Multifluid Algorithm Specification

Dan Martin and Phil Colella
Applied Numerical Algorithms Group

July 8, 2003

Contents

1 Overview **3**

2 Incompressible Flow **4**
2.1 Projections in a Multifluid Environment 5
2.2 Pressure Jump Formulation 7

3 Discretization for Incompressible Flow **8**
3.1 Basic Volume of Fluid Spatial Discretization 8
3.2 Fixed-boundary Formulation for Moving Interfaces 9
3.3 Solving Parabolic Equations with Moving Interfaces 13
3.4 Projection Method Time Discretization 14

4 Compressible Flow **15**
4.1 Velocity Field Decomposition 16
4.2 Discretization of Update 18

5 Spatial Discretizations **21**
5.1 Computing Interface Values 21
5.2 MAC Operators 23
5.3 Cell-Centered Projection Operators 25

6 Specific Algorithm Details and Notes **26**
6.1 Nonlinear Advection 26
6.2 Potential Advective Term 27

6.3 Viscous Terms 27
6.4 Pressure Update 28
6.5 Projections 28

7 AMR for Incompressible Multifluid Systems 29

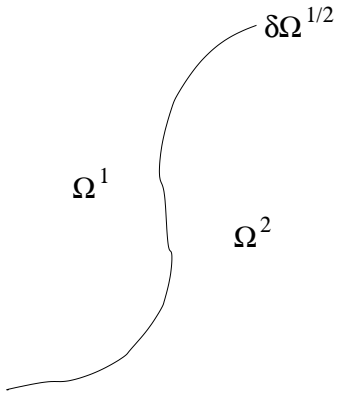

Figure 1: Multifluid system

1 Overview

We would like to solve the Navier-Stokes equations (1 - 4) for a multifluid system on the domain Ω. (Note that (3), (4), and 5) are merely different ways of writing the energy equation; (4) will be useful when evaluating the last term in (3), while (5) is a conservative form of the energy equation.)

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\vec{u}\rho) = 0 \tag{1}$$

$$\frac{\partial \vec{u}}{\partial at} + (\vec{u} \cdot \nabla)\vec{u} = -\frac{1}{\rho}\nabla p + \frac{1}{\rho}\nabla \cdot \boldsymbol{\tau} \tag{2}$$

$$\rho c_p \frac{DT}{Dt} = \nabla \cdot \lambda \ \nabla T + \boldsymbol{D} :: \boldsymbol{\tau} + \frac{Dp}{Dt} \tag{3}$$

$$\frac{Dp}{Dt} = -\rho c^2 (\nabla \cdot \vec{u}) + (\gamma - 1)\Big[\nabla \cdot \lambda \nabla T + \boldsymbol{D} :: \boldsymbol{\tau}\Big], \tag{4}$$

$$\frac{\partial e}{\partial t} + \nabla \cdot (\rho \vec{u} h - \lambda \nabla T) - \boldsymbol{D} :: \boldsymbol{\tau} = 0 \tag{5}$$

where ρ is the fluid density, $\vec{u}$ is the fluid velocity, p is the pressure, $\boldsymbol{\tau}$ is the stress tensor, T is the temperature, c_p is the specific heat, λ is the thermal conductivity, c is the sound speed, and $\boldsymbol{D} = def(\vec{u})$ is the deformation tensor: $D_{ij} = \frac{1}{2}(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i})$. γ is the ratio of specific heats. In (5), e is the internal energy, while h is the enthalpy. For a Newtonian fluid, $\boldsymbol{\tau} = \mu \boldsymbol{D}$. Solid wall

physical boundary conditions for the solution at $\partial\Omega$ are as follows:

$$\begin{aligned} \vec{u} &= 0 \\ \frac{\partial T}{\partial n} &= 0 \end{aligned}$$

For simplicity, we limit the discussion to the case where there are two phases ($\alpha = \{1, 2\}$). We will model the multifluid system by decomposing the domain into two parts, $\Omega = \Omega^{(1)} + \Omega^{(2)}$, as depicted in Figure 1. The region in the domain occupied by phase 1 at time t is denoted by $\Omega^{(1)}(t)$, while the region occupied by phase 2 is $\Omega^{(2)}(t)$. The interface between the two fluids is denoted by $\partial\Omega^{1/2}(t)$. We can think of a multifluid velocity field $\vec{u} = (\vec{u}^{(1)}, \vec{u}^{(2)})$; since the domains $\Omega^{(1)}$ and $\Omega^{(2)}$ are disjoint, this is unique.

The interface is assumed to move with the local fluid velocity:

$$\frac{\partial \vec{x}}{\partial t} = (\vec{u} \cdot \hat{n})\hat{n} \quad \text{on} \quad \partial\Omega^{1/2},$$

which leads to the condition that the fluid velocity normal to the interface is the same on both sides of the interface.

The different phases are linked by a set of jump conditions at the multifluid interface (6)-(10). The notation $[a]$ denotes the jump in a across $\partial\Omega^{1/2}$: $[a] = a^{(2)} - a^{(1)}$ at $\partial\Omega^{1/2}$.

$$[\vec{u}] = 0 \tag{6}$$

$$\left[\hat{t} \cdot \boldsymbol{\tau} \cdot \hat{n}\right] = 0 \tag{7}$$

$$\left[p\right] = \left[\boldsymbol{\tau}_{\hat{n}\hat{n}}\right] + \kappa\sigma \tag{8}$$

$$\left[T\right] = 0 \tag{9}$$

$$\left[\lambda \frac{\partial T}{\partial n}\right] = 0, \tag{10}$$

where $\kappa\sigma$ is the surface tension.

2 Incompressible Flow

For incompressible flow, the conservation of mass equation (1) reduces to the constraint that the velocity field be divergence-free ($\nabla \cdot \vec{u} = 0$). We first outline our approach for incompressible flow before generalizing to the case of compressible flow. For clarity, we denote the incompressible velocity field

by the subscript d (for "divergence-free"). In the incompressible case, and in the absence of heat transfer (T =constant), the equations (1-2) reduce to:

$$\frac{\partial \vec{u}_d}{\partial t} + (\vec{u}_d \cdot \nabla)\vec{u}_d = -\frac{1}{\rho}\nabla p + \frac{1}{\rho}\nabla \cdot \boldsymbol{\tau} \tag{11}$$

$$\nabla \cdot \vec{u}_d = 0 \tag{12}$$

$$\left[\boldsymbol{D}_{\hat{n}\hat{n}}\right] = 0. \tag{13}$$

Note that along with the jump conditions (6-8) the incompressibility constraint adds another jump relation (13) at the multifluid interface [6].

2.1 Projections in a Multifluid Environment

We will use the Hodge projection extensively in our approach, so a description of what this means in a multifluid environment is in order. The Hodge decomposition uniquely decomposes any vector field into a divergence-free part and the gradient of a scalar potential:

$$\begin{array}{lll} \vec{v} = & \vec{v}_d & + \nabla\phi \\ & \nabla \cdot \vec{v}_d = 0 & \Delta\phi = \nabla \cdot \vec{v} \\ & \vec{v}_d \cdot \hat{n} = 0 & \dfrac{\partial \phi}{\partial n} = \vec{v} \cdot \hat{n} \quad \text{on} \quad \partial\Omega. \end{array}$$

The Hodge decomposition is an orthogonal decomposition. Since $(\vec{u}_d \cdot \hat{n}) = 0$ on $\partial\Omega$,

$$\begin{aligned} \int_\Omega \vec{u_d}\ \nabla\phi\ dV &= -\int_\Omega \nabla \cdot \vec{u}_d\ \phi\ dV + \int_{\partial\Omega} (\vec{u}_d\phi) \cdot \hat{n}\ dA \\ &= 0. \end{aligned}$$

We use the Hodge decomposition to define the Hodge projection $\mathbb{P}_0$, which when applied to a vector field $\vec{v}$ returns only the divergence-free portion $\vec{v}_d$. Its complement, $\mathbb{Q}_0$, returns the gradient piece.

$$\begin{aligned} \mathbb{P}_0(\vec{v}) &= \vec{v}_d \\ \mathbb{Q}_0(\vec{v}) &= \nabla\phi. \end{aligned}$$

Symbolically the projection operator and its complement are:

$$\begin{aligned} \mathbb{P}_0 &= \left(\vec{I} - grad(\Delta)^{-1} div\right) \\ \mathbb{Q}_0 &= \left(grad(\Delta)^{-1} div\right). \end{aligned}$$

We will define discrete approximations to the continuous operators for divergence D, gradient G, and Laplacian L. Details of these discrete operators will be presented later (in Section 5).

