Notes on incompressible solvers

These summarize methods for solving the incompressible hydrodynamics equations using an cell-centered approximate projection method.

1 Incompressible flow

As a fluid parcel advects through a domain, it compresses and expands due to a variety of effects (stratification, local heat release, acoustic/shock waves). The Lagrangian derivative of the density captures the changes in the fluid, and is a measure of its compressiblity. From the continuity equation, we see:

$$\frac{1}{\rho}\frac{D\rho}{Dt} = -\nabla \cdot U \tag{1}$$

A fluid in which the density (and therefore volume) of a fluid element is not allowed to change is called *incompressible*. An incompressible fluid objects the velocity constraint:

$$\nabla \cdot U = 0 \tag{2}$$

(since $D\rho/Dt = 0$). The incompressible fluid approximation is a reasonable approximation when the Mach number of the fluid is small ($\ll 1$). To complete the system, we add the momentum equation. If we take the density to be constant everywhere in the domain (not just in our fluid element), then we have:

$$\frac{\partial U}{\partial t} + U \cdot \nabla U + \nabla p = 0 \tag{3}$$

Note that p here looks like a pressure, but it is not subject to any equation of state. This system is closed as written. The value of p is determined such that the velocity constraint is satisfied.

2 Projection methods

The basic idea behind a projection method is that any vector field can be decomposed into a divergence free part and the gradient of a scalar (this is sometimes called a *Hodge decomposition*). Given a velocity field $U^\star$, we can express it in terms of the divergence free part U^d and a scalar, ϕ as:

$$U^\star = U^d + \nabla\phi \tag{4}$$

Taking the divergence of each side, and noting that $\nabla \cdot U^d = 0$, we have

$$\nabla \cdot U^\star = \nabla^2\phi \tag{5}$$

This is an elliptic equation. Given suitable boundary conditions, we can solve for ϕ (for instance, using multigrid) and recover the divergence free part of $U^\star$ as:

$$U^d = U^\star - \nabla\phi \tag{6}$$

We call this operation of extracting the divergence free part of a velocity field a *projection*. This can be expressed by defining an operator P, such that $PU^\star = U^d$, and $(I - P)U^\star = \nabla\phi$. From the momentum equation, we see that $\partial U/\partial t + \nabla\phi$ is in the form of a divergence free term + the gradient of a scalar. This means that advancing the velocity field subject to the constraint involves solving:

$$U_t = P(U_t + \nabla p) = P(-U \cdot \nabla U) \tag{7}$$

See Bell, Colella, and Howell [4] for a nice discussion of this.

The original projection method for incompressible flows goes back to Chorin [7]. Instead of evolving Eq. 7 directly, we break the update into pieces. The basic idea is to evolve the velocity advection equation without regard to the constraint, yielding a *provisional velocity* field which is then subjected to a projection to enforce the divergence-free constraint. Bell, Colella, and Glaz (BCG) [3] introduced a projection method that uses standard Godunov methods for the advection terms (much like is done with compressible flow) and then solves an elliptic equation to enforce the constraint. This division of the operations in the algorithm (advect then project) is a type of *fractional step* method.

There are different ways to discretize the operators that make up the projection. We denote the discretized divergence as D and the discretized gradient operation as G. For an exact projection, the discretized Laplacian, L, would be the same as applying G and D in succession (i.e. $L = DG$). Depending on our data centerings, we may prefer to discretize the Laplacian independently to D and G, such that $L \neq DG$. This is called an *approximate projection*.

Many variations on this basic idea exist, using alternate forms of the projection, different grid centerings of the ϕ variable, and additional physics.

3 Cell-centered approximate projection solver

Here we describe an incompressible algorithm that uses cell-centered data throughout—U and p are both cell-centered. The projection at the end is an approximate projection. The basic algorithm flow is

- Create the time-centered advective velocities through the faces of the zones.
- Project the advective velocities such that they obey the velocity constraint
- Construct the time-centered interface states of all quantities on the faces of the zones using the advective velocity.
- Update the velocity to the new time. This is defines the provisional velocity field—it does not yet satisfy the constraint.
- Enforce the velocity constraint by projecting the velocity.

