439–471, 1988.

[49] P. Smereka. Semi-implicit level set methods for curvature and surface diffusion motion. J. Sci. Comput., 19(1):439–456, 2003.

[50] P. K. Stansby, A. Chegini, and T. C. D. Barnes. The initial stages of dam-break flow. Journal of Fluid Mechanics, 374:407–424, 10 1998.

[51] M. Sussman. A method for overcoming the surface tension time step constraint in multiphase flows II. International Journal for Numerical Methods in Fluids, 56(6), 2006.

[52] M. Sussman and E. Fatemi. An efficient interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow. SIAM J. Scientific Comput., 20:1165–1191, 1999.

[53] M. Sussman, E. Fatemi, P. Smereka, and S. Osher. An improved level set method for incompressible two-phase flows. Computers and Fluids, 27:663–680, 1998.

[54] M. Sussman and M. Ohta. A stable and efficient method for treating surface tension in incompressible two-phase flow. J. Sci. Comput, 31(4):2447–2471, 2009.

[55] K. Szwec, J. Pozorski, and J.-P. Minier. Analysis of the incompressibility constraint in the smoothed particle hydrodynamics method. International Journal for Numerical Methods in Engineering, 92(4):343–369, June 2012.

[56] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y. J. Jan. A front-tracking method for the computations of multiphase flow. J. Comput. Phys., 169:708–759, 2001.

[57] J. Tsitsiklis. Efficient algorithms for globally optimal trajectories. In Proc. 33rd Conf. on Decision and Control, pages 1368–1373, 1994.

[58] J. Tsitsiklis. Efficient algorithms for globally optimal trajectories. IEEE Trans. on Automatic Control, 40:1528–1538, 1995.

[59] S. O. Unverdi and G. Tryggvason. A front-tracking method for viscous, incompressible, multifluid flows. J. Comput. Phys., 100:25–37, 1992.

[60] D. F. Watson. Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. The Computer Journal, 24(2):167–172, 1981.

[61] J.J. Xu and H.K. Zhao. An Eulerian formulation for solving partial differential equations along a moving interface. J. of Sci. Comput., 19(1):573–594, 2003.

[62] Rui Xu, Peter Stansby, and Dominique Laurence. Accuracy and stability in incompressible sph (isph) based on the projection method and a new approach. J. Comput. Phys., 228(18):6703–6725, October 2009.

[63] S.T. Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. J. of Comput. Phys., 31(3):335–362, 1979.

[64] W. Zheng, J.-H. Yong, and J.-C. Paul. Simulation of bubbles. In SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 325–333, 2006.