In a multifluid environment, the projection is applied to each phase simultaneously, with the projection operators linked through boundary conditions at the multifluid interface.

$$\begin{aligned} \vec{v} = \quad & \vec{v}_d && + \nabla\phi \\ & \nabla \cdot \vec{v}_d = 0 && \Delta\phi = \nabla \cdot \vec{v} \\ & \left[\vec{v}_d \cdot \hat{n}\right] = 0 && \left[\frac{\partial \phi}{\partial n}\right] = \left[\vec{v} \cdot \hat{n}\right] \quad \text{on} \quad \partial\Omega^{1/2} \\ & && \left[\phi\right] = 0 \quad \text{on} \quad \partial\Omega^{1/2}. \end{aligned}$$

To compute the potential ϕ, we solve the following equation with the accompanying matching conditions at $\partial\Omega^{1/2}$:

$$L\phi = \nabla \cdot \vec{u} \quad \text{on} \quad \Omega^{(1)} \cup \Omega^{(2)}$$

$$\left[\phi\right] = 0 \tag{14}$$

$$\left[\frac{\partial \phi}{\partial n}\right] = \left[\vec{u} \cdot \hat{n}\right]. \tag{15}$$

The jump conditions (14 - 15) are sufficient for the orthogonality of the projection:

$$\int_{\Omega^1 \cup \Omega^2} \vec{u}_d \; grad(\phi) \; dV = -\int_{\Omega^1 \cup \Omega^2} \nabla \cdot (\vec{u}_d)\phi \; dV + \int_{\partial\Omega^{1/2}} \left[\vec{u}_d \phi\right] \cdot \hat{n} \; dA + ... \tag{16}$$

where the "..." represents the boundary terms along the physical boundary, which vanish if $\vec{u}_d \cdot \hat{n} = 0$ on the physical boundary. The boundary conditions (14-15) guarantee that the second term of the right-hand-side in (16) vanishes. The jump condition for the gradient of ϕ (15) will also ensure that $\left[\vec{u}_d \cdot \hat{n}\right]$ vanishes ($\left[\mathbb{P}(\vec{u})\right] = 0$).

The correction is then applied to each phase:

$$\vec{u}_d = \vec{u} - \nabla\phi.$$

The variable density Hodge decomposition leads to the variable density projection $\mathbb{P}_\rho$:

$$\mathbb{P}_\rho(\vec{u}) = \vec{u}_d \tag{17}$$

$$\mathbb{Q}(\vec{u}) = \frac{1}{\rho}\nabla\phi. \tag{18}$$

In the constant density case, if $\rho = 1$, then $\mathbb{P}_0 = \mathbb{P}_\rho$. Symbolically,

$$\mathbb{P}_\rho(\vec{v}) = \big(I - \frac{1}{\rho} grad(L_\rho)^{-1} div\big)\vec{v}$$

$$\mathbb{Q}_\rho(\vec{v}) = \frac{1}{\rho}\big(grad(L_\rho)^{-1} div\big)\vec{v}$$

$$L_\rho = div\Big(\frac{1}{\rho} grad\Big).$$

The variable-density projection is applied in the same way as the constant-coefficient projection, using the same matching conditions at $\partial\Omega^{1/2}$:

$$\begin{aligned} \vec{v} \quad = \quad \vec{v}_d \qquad\qquad & + \frac{1}{\rho}\nabla\phi \\ \nabla \cdot \vec{v}_d = 0 \qquad\qquad & \nabla \cdot \frac{1}{\rho}\nabla\phi = \nabla \cdot \vec{v} \\ \vec{v}_d \cdot \hat{n} = 0 \qquad\qquad & \frac{1}{\rho}\frac{\partial\phi}{\partial n} = \vec{v} \cdot \hat{n} \ \text{ on } \ \partial\Omega \\ \big[\vec{v}_d \cdot \hat{n}\big] = 0 \qquad\qquad & \Big[\frac{1}{\rho}\frac{\partial\phi}{\partial n}\Big] = \big[\vec{v} \cdot \hat{n}\big] \\ & \big[\phi\big] = 0. \end{aligned}$$

2.2 Pressure Jump Formulation

In a single incompressible fluid, the pressure gradient (∇p) is the gradient component returned by a projection, and is kinematically defined by the divergence constraint. In the presence of a multifluid interface with surface tension, there is a jump in the pressure balancing the surface tension and any difference in normal stresses at the interface (8). We decompose the pressure into two components:

$$p = \pi + \chi$$

where π is the pressure field required to enforce the divergence constraint (i.e. the potential returned by the projection) while χ contains the pressure field induced by the jump in the pressure at $\partial\Omega^{1/2}$:

$$\begin{aligned} \nabla \cdot \frac{1}{\rho}\nabla\chi = 0 \qquad\qquad & (19) \\ \big[\chi\big] = \big[p\big] & \\ = \big[\boldsymbol{\tau}_{\hat{n}\hat{n}}\big] - \kappa\sigma & \end{aligned}$$

3 Discretization for Incompressible Flow

3.1 Basic Volume of Fluid Spatial Discretization

The underlying discretization of space is given by rectangular control volumes on a Cartesian grid: $\Upsilon_{\boldsymbol{i}} = [(\boldsymbol{i} - \frac{1}{2}\boldsymbol{u})h, (\boldsymbol{i} + \frac{1}{2}\boldsymbol{u})h]$, $\boldsymbol{i} \in \mathbb{Z}^d$, where d is the dimensionality of the problem, h is the mesh spacing, and $\boldsymbol{u}$ is the vector whose entries are all ones. In the case of a fixed, irregular domain Ω, the geometry is represented by the intersection of Ω with the Cartesian grid (figure 2). We obtain control volumes $V_{\boldsymbol{i}} = \Upsilon_{\boldsymbol{i}} \cap \Omega$, and faces $A_{\boldsymbol{i} \pm \frac{1}{2}\boldsymbol{e}_s}$ that are the intersection of $\partial V_{\boldsymbol{i}}$ with the coordinate planes $\{\boldsymbol{x} : x_s = (i_s \pm \frac{1}{2})h\}$. We also define $A^B_{\boldsymbol{i}}$ to be the intersection of the boundary of the irregular domain with the Cartesian control volume: $A^B_{\boldsymbol{i}} = \partial\Omega \cap \Upsilon_{\boldsymbol{i}}$. We will assume here that there is a one-to-one correspondence between the control volumes and faces and the corresponding geometric entities on the underlying Cartesian grid. The description can be generalized to allow for boundaries whose width is less than the mesh spacing, or sharp trailing edges.

In order to construct finite difference methods, we will need only a small number of real-valued quantities that are derived from these geometric objects.

- The areas / volumes, expressed in dimensionless terms: volume fractions $\kappa_{\boldsymbol{i}} = |V_{\boldsymbol{i}}|h^{-d}$, face apertures $\alpha_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_s} = |A_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_s}|h^{-(d-1)}$ and boundary apertures $\alpha^B_{\boldsymbol{i}} = |A^B_{\boldsymbol{i}}|h^{-(d-1)}$. We assume that we can compute estimates of the dimensionless quantities that are accurate to $O(h^2)$.

- The locations of centroids, and the average outward normal to the boundary.

$$\boldsymbol{x}_{\boldsymbol{i}} = \frac{1}{|V_{\boldsymbol{i}}|}\int_{V_{\boldsymbol{i}}} \boldsymbol{x} dV$$

$$\boldsymbol{x}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_s} = \frac{1}{|A_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_s}|}\int_{A_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}_s}} \boldsymbol{x} dA$$

$$\boldsymbol{x}^B_{\boldsymbol{i}} = \frac{1}{|A^B_{\boldsymbol{i}}|}\int_{A^B_{\boldsymbol{i}}} \boldsymbol{x} dA$$

$$\vec{n}^B_{\boldsymbol{i}} = \frac{1}{|A^B_{\boldsymbol{i}}|}\int_{A^B_{\boldsymbol{i}}} \vec{n}^B dA$$

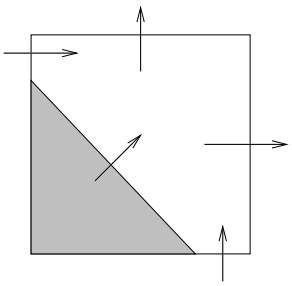

Figure 2: Cartesian grid cell containing volume-of-fluid representation of an irregular boundary. The shaded region indicates the part of the cell not contained in the irregular domain. The arrows indicate the centering of fluxes in a finite volume approximation to the divergence.

where $\vec{n}^B$ is the outward normal to $\partial\Omega$, defined for each point on $\partial\Omega$. Again, we assume that we can compute estimates of these quantities that are accurate to $O(h^2)$.

Using just these quantities, we can define conservative discretizations for the divergence operator. Let $\vec{F} = (F^{(1)} \dots F^d)$ be a function of $\boldsymbol{x}$. Then

$$\begin{aligned} \nabla \cdot \vec{F} &\approx \frac{1}{|V_{\boldsymbol{i}}|} \int_{V_{\boldsymbol{i}}} \nabla \cdot \vec{F} dV = \frac{1}{|V_{\boldsymbol{i}}|} \int_{\partial V_{\boldsymbol{i}}} \vec{F} \cdot \vec{n} dA \qquad (20) \\ &\approx \frac{1}{\kappa_{\boldsymbol{i}} h} \Big(\sum_{\pm=+,-} \sum_{s=1}^{d} \pm \alpha_{\boldsymbol{i} \pm \frac{1}{2}\boldsymbol{e}_s} F^s(\boldsymbol{x}_{\boldsymbol{i} \pm \frac{1}{2}\boldsymbol{e}_s}) + \alpha_{\boldsymbol{i}}^B \vec{n}_{\boldsymbol{i}}^B \cdot \vec{F}(\boldsymbol{x}_{\boldsymbol{i}}^B) \Big) \end{aligned}$$

where (20) is obtained by replacing the integrals of the normal components of the vector field $\vec{F}$ with the values at the centroids.