The description below is pieced together from a variety of sources. BCH describes a cell-centered method, but with an exact projection (with a larger, decoupled stencil). Almgren, Bell, and Szymczak (ABS) [2] describes an approximate projection method, but with a node-centered final projection. We follow this paper closely up until the projection. Martin and Colella [10] (and Martin's PhD thesis) method uses a cell-centered projection, as is described here. They go into additional effort to describe this for a refined grid. All of these method are largely alike, aside from how the discretization of the final projection is handled.

3.1 Advective velocity

In predicting the interface states, we first seek to construct the velocities through the interfaces. A key concept throughout the advection step is that once we have the normal velocities on the interfaces, we can use these to upwind left and right states of any quantity to get their interface value. The advective velocities we construct here, u^{adv} and v^{adv}, will later be used to upwind the various states we predict to the interfaces. We only need the velocity through the interfaces, as shown in the figure 1. This staggered grid arrangement is sometimes called a MAC grid.

We follow ABS. Our velocity evolution system (writing out the individual components of U: u and v) is

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - v\frac{\partial u}{\partial y} - \frac{\partial p}{\partial x} = 0 \tag{8}$$

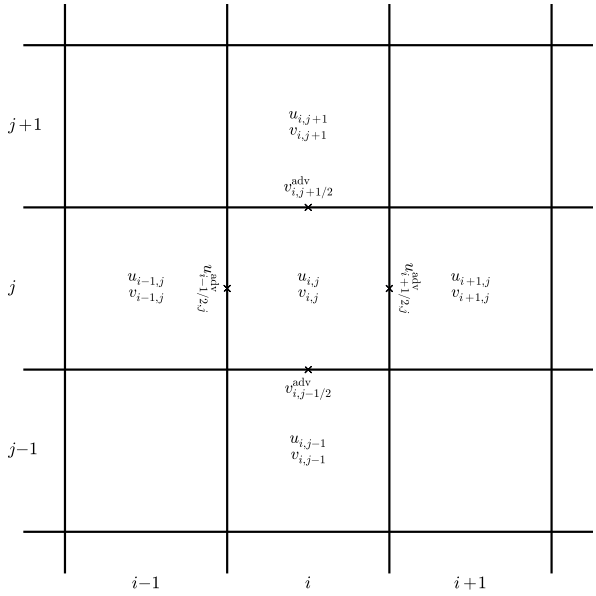

Figure 1: The staggered 'MAC' grid for the advective velocities

$$\frac{\partial v}{\partial t} = -u\frac{\partial v}{\partial x} - v\frac{\partial v}{\partial y} - \frac{\partial p}{\partial y} = 0 \tag{9}$$

Our goal in this step is to predict time-centered interface values of the normal velocity (u on x-edges and v on y-edges). The prediction follows from Taylor expanding the state to the interface (through $\Delta x/2$ or $\Delta y/2$) and to the half-time (through $\Delta t/2$). As with the regular advection, we can have left and right states which we will resolve by solving a Riemann problem. The left interface state of u at $i+1/2, j$ is found as:

$$u^{n+1/2}_{i+1/2,j,L} = u_{i,j} + \frac{\Delta x}{2}\left.\frac{\partial u}{\partial x}\right|_{i,j} + \frac{\Delta t}{2}\left.\frac{\partial u}{\partial t}\right|_{i,j} \tag{10}$$

$$= u_{i,j} + \frac{\Delta x}{2}\left.\frac{\partial u}{\partial x}\right|_{i,j} + \frac{\Delta t}{2}\left.\left(-u\frac{\partial u}{\partial x} - v\frac{\partial u}{\partial y} - \frac{\partial p}{\partial x}\right)\right|_{i,j} \tag{11}$$

$$= u_{i,j} + \frac{\Delta x}{2}\left(1 - \frac{\Delta t}{\Delta x}u_{i,j}\right)\left.\frac{\partial u}{\partial x}\right|_{i,j} - \frac{\Delta t}{2}\left(v\frac{\partial u}{\partial y}\right)_{i,j} - \frac{\Delta t}{2}\left.\frac{\partial p}{\partial x}\right|_{i,j} \tag{12}$$

$$\tag{13}$$

We express $\partial u/\partial x|_{i,j}$ as $\overline{\Delta u}^{(x)}_{i,j}/\Delta x$, where $\overline{\Delta u}^{(x)}_{i,j}$ is the limited slope of u in the x-direction in zone i,j. Our interface state is then:

$$u^{n+1/2}_{i+1/2,j,L} = \underbrace{\frac{1}{2}\left(1 - \frac{\Delta t}{\Delta x}u\right)\overline{\Delta u}^{(x)}_{i,j}}_{\equiv \hat{u}^{n+1/2}_{i+1/2,j,L}} - \underbrace{\frac{\Delta t}{2}\left(v\frac{\partial u}{\partial y}\right)_{i,j}}_{\text{transverse term}} - \frac{\Delta t}{2}\left.\frac{\partial p}{\partial x}\right|_{i,j} \tag{14}$$