3.2 Fixed-boundary Formulation for Moving Interfaces

In a single-fluid projection method such as that in [2], different parts of the update are evaluated at different times between t^n and $t^{n+1} = t^n + \Delta t$. In a fixed-geometry problem, this is straightforward. However, with a moving interface ($\partial\Omega^{1/2} = \partial\Omega^{1/2}(t)$) this can present problems, since the operators and domains are changing as a function of time. Based on an approach

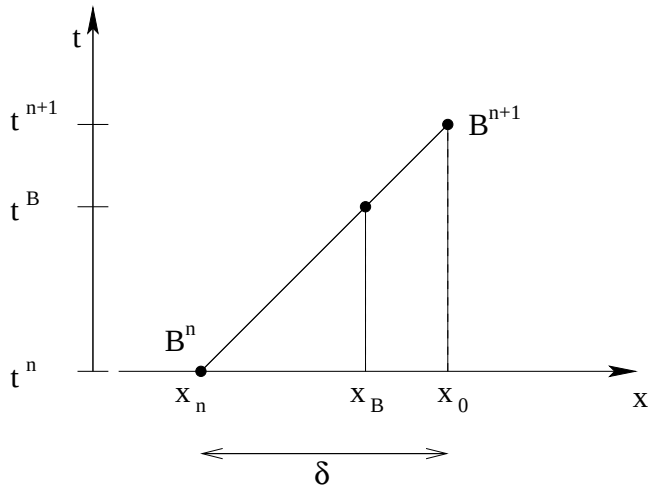

Figure 3: Time-space depiction of 1-D moving interface. Interface moves from location x_n at time t^n to location $x_{n+1} = x_0$ at time t^{n+1}. Approach will be to fix operator discretizations at the final location, and extrapolate data and matching conditions to this location (i.e. extrapolate from x^n and x_B to x_0).

developed by McCorquodale and Colella [8], we will transform the moving-boundary problem into a fixed-boundary problem for the duration of the timestep from t^n to t^{n+1}.

To illustrate our approach to moving interfaces, consider a conservation law with a source term:

$$\frac{\partial q}{\partial t} + \nabla \cdot (\vec{u}q) = \mathcal{S} \quad \text{on} \quad \Omega^{1/2}(t). \tag{21}$$

In a multifluid environment with a moving interface, we would discretize (21) as

$$\kappa^{n+1}\overline{q}^{n+1} = \kappa^n \overline{q}^n = \Delta t \sum_s (\overline{\alpha}_s \vec{u}_s \cdot \hat{n}_s q_s) + \Delta t \overline{\mathcal{S}}, \tag{22}$$

where $\overline{q}$ is the cell average of q, $\overline{\alpha}$ is the time average of the aperture α, and the sum over S is the sum over faces and signs from (20). Note that there is no contribution to the flux on the moving multifluid interface, since the boundary velocity is equal to the normal component of the fluid velocity: $\frac{d\vec{x}_B}{dt} = \vec{u}$.

However, if $\mathcal{S}$ requires an implicit update approach (for example, in a diffusion equation), this is an inconvenient form for the time discretization,

since $\mathcal{S}$ is averaged over a space-time region with a nontrivial temporal variation. Instead, we will represent the solution of (21) on a time-varying domain as the solution to an equivalent problem on a domain which remains fixed during the course of each time step. At the beginning of the time step from t^n to t^{n+1}, we predict the location of the boundary at the end of the timestep using a forward-Euler method based on the velocities at t^n. We then move the boundary and map the t^n solution onto the new domain, extrapolating where necessary to fill an extension to the t^n solution in regions which have been uncovered by the interface motion. We can then advance the solution on the fixed t^{n+1} domain to compute the solution at time t^{n+1}. This leads to a temporal discretization of the following form:

$$\kappa^{n+1}q^{n+1} = \kappa^{n+1}q^n - \frac{\Delta t}{h}\Big(\sum_s \alpha_s^{n+1}(q\vec{u}\cdot\hat{n})_s^{n+\frac{1}{2}} - \alpha_B^{n+1}(q_B\vec{u}_B\cdot\hat{n}_B)^{n+\frac{1}{2}}\Big) + \Delta t\overline{\mathcal{S}}$$

where $\overline{\mathcal{S}}$ is an average over the fixed space-time domain of the timestep. We will need to evaluate jump conditions at the moving boundary. For example, if $\mathcal{S} = \nabla\cdot D\nabla q$ then we may have jump conditions of the form:

$$\big[q\big] = 0 \tag{23}$$

$$\big[D\frac{\partial q}{\partial x}\big] = 0 \ \text{ on } \ \partial\Omega^{1/2}(t). \tag{24}$$

When jump conditions must be evaluated during the advance at times other than t^{n+1}, we compute a modified jump condition by extrapolation, as depicted in Figure 3. If the true jump condition at time t is

$$\big[q\big](\boldsymbol{x}_B, t) = f(\boldsymbol{x}_B, t) \ \text{ on } \ \boldsymbol{x}_{\mathrm{B}} \in \partial\Omega^{1/2}(\mathrm{t}),$$

then we approximate the jump condition at the fixed interface using a Taylor series:

$$\big[q\big](\boldsymbol{x}_B, t) = \big[q\big](\boldsymbol{x}_0, t) + d(\boldsymbol{x}_0, t)\ \hat{n}^{n+1}\cdot\big[(\nabla q^n)\big](\boldsymbol{x}_0, t) + O(h^2) \quad \text{on } \ \boldsymbol{x}_0 \in \partial\Omega^{1/2}(\mathrm{t}^{\mathrm{n+1}}), \tag{25}$$

where $d(\boldsymbol{x}_0, t)$ is the signed distance between x_0 and $\partial\Omega^{1/2}(t)$ and $\hat{n}^{n+1}$ is the normal to the interface $\partial\Omega^{1/2}(t^{n+1})$ at $\boldsymbol{x}_0$. The distance $d(\boldsymbol{x}_0, t)$ is computed by interpolation. Given the position of the multifluid interface at t^n and t^{n+1}, we compute the signed distance d from each of the vertices (black circles in

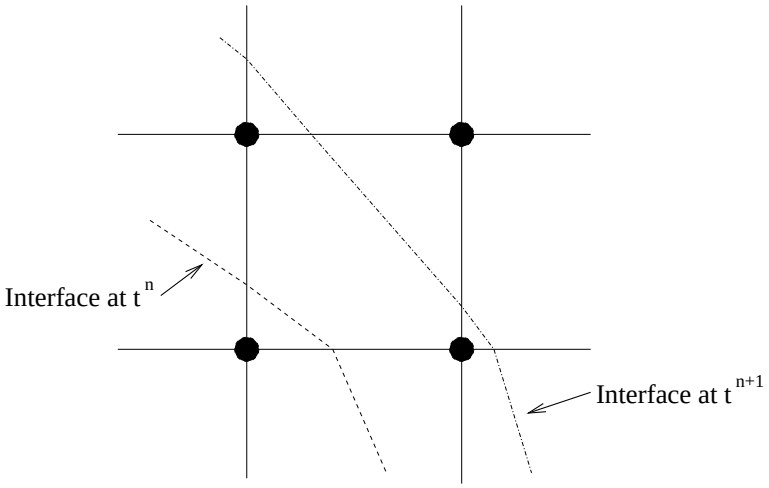

Figure 4: Depiction of multifluid interface moving from time t^n to time t^{n+1}. The interface location may be parameterized by computing the signed distance from the four vertices of the cell (circles in figure) to the plane of the interface at a given time.

Figure 4) of the cell at times t^n and t^{n+1}. Then, we interpolate these values linearly in time at each vertex to compute the signed distances from the vertices to $\partial\Omega(t)$. Since the distance is a smooth function in space, we may then use bilinear interpolation of these time-interpolated values to compute $d(\boldsymbol{x}_0, t)$, the distance between $\boldsymbol{x}_0$ and the interface at time t. Solving (25) for the jump condition at the fixed boundary location (to $O(h^2)$),

$$\begin{aligned} \big[q\big](\boldsymbol{x}_0, t) &= \big[q\big](\boldsymbol{x}_B, t) - d\ \hat{n}^{n+1} \cdot \big[(\nabla q^n)\big](\boldsymbol{x}_0, t) \\ &= f(\boldsymbol{x}_B(\boldsymbol{x}_0, t), t) - d\ \hat{n}^{n+1} \cdot \big[(\nabla q^n)\big](\boldsymbol{x}_0, t), \end{aligned}$$

where $f(\boldsymbol{x}_B(\boldsymbol{x}_0, t), t)$ is f evaluated at the point $\boldsymbol{x}_B(t)$, the actual location of the interface at time t. In the case where $f(\boldsymbol{x}, t) = f(t)$, this is simple, since we don't need to know $\boldsymbol{x}_B$ precisely. In the case where f is a function of $\boldsymbol{x}$, $\boldsymbol{x}_B$ is located by computing the closest point on $\partial\Omega^{1/2}(t)$ to the point $\boldsymbol{x}_0$. Likewise, for the gradient jump relation (24), we use a Taylor series to extrapolate values to the fixed-boundary location.

In the presence of time-dependent geometries, both operator discretizations and variables have time centerings. For example, $div^{n+\frac{1}{2}}$ will denote the divergence centered at time $t^{n+\frac{1}{2}}$, which will use the location of the interface at time $t^{n+\frac{1}{2}}$ in its discretization. Since we are using the artificially

fixed interface approach detailed above, the effect of operator time centering only appears in the Taylor series used to compute the jump relation at the interface (evaluated at the time-centering of the operator). Away from the multifluid interface, the time-centering of the operator has no effect. Note that the time centering of the operator and the data on which it is applied need not be the same – for example, $\nabla^{n+1}\phi^n$ is the gradient operator defined using the interface geometry at time t^{n+1} applied to the variable ϕ centered at time t^n.

3.3 Solving Parabolic Equations with Moving Interfaces

Other projection methods (for example, that in [1]) use the Crank-Nicolson scheme to compute the viscous terms in the projection method. Unfortunately, the Crank-Nicolson scheme is unstable in the presence of moving interfaces [8], so we employ a scheme based on the second-order L_0-stable scheme in [11] to compute a second-order approximation of the viscous terms. This scheme requires two elliptic solves for each update (rather than the single solve required for Crank-Nicolson), but is stable in the presence of moving interfaces. For a heat equation with a source term:

$$\frac{\partial \phi}{\partial t} = L\phi + \mathcal{S}, \tag{26}$$

we can discretize the update from t to $(t + \Delta t)$ as

$$\phi^{n+1} = \phi^n + \Delta t L(\phi^n, \phi^{n+1}) + \Delta t \mathcal{S}^{n+\frac{1}{2}}, \tag{27}$$

where $\mathcal{S}^{n+\frac{1}{2}}$ is a second-order approximation to the source term at the half-time $(t+\frac{1}{2}\Delta t)$. The expression $L(\phi^n, \phi^{n+1})$ denotes the diffusive terms in the update computed using the semi-implicit approach outlined here.

First, compute the diffused source term $\mathcal{S}^*$:

$$\mathcal{S}^* = \Delta t(I + (\frac{1}{2} - a)\Delta t L^{n+\frac{1}{2}})\mathcal{S}.$$

This differs from the algorithm presented in [11] because our source term is centered at the half time $(t + \frac{\Delta t}{2})$, while the source term in the original reference is centered at the times t and t^{n+1}. Then, we compute the intermediate quantity ϕ^e at the intermediate time $t^e = t^n + (1 - r_1)\Delta t$:

$$(I - r_2 \Delta t L^{t^e})\phi^e = \phi^n + (1 - a)\Delta t L^n \phi^n + \vec{\mathcal{S}}^*.$$

Finally, we solve for the approximation to the new-time solution $\phi^{n+1} = \phi(t + \Delta t)$:

$$\big(I - r_1 \Delta t L^{n+1})\big)\phi^{n+1} = \phi^e.$$

The computed value of ϕ^{n+1} may be used directly, or if needed, the diffusive term may be computed:

$$L(\phi^n, \phi^{n+1}) = \frac{\phi^{n+1} - \phi^n}{\Delta t} - \mathcal{S}^{n+\frac{1}{2}}.$$

The quantities a, r_1, and r_2 are the values suggested in [11]:

$$\begin{aligned} a &= 2 - \sqrt{2} - \epsilon, \\ discr &= \sqrt{a^2 - 4a + 2}, \\ r_1 &= \frac{2a - 1}{a + discr}, \\ r_2 &= \frac{2a - 1}{a - discr}, \end{aligned}$$

where ϵ is a small quantity (we use 10^{-8}).

3.4 Projection Method Time Discretization

To discretize the projection method in the presence of a moving multifluid interface, we follow the approach of Trebotich and Colella [10]. Following [2] and [7], we first compute an approximation to the updated velocity field, $\vec{u}^*$:

$$\rho^{n+1} = \rho^n - \Delta t \ \nabla \cdot (\rho \vec{u})^{n+\frac{1}{2}}$$

$$\vec{u}_d^* = \vec{u}_d^n + \Delta t \Big(-\mathcal{A}_d(\vec{u}_d)^{n+\frac{1}{2}} - \frac{1}{\rho^{n+\frac{1}{2}}} \nabla^{n+\frac{1}{2}} (\pi^{n-\frac{1}{2}} + \chi) + \frac{1}{\rho^{n+\frac{1}{2}}} \nabla \cdot \boldsymbol{\tau}(\vec{u}_d^n, \vec{u}_d^*) \Big) \tag{28}$$

$$\rho^{n+\frac{1}{2}} = \frac{1}{2}(\rho^n + \rho^{n+1})$$

$$\big[\vec{u}_d\big] = 0 \tag{29}$$

$$\big[\boldsymbol{D}_{nn}\big] = 0 \tag{30}$$

$$\big[\hat{t} \cdot \boldsymbol{\tau} \cdot \hat{n}\big] = 0, \tag{31}$$

where $\mathcal{A}_d(\vec{u}_d)$ is an approximation to $(\vec{u}_d \cdot \nabla)\vec{u}_d$ centered at the half-time ($t^{n+\frac{1}{2}} = t^n + \frac{\Delta t}{2}$) and computed using a second-order Godunov approach

as described in [7]. The jump relations (29-31) are specified at the actual multifluid location interface, but are applied by shifting them to the fixed-boundary location as specified in Section (3.2). The $\nabla \cdot (\boldsymbol{\tau}(\vec{u}_d^n, \vec{u}_d^*))$ notation indicates that the viscous terms are computed using a semi-implicit method such as that found in [11].

When evaluating the viscous terms in (28) using the algorithm described in section 3.3, we will use the jump relations:

$$\begin{aligned}\left[\vec{u}_d^*\right] &= 0 \\ \left[\boldsymbol{D}_{nn}\right] &= 0 \\ \left[\hat{t} \cdot \boldsymbol{\tau} \cdot \hat{n}\right] &= 0.\end{aligned}$$

Since we need $2Dim$ jump relations to solve the Helmholtz equation in Dim-dimensional space, this is correctly specified (the $\left[\vec{u}_d^*\right]$ condition supplies Dim conditions, the $\left[\boldsymbol{D}_{nn}\right]$ condition supplies 1, and the $\left[\hat{t} \cdot \boldsymbol{\tau} \cdot \hat{n}\right]$ condition supplies $Dim - 1$).

The intermediate velocity field $\vec{u}_d^*$ will not, in general, satisfy the divergence constraint (12). To complete the velocity update, we project:

$$\vec{u}^{n+1} = \mathbb{P}_{\rho^{n+1}}^{n+1}\left(\vec{u}_d^* + \frac{\Delta t}{\rho^{n+1}} \nabla^{n+1} \pi^{n-\frac{1}{2}}\right) \tag{32}$$

$$\pi^{n+\frac{1}{2}} = \frac{1}{\Delta t}(L_{\rho^{n+1}}^{n+1})^{-1}(\nabla \cdot)^{n+1}\left(\vec{u}_d^* + \frac{\Delta t}{\rho^{n+1}} \nabla^{n+1} \pi^{n-\frac{1}{2}}\right) \tag{33}$$

The χ pressure component is computed by solving (19). The main difference between the formulation in (32-33) and that in [10] is that in [10] the approximation to the velocity ($\vec{u}^*$) is projected, solving for the increment to the pressure $\pi^{n+\frac{1}{2}} - \pi^{n-\frac{1}{2}}$, while our approach will be to project $(\vec{u}^* + \frac{1}{\rho}\nabla\pi)$, solving for the pressure field itself ($\pi^{n+\frac{1}{2}}$).

4 Compressible Flow

To generalize beyond incompressible flow, we again take advantage of the Hodge decomposition, which allows us to decompose a general velocity field into a divergence-free part ($\vec{u}_d$) and a potential part ($\vec{u}_p = \nabla\phi$). The effects of the acoustic modes (which can severely restrict the stable timestep in a fully compressible algorithm because of a CFL condition based on the

acoustic speeds) are confined to the potential velocity field $\vec{u}_p$. We can then treat the divergence-free part in the same way as outlined for incompressible flow, while the potential flow part may be advanced implicitly for stability. This approach was used successfully in [3] and [10]. The method outlined for incompressible flow in section 3 may then be generalized for non-incompressible flows while using the same advectively-defined (rather than acoustically-defined) CFL stability condition.