Similarly, for v through the y faces, we find:

$$v^{n+1/2}_{i,j+1/2,L} = \underbrace{\frac{1}{2}\left(1 - \frac{\Delta t}{\Delta x}v\right)\overline{\Delta v}^{(y)}_{i,j}}_{\equiv \hat{v}^{n+1/2}_{i,j+1/2,L}} - \underbrace{\frac{\Delta t}{2}\left(u\frac{\partial v}{\partial x}\right)_{i,j}}_{\text{transverse term}} - \frac{\Delta t}{2}\left.\frac{\partial p}{\partial y}\right|_{i,j} \tag{15}$$

As indicated above (and following ABS and the similar notation used by Colella [8]), we denote the quantities that will be used to evaluate the transverse states (consisting only of the normal predictor) with a '^'. These will be used to evaluate the transverse terms labeled above.

We predict u and v to both the x and y interfaces, using only the normal part of the predictor. This gives us the left and right 'hat' states on each interface.

u on x-interfaces:

$$\hat{u}^{n+1/2}_{i+1/2,j,L} = u_{i,j} + \frac{1}{2}\left(1 - \frac{\Delta t}{\Delta x}u_{i,j}\right)\overline{\Delta u}^{(x)}_{i,j} \tag{16}$$

$$\hat{u}^{n+1/2}_{i+1/2,j,R} = u_{i+1,j} - \frac{1}{2}\left(1 + \frac{\Delta t}{\Delta x}u_{i+1,j}\right)\overline{\Delta u}^{(x)}_{i+1,j} \tag{17}$$

v on x-interfaces:

$$\hat{v}^{n+1/2}_{i+1/2,j,L} = v_{i,j} + \frac{1}{2}\left(1 - \frac{\Delta t}{\Delta x}u_{i,j}\right)\overline{\Delta v}^{(x)}_{i,j} \tag{18}$$

$$\hat{v}^{n+1/2}_{i,j+1/2,R} = v_{i+1,j} - \frac{1}{2}\left(1 + \frac{\Delta t}{\Delta x}u_{i+1,j}\right)\overline{\Delta v}^{(x)}_{i+1,j} \tag{19}$$

u on y-interfaces:

$$\hat{u}^{n+1/2}_{i,j+1/2,L} = u_{i,j} + \frac{1}{2}\left(1 - \frac{\Delta t}{\Delta y}v_{i,j}\right)\overline{\Delta u}^{(y)}_{i,j} \tag{20}$$

$$\hat{u}^{n+1/2}_{i,j+1/2,R} = u_{i,j+1} - \frac{1}{2}\left(1 + \frac{\Delta t}{\Delta y}v_{i,j+1}\right)\overline{\Delta u}^{(y)}_{i,j+1} \tag{21}$$

v on y-interfaces:

$$\hat{v}^{n+1/2}_{i,j+1/2,L} = v_{i,j} + \frac{1}{2}\left(1 - \frac{\Delta t}{\Delta y}v_{i,j}\right)\overline{\Delta v}^{(y)}_{i,j} \tag{22}$$

$$\hat{v}^{n+1/2}_{i,j+1/2,R} = v_{i,j+1} - \frac{1}{2}\left(1 + \frac{\Delta t}{\Delta y}v_{i,j+1}\right)\overline{\Delta v}^{(y)}_{i,j+1} \tag{23}$$

Note that the 'right' state is constructed using the data to the right of the interface. Also note that in constructing these transverse velocities, we do not include the p term.