4.1 Velocity Field Decomposition

Following [3] and [10], we split the velocity field into divergence-free and potential-flow parts:

$$\begin{array}{ccccc} \vec{u} & = & \vec{u}_d & + & \vec{u}_p \\ & & \vec{u}_d = \mathbb{P}_0(\vec{u}) & & \vec{u}_p = \mathbb{Q}_0(\vec{u}) \end{array}$$

The momentum equation is split into two parts:

$$\frac{\partial \vec{u}_d}{\partial t} = -\mathcal{A}_d(\vec{u}_d, \vec{u}_p) + \frac{1}{\rho}\nabla \cdot (\mu\ def(\vec{u}_d)) - \frac{1}{\rho}\nabla\pi - \vec{\zeta} + \frac{1}{\rho}\vec{f}_d \qquad (34)$$

$$\frac{\partial \vec{u}_p}{\partial t} = -\nabla\Big(\frac{|\vec{u}_p|^2}{2}\Big) + \frac{1}{\rho}\nabla \cdot (\mu\ def(\vec{u}_p)) - \frac{1}{\rho}\nabla\delta + \vec{\zeta} + \frac{1}{\rho}\vec{f}_p. \qquad (35)$$

The nonlinear advective term for $\vec{u}_d$, $\mathcal{A}_d(\vec{u}_d, \vec{u}_p)$ is:

$$\mathcal{A}_d(\vec{u}_d, \vec{u}_p) = (\vec{u} \cdot \nabla)\vec{u} - \nabla\Big(\frac{|\vec{u}_p|^2}{2}\Big).$$

The $\vec{\zeta}$ term in (34) and (35) represents the generation of vorticity from $\vec{u}_p$:

$$\vec{\zeta} = -\mathbb{P}_0\big(\nabla \cdot (\mu\ def(\vec{u}_p)) - \frac{1}{\rho}\nabla\delta\big).$$

Also, $\vec{f}_d$ and $\vec{f}_p$ are the divergence-free and potential parts of the force:

$$\begin{aligned} \vec{f}_d &= \mathbb{P}_0(\vec{f}) \\ \vec{f}_p &= \mathbb{Q}_0(\vec{f}). \end{aligned}$$

We likewise decompose the pressure into three parts (recall that χ contains the part of the pressure resulting from the pressure jump at the multifluid

interface):

$$
\begin{aligned}
p &= \pi + \delta + \chi \\
\frac{1}{\rho}\nabla\pi &= \mathbb{Q}_\rho\Big(-\mathcal{A}_d(\vec{u}_d, \vec{u}_p) + \frac{1}{\rho}\nabla\cdot(\mu\ def(\vec{u}_d))\Big) \\
\pi &= \big(L_\rho\big)^{(-1)}\nabla\cdot\Big(-\mathcal{A}_d(\vec{u}_d, \vec{u}_p) + \frac{1}{\rho}\nabla\cdot(\mu\ def(\vec{u}_d))\Big) \\
\frac{\partial\delta}{\partial t} + \rho c^2\nabla\cdot\vec{u}_p + \frac{\partial\pi}{\partial t} + \frac{\partial\chi}{\partial t} + \vec{u}\cdot\nabla p &= (\gamma-1)\big(\nabla\cdot\lambda\nabla T + \boldsymbol{D}::\boldsymbol{\tau}\big) \qquad (36)
\end{aligned}
$$

χ is computed using (19). The evolution equation for the temperature is given by (3), with boundary conditions (9-10).

At the free boundary $\partial\Omega^{1/2}$, we have a set of jump relations linking the two phases. In the dissipation-free case ($\mu = 0, \lambda = 0$),

$$
\begin{aligned}
\big[\vec{u}_p\big] &= 0 \\
\big[\vec{u}_d\cdot\hat{n}\big] &= 0 \\
\big[p\big] &= \kappa\sigma.
\end{aligned}
$$

Note that any slip at the inviscid multifluid interface is contained in $\vec{u}_d$, as a result of defining $\vec{u}_p$ as the gradient of a single-valued potential. In the viscous case ($\mu > 0$), the strategy will be for the incompressible component to satisfy viscous boundary conditions. Boundary conditions for $\vec{u}_p$ are derived by subtracting the viscous boundary conditions on $\vec{u}_d$ from the viscous boundary conditions on the complete velocity field. Jump conditions for $\vec{u}_d$ are:

$$
\begin{aligned}
\big[\boldsymbol{D}^d_{nn}\big] &= 0 \qquad (37) \\
\big[\hat{t}\cdot\boldsymbol{\tau}^d\cdot\hat{n}\big] &= 0 \\
\big[\vec{u}_d\cdot\hat{t}\big] &= 0.
\end{aligned}
$$

The jump condition (37) is a kinematic requirement of incompressible flow. Jump conditions for $\vec{u}_p$ are:

$$
\begin{aligned}
\big[\vec{u}_p\big] &= 0 \\
\big[\hat{t}\cdot\boldsymbol{\tau}^p\cdot\hat{n}\big] &= 0 \\
\big[\boldsymbol{\tau}^p_{nn}\big] &= \big[p\big] - \big[\chi\big]. \qquad (38)
\end{aligned}
$$

Recall that the total pressure p is determined thermodynamically: $p = p(\rho, T)$ ($= \rho RT$ for ideal gases).

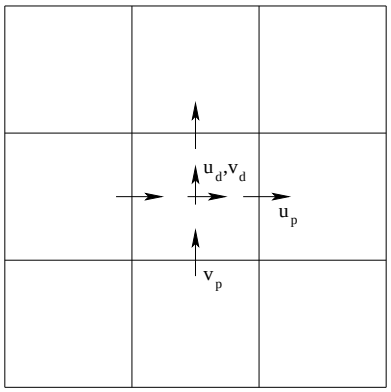

Figure 5: Split velocity discretization – $\vec{u}_d$ is cell-centered, while $\vec{u}_p$ is staggered (face-centered).

Also, for the pressure components π, χ, and δ,

$$\begin{aligned} \big[\pi\big] &= 0 \\ \big[\chi\big] &= \big[\boldsymbol{\tau}^d_{nn}\big] + \kappa\sigma \\ \big[\delta\big] &= \big[p\big] - \big[\chi\big]. \end{aligned} \tag{39}$$

4.2 Discretization of Update

The natural discretization for the split velocity field results in a cell-centered $\vec{u}_d$ and a face-centered (staggered-grid) $\vec{u}_p$ (Fig 5). The pressures p, π, χ and δ are cell-centered, as is the vorticity-generation quantity $\vec{\zeta}$. At the beginning of the timestep to advance the solution from time t^n to time $t^{n+1} = t^n + \Delta t$, we have the velocities, $\vec{\zeta}$, and pressures at time t^n ($\vec{u}^n = (\vec{u}^n_d, \vec{u}^n_p)$), $\vec{\zeta}^n$, p^n, χ^n, and δ^n). As in [2], π is lagged at the half-time $t^{n-\frac{1}{2}}$: $\pi^{n-\frac{1}{2}}$. The update will advance all quantities by Δt.

As in [3], we will maintain second-order accuracy in time for $\vec{u}_d$, while $\vec{u}_p$ will only be first-order accurate in time. Note that this implies that $\vec{\zeta}$ is also first-order in time in the current approach.

The update proceeds as follows. First, compute an approximation to the incompressible velocity $\vec{u}_d^{\,n+\frac{1}{2}}$ at cell faces at the half-time $t^{n+\frac{1}{2}}$, using the unsplit approach outlined in [7]. Similarly, compute face-centered approximations to the density and temperature at $t^{n+\frac{1}{2}}$, using the face-centered velocity field $\vec{u}^{n+\frac{1}{2}} = \vec{u}_d^{\,n+\frac{1}{2}} + \vec{u}_p^n$.

Then, we compute the update to the density field:

$$\rho^{n+1} = \rho^n - \Delta t \ \nabla \cdot (\rho \vec{u})^{n+\frac{1}{2}}. \tag{40}$$

At this point, do an energy equation update. First, compute an approximation to the temperature at t^{n+1}:

$$T^* = T^n - \Delta t(\vec{u} \cdot \nabla)T^{n+\frac{1}{2}} + \frac{\Delta t}{\rho^{n+\frac{1}{2}} c_v}\Big(L(T^n, T^*, \lambda) + \boldsymbol{D}^* :: \boldsymbol{\tau}^n\Big) - (\gamma - 1)T^n \Delta t(\nabla \cdot \vec{u}_p)$$

$$\Big[\lambda \frac{\partial T^*}{\partial n}\Big] = 0 \quad \text{at} \ \ \partial\Omega^{1/2}(t^{n+1})$$

$$\big[T^*\big] = 0 \quad \text{at} \ \ \partial\Omega^{1/2}(t^{n+1}),$$

where $L(T^n, T^*, \lambda)$ is the diffusive term $\nabla \cdot \lambda \nabla T$ computed using the algorithm specified in section 3.3 and (4) has been used to eliminate the $\frac{Dp}{Dt}$ term in (3). Then, do a conservative energy equation update:

$$(\rho e)^{n+1} = (\rho e)^n + \Delta t(-\nabla \cdot (\rho \vec{u} h)^{n+\frac{1}{2}} + L(T^n, T^*, \lambda)) + \boldsymbol{D} :: \boldsymbol{\tau})$$

The new-time temperature t^{n+1} may then be found by inverting $e(T)$.

As before, compute a provisional approximation to the incompressible velocity field:

$$\vec{u}_d^* = \vec{u}_d^n + \Delta t\Big(-\mathcal{A}_d(\vec{u}_d, \vec{u}_p) - \frac{1}{\rho^{n+\frac{1}{2}}}\nabla \pi^{n-\frac{1}{2}} + \frac{1}{\rho^{n+\frac{1}{2}}}\nabla \cdot (\boldsymbol{\tau}(\vec{u}_d^*, \vec{u}_d^n) + \zeta^n\Big)$$

with jump boundary conditions:

$$\big[\hat{n} \cdot def(\vec{u}_d^*) \cdot \hat{n}\big] = 0$$

$$\big[\vec{u}_d^*\big] = 0.$$

Then, compute χ^{n+1} and increment $\vec{u}_d^*$:

$$\nabla \cdot \frac{1}{\rho^{n+1}}\nabla \chi^{n+1} = 0$$

$$\Big[\frac{1}{\rho^{n+1}}\frac{\partial \chi}{\partial n}\Big] = 0 \ \ \text{at} \quad \partial\Omega^{1/2}(\mathrm{t}^{\mathrm{n+1}})$$

$$\big[\chi\big] = \kappa\sigma - \big[\boldsymbol{\tau}_{nn}(\vec{u}_d^*)\big] \ \ \text{at} \quad \partial\Omega^{1/2}(\mathrm{t}^{\mathrm{n+1}})$$

$$\vec{u}_d^* := \vec{u}_d^* - \frac{1}{2}(\frac{1}{\rho^{n+1}}\nabla \chi^{n+1} + \frac{1}{\rho^n}\nabla \chi^n)$$

Finally, project $\vec{u}_d^*$:

$$\vec{u}_d^* := \mathbb{P}_{\rho^{n+1}}\Big(\vec{u}_d^* + \frac{\Delta t}{\rho^{n+1}}\nabla\pi^{n-\frac{1}{2}} - \Delta t\zeta^n\Big)$$

$$\pi^{n+\frac{1}{2}} = \frac{1}{\Delta t}(L_{\rho^{n+1}})^{(-1)}\nabla\cdot\Big(\vec{u}_d^* + \frac{\Delta t}{\rho^{n+1}}\nabla\pi^{n-\frac{1}{2}} - \Delta t\zeta^n\Big)$$

$$\Big[\pi^{n+\frac{1}{2}}\Big] = \big[\boldsymbol{\tau}_{nn}\big] + \kappa\sigma$$

$$\Big[\frac{1}{\rho^{n+1}}\frac{\partial\pi}{\partial n}\Big] = \big[\vec{u}^*\cdot\hat{n}\big]$$

The equations for the preliminary updates for $\vec{u}_p^*$ and δ^* are:

$$\begin{aligned}\vec{u}_p^* &= \vec{u}_p^n + \Delta t\Big(-\nabla\big(\frac{|\vec{u}_p|^2}{2}\big) - \frac{1}{\rho}\nabla\delta^* - \zeta^n + \frac{1}{\rho}f_p \\ &\quad + Av^{C\to F}\frac{1}{\rho^{n+\frac{1}{2}}}\big(\nabla\cdot(\boldsymbol{\tau}(\vec{u}_p^n))\big)\Big) \qquad (41)\\ \delta^* &= \delta^n + \Delta t\Big(-\rho c^2\ \nabla\cdot\vec{u}_p^* - \vec{u}\cdot\nabla p - \frac{\partial\pi}{\partial t} - \frac{\partial\chi}{\partial t}\Big) \qquad (42)\\ &\quad + \Delta t(\gamma-1)\Big(L(T^{n+1},\lambda) + \boldsymbol{D}::\boldsymbol{\tau}\Big)\\ &\quad + \Delta t\ \eta(\delta^* + \pi^{n+\frac{1}{2}} + \chi^{n+1} - p(\rho^{n+1},T^{n+1})).\end{aligned}$$

Since there are two redundant ways that the pressure is computed in this approach which are equivalent at the PDE level, but which can differ due to numerical drift, the last term in (42) is designed to drive the kinematically-determined pressure $p = \pi + \chi + \delta$ and the thermodynamic pressure $p = p(\rho, T)$ together. Similar approaches have been used in [7, 9].

Solve (41) and (42) simultaneously (this leads to a Helmholtz equation for δ^*) with the jump relations

$$\Big[\vec{u}_p\Big] = 0$$

$$\big[\delta\big] = \big[p\big] - \big[\chi\big]$$

$$\begin{aligned}\Big[\frac{1}{\rho}\frac{\partial\delta}{\partial n}\Big] =& \frac{1}{\Delta t}\big[\vec{u}_p^n\cdot\hat{n}^{n+1}\big]\\ &+ \Big[\Big(-\zeta^n - \nabla\Big(\frac{|\vec{u}_p^n|^2}{2}\Big) + \frac{1}{\rho^n}Av^{C\to F}(\nabla\cdot\boldsymbol{\tau}_p^n)\Big)\cdot\hat{n}^{n+1}\Big].\end{aligned}$$

Then, compute the backward Euler estimate of the viscous terms for $\vec{u}_p$:

$$\vec{u}_p^{**} = Av^{F\to C}(\vec{u}_p^{*}) - \Delta t\ \nabla\cdot(\boldsymbol{\tau}(\vec{u}_p^{n})) + \Delta t\ \nabla\cdot(\boldsymbol{\tau}(\vec{u}_p^{**}))$$
$$\vec{u}_p^{*} := \vec{u}_p^{*} + \Delta t Av^{C\to F}\Big(\nabla\cdot(\boldsymbol{\tau}(\vec{u}_p^{**})) - \nabla\cdot(\boldsymbol{\tau}(\vec{u}_p^{n}))\Big).$$

We perform this update in two steps due to centering requirements – $\vec{u}_p^{**}$ is cell-centered, while $\vec{u}_p^{*}$ is face-centered. Finally, use the projection operators to complete the velocity updates:

$$\vec{u}_p^{n+1} = \mathbb{Q}_0\vec{u}_p^{*}$$
$$\zeta^{n+1} = \zeta^{n} + \frac{\vec{u}_p^{n+1} - \vec{u}_p^{*}}{\Delta t}$$
$$\vec{u}_d^{n+1} = \vec{u}_d^{*} + \Delta t\zeta^{n+1}.$$

5 Spatial Discretizations

5.1 Computing Interface Values

To complete operator discretizations at multifluid interfaces, we will follow the approach used in [5] for discretizing Poisson's equation at embedded boundaries. In cells intersected by the multifluid interface ("irregular cells") or when regular operator discretizations cross the multifluid interface, we will need to compute solution values and derivatives at the interface itself, in order to enforce the matching conditions. We will demonstrate the case where both a Dirichlet and a Neumann jump condition must be satisfied (as when computing the projection). Simple Dirichlet or Neumann conditions will use the same discretizations outlined here. Figure 6 depicts an example of a multifluid interface, with $\alpha = 1$ on the left of the interface, and $\alpha = 2$ on the right. Point B is at the center of the interface in cell (i, j), and $\hat{n}_B^{(\alpha)}$ is the outward normal to the interface at point B for phase α (so there are two normal vectors, one for each phase.) As in [5], values in partial cells are considered to be located at the center of the uncut cell, instead of at the centroid of the partial cell. Therefore, cells which contain the interface $\partial\Omega^{1/2}$ contain two cell-centered values for each variable – one for each phase.

Since the angle of the interface at B in Figure 6 is less than $\frac{\pi}{4}$, points $1^{(\alpha)}$ and $2^{(\alpha)}$ are intersections of $\hat{n}_B^{(\alpha)}$ with the cell-centers in the x-direction. (If the angle of the interface were between $\frac{\pi}{4}$ and $\frac{3\pi}{4}$, we would look at cell

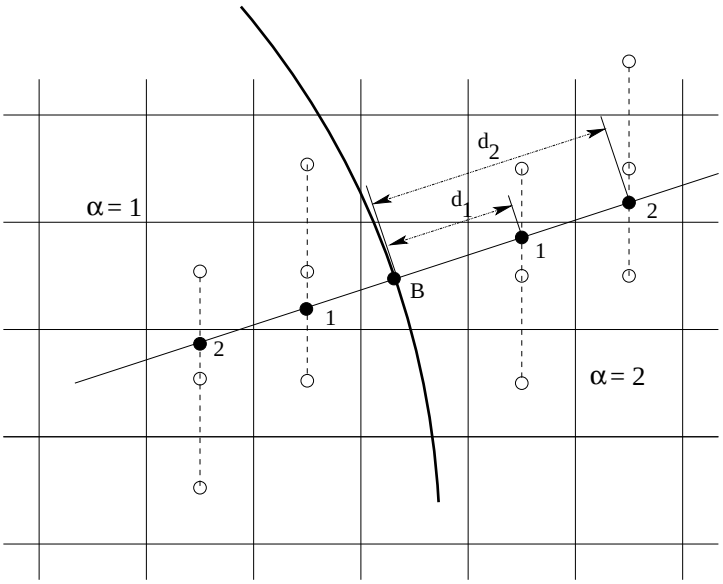

Figure 6: Discretization at fluid interface. The point B is at the centroid of the fluid interface $\partial\Omega^{1/2}$ for the cell (i, j).

centers in the $y-$direction, etc.) The distances $d_1^{(\alpha)}$ and $d_2^{(\alpha)}$ are the positive distances between point B and points $1^{(\alpha)}$ and $2^{(\alpha)}$ respectively. Given these, we can define the gradient (for example, for p) at point B as follows:

$$\left(\frac{\partial p}{\partial n}\right)^{(\alpha)} = (\pm)_\alpha \frac{1}{d_2^{(\alpha)} - d_1^{(\alpha)}} \left((p_B^{(\alpha)} - p_1^{(\alpha)}) \frac{d_2^{(\alpha)}}{d_1^{(\alpha)}} - (p_B^{(\alpha)} - p_2^{(\alpha)}) \frac{d_1^{(\alpha)}}{d_2^{(\alpha)}} \right), \qquad (43)$$

where

$$(\pm)_\alpha = \begin{cases} + & \alpha = 1 \\ - & \alpha = 2. \end{cases}$$

As in [5], we compute $p_1^{(\alpha)}$ and $p_2^{(\alpha)}$ using quadratic interpolation of the cell-centered values of $p^{(\alpha)}$ along the line "normal" to $\hat{n}$ (the dashed lines in Figure 6). On both sides of the interface, $\hat{n}$ is defined to point outward from

B to points 1 and 2. The two phases are linked through the jump relations:

$$\begin{aligned} g_N(\boldsymbol{x}_B) &= \Big[\frac{\partial p}{\partial n}\Big] \\ &= \Big(\frac{\partial p}{\partial n}\Big)_B^{(1)} - \Big(\frac{\partial p}{\partial n}\Big)_B^{(2)} \\ g_D(\boldsymbol{x}_B) &= \big[p\big] = p_B^{(1)} - p_B^{(2)}. \end{aligned}$$

This gives two equations in two unknowns:

$$\begin{pmatrix} \eta^{(1)} & \eta^{(2)} \\ 1 & -1 \end{pmatrix} \begin{pmatrix} p_B^{(1)} \\ p_B^{(2)} \end{pmatrix} = \begin{pmatrix} r^N \\ r^D \end{pmatrix}$$

where

$$\eta^{(\alpha)} = \frac{1}{d_2^{(\alpha)} - d_1^{(\alpha)}} \Big(\frac{d_2^{(\alpha)}}{d_1^{(\alpha)}} - \frac{d_1^{\alpha}}{d_2^{(\alpha)}} \Big)$$

where $d_2^{(\alpha)} > d_1^{(\alpha)}$, and

$$r^N = g_N(\boldsymbol{x}_B) + \sum_{\alpha} (\pm)_{\alpha} \big(p_1^{(\alpha)} \frac{d_2^{(\alpha)}}{d_1^{(\alpha)}} - p_2^{(\alpha)} \frac{d_1^{(\alpha)}}{d_2^{(\alpha)}} \big) \frac{1}{d_2^{(\alpha)} - d_1^{(\alpha)}}$$

$$r^D = g_D(\boldsymbol{x}_B).$$

Since $\eta^{(\alpha)} > 0$ for $\alpha = \{1, 2\}$, this is always invertible.

Implementing this algorithm requires both face-centered (MAC) and cell-centered (CC) projections. As in [7], we will start with the MAC projection discretization, then build the cell-centered projection based on the MAC operators in combination with appropriate averaging operators.

5.2 MAC Operators

The MAC divergence is a cell-centered divergence of a face-centered vector field. The face-centered vector field required for this operator consists of the component normal to each face, including a normal velocity at the centroid of the multifluid interface ($\vec{u} \cdot \hat{n^B}$). Note that in the presence of a multifluid interface, the value of the face-centered vector field may be double-valued at the interface, since the values in each phase at the interface may not be the same.

In cells without a multifluid interface, discretization is straightforward. We use a finite-volume discretization to make extension to the partial-cell case more straightforward. In two dimensions,

$$D(\vec{u})_{ij} = \frac{1}{\Delta x \Delta y}\big(\Delta y(u_{i+\frac{1}{2},j} - u_{i-\frac{1}{2},j}) + \Delta x(v_{i,j+\frac{1}{2}} - v_{i,j-\frac{1}{2}})\big).$$

In the case of a cell with a fluid interface, we use the approach already outlined in (20). The interface value $\vec{u}_B$ is centered at point B, which is the centroid of the interface. Then, the face-centered divergence for phase α in cell $\boldsymbol{i}$ is:

$$\begin{aligned} D(\vec{u}^{(\alpha)})_{\boldsymbol{i}} = \ & \tfrac{1}{\Delta x \Delta y \kappa_{\boldsymbol{i}}^{(\alpha)}}\big(\ (a^{\alpha}_{i+\frac{1}{2},j}\Delta y)u^{(\alpha)}_{i+\frac{1}{2},j} - (a^{\alpha}_{i-\frac{1}{2},j}\Delta y)u^{(\alpha)}_{i-\frac{1}{2},j} \\ & + (a^{\alpha}_{i,j+\frac{1}{2}}\Delta x)v^{\alpha}_{i,j+\frac{1}{2}} - (a^{\alpha}_{i,j-\frac{1}{2}}\Delta x)v^{\alpha}_{i,j-\frac{1}{2}} \\ & - \alpha^{B}_{\boldsymbol{i}}(\vec{u}^{\alpha}_{B}\cdot\hat{n}^{\alpha}_{B})\big). \end{aligned}$$

The MAC gradient is computed in the same way as in [5]. For faces which are not coincident with multifluid interfaces, we use the standard centered-difference discretization:

$$\begin{aligned} G_x(\phi)^{(\alpha)}_{i+\frac{1}{2},j} &= \frac{1}{\Delta x}(\phi^{\alpha}_{i+1,j} - \phi^{\alpha}_{i,j}) \\ G_y(\phi)^{\alpha}_{i,j+\frac{1}{2}} &= \frac{1}{\Delta y}(\phi^{\alpha}_{i,j+1} - \phi^{\alpha}_{i,j}). \end{aligned} \tag{44}$$

On the interface, we compute the normal derivative at the interface using (43), which gives $(G(\phi)^{(\alpha)}\cdot\hat{n}_B)$.

To compute the transverse component of the MAC gradients (for correcting transverse components of the velocity field for example) near the multifluid interface, we simply use quadratic extrapolation of the cell-centered ϕ to any completely covered cells in the usual stencil for computing the transverse gradient. For example, on the $x-$faces, the stencil for computing $G^{MAC}_{y}\phi$ is normally

$$\begin{aligned} (G^{MAC}_{x}\phi)_{i,j+\frac{1}{2}} &= \frac{\phi_{i+1,j+1} + \phi_{i+1,j} - \phi_{i-1,j+1}, \phi_{i-1,j}}{4\Delta x} \\ (G^{MAC}_{y}\phi)_{i+\frac{1}{2},j} &= \frac{\phi_{i+1,j+1} + \phi_{i,j+1} - \phi_{i+1,j-1}, \phi_{i,j-1}}{4\Delta y}. \end{aligned} \tag{45}$$

In the example shown in Figure 6, when computing the $x-$component of the gradient at the $(i, j+\frac{1}{2})$ face for the $\alpha = 2$ phase, the $(i-1, j)$ cell is

completely out of the $\Omega^{(2)}$ domain. We extrapolate a value to the cell center of the $(i-1,j)$ cell using the values of $\phi^{(2)}$ which we do have, then use the extrapolated value in the standard stencil (45). In the current approach, we will not need transverse gradients at the multifluid interface (point B in Figure 6).

Then, the Laplacian operator used in the MAC computation is simply the divergence of the gradient:

$$L(\phi)_{\boldsymbol{i}}^{(\alpha)} = D^{MAC}(G^{MAC}(\phi))_{\boldsymbol{i}}^{\alpha}.$$

The face-centered (MAC) projection operator applied to a staggered-grid velocity field $\vec{u}_{face}$ is then discretized as

$$\mathbb{P}^{\mathbb{MAC}}(\vec{u}_{face}) = (I - G(L^{-1})D)\vec{u}_{face} \tag{46}$$

5.3 Cell-Centered Projection Operators

We construct the operators for the cell-centered projection in the same way as in [7], by combining the MAC-centered operators defined previously with appropriate averaging from cells-to-faces ($Av^{C\to F}$) or faces-to-cells ($Av^{F\to C}$).

To construct a cell-centered divergence operator $D^{CC}\vec{u}^{CC}$, we average the cell-centered velocity field to faces, then apply the face-centered divergence operator:

$$D^{CC}\vec{u}^{CC} = D^{MAC}(Av^{C\to F}\vec{u}^{CC}).$$

Since every face where we need a face-centered velocity will always have valid data on either side of it, there is no need to extrapolate data. However, since the face-centered velocities needed for the MAC divergence are centered at the center of the aperture, while the cell-centered data is located at cell centers, we will need to interpolate the face-centered data to the center of the aperture (Figure 7). To complete the stencil for the MAC divergence in cells with multifluid interfaces, we also need to compute a value for $\vec{u}\cdot\hat{n}$. To maintain the global accuracy of the operator, we require an $O(h^2)$ value, which we interpolate using bilinear interpolation of the cell-centered vector field $\vec{u}$; these interpolated values may likely be double-valued at the interface.

To compute the cell-centered gradient $G^{CC}\phi^{CC}$, we compute the normal component of the face-centered gradient, then average the face-centered gradient to cell centers:

$$G^{CC}\phi^{CC} = Av^{F\to C}G^{MAC}\phi^{CC}.$$

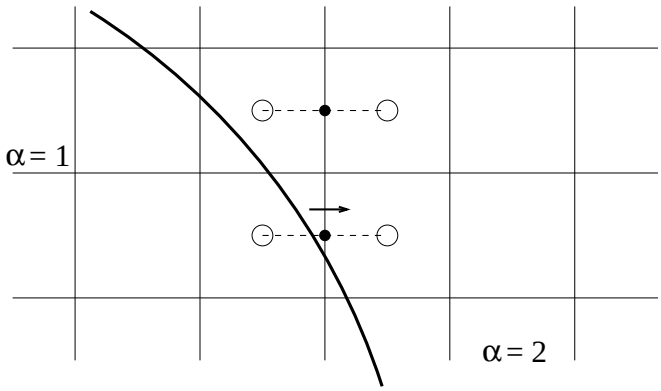

Figure 7: Cell-to-face averaging for a partial face. First average cell-centered data (open circles) to centers of faces (solid circles), then interpolate to get aperture-centered face value (arrow).

Near the multifluid interface, both faces necessary for simple averaging of faces to cells may not be available. In that case, extrapolate face-centered values to the unavailable face, then average to the cell center as usual (Figure 8).

Finally, for the cell-centered approximate projection formulation, we use the same Laplacian operators as for the MAC discretization (L and L_ρ).

The cell-centered projection operator applied to a cell-centered velocity field $\vec{u}^{CC}$ is then discretized as

$$\mathbb{P}^{\mathbb{CC}}(\vec{u}^{CC}) = (I - G^{CC}(L^{-1})D^{CC})\vec{u}^{CC} \tag{47}$$

6 Specific Algorithm Details and Notes

6.1 Nonlinear Advection

Following [3], we compute $\mathcal{A}_d(\vec{u}_d, \vec{u}_p)$ at cell centers. We first average $\vec{u}_p$ to cell centers, then predict an upwinded, face-centered $\vec{u}_d^{n+\frac{1}{2}}$, applying a MAC projection to ensure that $\vec{u}_d^{n+\frac{1}{2}}$ is divergence-free. We then use the face-centered ($\vec{u}_d^{n+\frac{1}{2}}$) and ($\vec{u}_p^n$). Note that we make no attempt to compute time-centered $\vec{u}_p$ terms, since $\vec{u}_p$ is only first-order in time anyway.

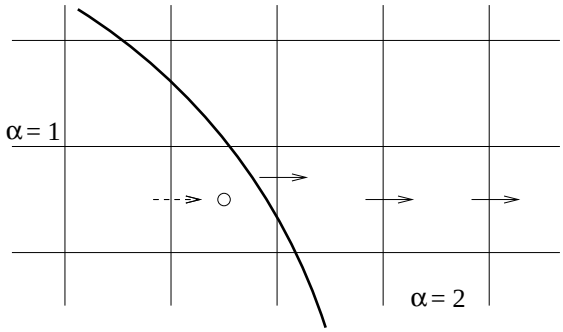

Figure 8: Face-to-cell averaging near a multifluid interface. To compute averaged value at cell center (open circle) requires an unavailable face-centered value (dashed arrow). Compute value for that face by extrapolating using the three solid arrows.

6.2 Potential Advective Term

We average transverse components of $\vec{u}_p^n$ to faces so that we have all components of $\vec{u}_p$ co-located at each face, then compute $|\vec{u}_p^n|^2$ at faces. Then we use the face-centered gradient operator to compute $G^{MAC}(\frac{|\vec{u}_p^n|^2}{2})$.

6.3 Viscous Terms

For the vortical component $\vec{u}_d$, we compute the viscous terms $[\frac{1}{\rho}\nabla\cdot(\mu\ def(\vec{u}_d))]^{n+\frac{1}{2}}$ using the TGA algorithm outlined in Section 3.3 in the same way as in the INS algorithm. For the potential component, we average $\vec{u}_p$ to cell-centers and use backward Euler to compute the viscous terms $[\frac{1}{\rho}\nabla\cdot(\mu\ def(\vec{u}_p))]^{n+1}$. We then average back to faces.

6.4 Pressure Update

As in [3], we combine (41) and (42) and solve for δ^{n+1}, resulting in a Helmholtz equation for δ^{n+1}:

$$\begin{aligned}((1-\Delta t\eta)I-\rho(c\Delta t)^2L_\rho)\delta^{n+1} &= \delta^n-\Delta t\rho c^2D^{MAC}\vec{u}_p^n \\ &+\Delta t^2\rho c^2D^{MAC}G^{MAC}\big(\frac{|\vec{u}_p|^2}{2}\big)^n \\ &-(\Delta t)^2\rho c^2D^{MAC}[\frac{1}{\rho^{n+\frac{1}{2}}}\nabla\cdot(\mu\ def(\vec{u}_p))]^{n+1} \\ &-\Delta t^2\rho c^2D^{MAC}\big(\vec{\zeta}^n+\vec{f}_p\big) \\ &+\Delta t(\gamma-1)\Big(L^(T^{n+1},\lambda)+\boldsymbol{D}::\boldsymbol{\tau}\Big) \\ &-\Delta t\big(\frac{\partial\pi}{\partial t}\big)^n-\Delta t\big(\frac{\partial\chi}{\partial t}\big)^n-\Delta t(\vec{u}\cdot G^{MAC}p)^n \\ &+\Delta t\eta(\pi+\chi-\rho RT)\end{aligned} \tag{48}$$

Note that to compute the pressure in the last term in (48), we have δ^n, but $\pi^{n\pm\frac{1}{2}}$. We compute $\pi^n=\frac{1}{2}(\pi^{n-\frac{1}{2}}+\pi^{n+\frac{1}{2}})$, then $p^n=\pi^n+\delta^n$. The time derivative of π is

$$\Big(\frac{\partial\pi}{\partial t}\Big)^n=\frac{\pi^{n+\frac{1}{2}}-\pi^{n-\frac{1}{2}}}{\Delta t}.$$

6.5 Projections

For the vortical velocity field $\vec{u}_d$, we use the cell-centered approximate projection used in [7]:

$$\vec{u}_d^*=\vec{u}_d^n+\Delta t\Big(-\mathcal{A}_d(\vec{u}_d,\vec{u}_p)+[\frac{1}{\rho}\nabla\cdot(\mu\ def(\vec{u}_d))]^{n+\frac{1}{2}}+\vec{f}_d\Big)$$
$$\vec{u}_d^{n+1}=\mathbb{P}_0(\vec{u}_d^*).$$

In the course of projecting $\vec{u}_d$, we also compute $\pi^{n+\frac{1}{2}}$.

To update $\vec{u}_p$ and compute the vorticity-generation term $\vec{\zeta}$, we apply a face-centered (MAC) projection to the pressure gradient and viscous terms

in (35):

$$\begin{aligned}
\vec{u}_p^{n+1} &= \vec{u}_p^n - \Delta t\ grad\Big(\frac{|\vec{u}_p|^2}{2}\Big) \\
&\quad +\mathbb{Q}_0^{MAC}\Big(\frac{\Delta t}{\rho}\big(\nabla\cdot(\mu\ def(\vec{u}_p))^{n+1} - \nabla\delta^{n+1}\big)\Big) \\
\vec{\zeta}^{n+1} &= \mathbb{P}_0^{MAC}\Big(\frac{\Delta t}{\rho}[\nabla\cdot(\mu\ def(\vec{u}_p))]^{n+1} - \frac{\Delta t}{\rho}\nabla\delta^{n+1}\Big)
\end{aligned} \tag{49}$$

7 AMR for Incompressible Multifluid Systems

As a special case, we will compute locally refined solutions for incompressible multifluid systems. In this case, the velocity field will be divergence-free ($\vec{u} = \vec{u}_d, \vec{u}_p = 0$). Away from the multifluid interface, the algorithm reduces to the incompressible projection method outlined in [4].

Our strategy for computing locally refined solutions will be to refine the multifluid interface to the finest resolution in the AMR hierarchy. In this way, the interface dynamics are resolved as finely as possible, while also simplifying the algorithm by avoiding the complications of a multifluid interface crossing a coarse-fine interface. Once this has been specified, the multifluid and AMR aspects of the algorithm become decoupled; the multifluid interface dynamics are treated as outlined in this document, while the otherwise the algorithm used will be the one outlined in [4] for computing solutions to the incompressible Navier-Stokes equations with local refinement.

References

[1] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. Welcome. A conservative adaptive projection method for the variable density incompressible Navier-Stokes equations. *Journal of Computational Physics*, 142(1):1–46, May 1998.

[2] J. B. Bell, P. Colella, and H. M. Glaz. A second-order projection method for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 85:257–283, 1989.

[3] Phillip Colella and Karen Pao. A projection method for low speed flows. *J. Comput. Phys.*, 1999.

[4] Applied Numerical Algorithms Group. Incompressible Navier-Stokes software design. Technical report, NERSC, 2002.

[5] Hans Johansen and Phillip Colella. A cartesian grid embedded boundary method for Poisson's eqaution on irregular domains. *J. Comput. Phys.*, 1998.

[6] Ming-Chih Lai and Zhilin Li. A remark on jump conditions for the three-dimensional Navier-Stokes equations involving an immersed moving membrane. *Applied Mathematics Letters*, 2001.

[7] D Martin and P Colella. A cell-centered adaptive projection method for the incompressible Euler equations. *J. Comput. Phys.*, 2000.

[8] P McCorquodale and P Colella. private communication? 2003.

[9] R. B. Pember, L. H. Howell, J. B. Bell, P. Colella, W. Y. Crutchfield, W. A. Fiveland, and J. P. Jessee. An adaptive projection method for unsteady, low mach number combustion. *Combustion Science and Technology*, 140:123–168, 1998.

[10] D Trebotich and P Colella. A projection method for incompressible viscous flow on moving quadrilateral grids. *J. Comput. Phys.*, 2001.

[11] E.H. Twizell, A.B. Gumel, and M.A. Arigu. Second-order, l_0-stable methods for the heat equation with time-dependent boundary conditions. *Advances in Computational Mathmatics*, 6:333–352, 1996.