Next we find the advective velocity through each interface. The incompressible velocity equation looks like the inviscid Burger's equation, and the Riemann solver follows that construction. BCG provide the implementation used here (and throughout the incompressible literature). Also see Toro [13]. We denote the resulting velocities with the 'adv' superscript, as these are the normal velocities used to advect the hat states. The Riemann problem solution is:

$$\mathcal{R}(q_L, q_R) = \begin{cases} q_L & \text{if } q_L > 0, \quad q_L + q_R > 0 \\ 0 & \text{if } q_L \le 0, \quad q_R \ge 0 \\ q_R & \text{otherwise} \end{cases} \tag{24}$$

We solve this for each of the normal velocities, giving:

$$\hat{u}^{\text{adv}}_{i+1/2,j} = \mathcal{R}(\hat{u}^{n+1/2}_{i+1/2,j,L}, \hat{u}^{n+1/2}_{i+1/2,j,R}) \tag{25}$$

$$\hat{v}^{\text{adv}}_{i,j+1/2} = \mathcal{R}(\hat{v}^{n+1/2}_{i,j+1/2,L}, \hat{v}^{n+1/2}_{i,j+1/2,R}) \tag{26}$$

These advective velocities (sometimes called the *transverse velocities*) are used to resolve the left and right states of all the hat quantities by simple upwinding. For a u or v state on the x-interface, we upwind based on $\hat{u}^{\text{adv}}$; and for a u or v state on the y-interface, we upwind based on $\hat{v}^{\text{adv}}$. If we write the upwinding as:

$$\mathcal{U}[s^{\text{adv}}](q_L, q_R) = \begin{cases} q_L & \text{if } s^{\text{adv}} > 0 \\ \frac{1}{2}(q_L + q_R) & \text{if } s^{\text{adv}} = 0 \\ q_R & \text{if } s^{\text{adv}} < 0 \end{cases} \tag{27}$$

Then the interface states are:

$$\hat{u}_{i+1/2,j} = \mathcal{U}[\hat{u}^{\rm adv}_{i+1/2,j}](\hat{u}^{n+1/2}_{i+1/2,j,L}, \hat{u}^{n+1/2}_{i+1/2,j,R}) \quad (28)$$

$$\hat{v}_{i+1/2,j} = \mathcal{U}[\hat{u}^{\rm adv}_{i+1/2,j}](\hat{v}^{n+1/2}_{i+1/2,j,L}, \hat{v}^{n+1/2}_{i+1/2,j,R}) \quad (29)$$

$$\hat{u}_{i,j+1/2} = \mathcal{U}[\hat{v}^{\rm adv}_{i,j+1/2}](\hat{u}^{n+1/2}_{i,j+1/2,L}, \hat{u}^{n+1/2}_{i,j+1/2,R}) \quad (30)$$

$$\hat{v}_{i,j+1/2} = \mathcal{U}[\hat{v}^{\rm adv}_{i,j+1/2}](\hat{v}^{n+1/2}_{i,j+1/2,L}, \hat{v}^{n+1/2}_{i,j+1/2,R}) \quad (31)$$

Now we can construct the full left and right predictions for the normal velocities on each interface (Eqs. 14 and 15). This involves simply adding the transverse term to the hat quantities and adding the pressure gradient.

$$u^{n+1/2}_{i+1/2,j,L} = \hat{u}^{n+1/2}_{i+1/2,j,L} - \frac{\Delta t}{2}\left[\frac{1}{2}\left(\hat{v}^{\rm adv}_{i,j-1/2} + \hat{v}^{\rm adv}_{i,j+1/2}\right)\right]\left(\frac{\hat{u}^{n+1/2}_{i,j+1/2} - \hat{u}^{n+1/2}_{i,j-1/2}}{\Delta y}\right) - \frac{\Delta t}{2}(Gp)^{(x),n-1/2}_{i,j} \quad (32)$$

and

$$v^{n+1/2}_{i,j+1/2,L} = \hat{v}^{n+1/2}_{i,j+1/2,L} - \frac{\Delta t}{2}\left[\frac{1}{2}\left(\hat{u}^{\rm adv}_{i-1/2,j} + \hat{u}^{\rm adv}_{i+1/2,j}\right)\right]\left(\frac{\hat{v}^{n+1/2}_{i+1/2,j} - \hat{v}^{n+1/2}_{i-1/2,j}}{\Delta x}\right) - \frac{\Delta t}{2}(Gp)^{(y),n-1/2}_{i,j} \quad (33)$$

Here $(Gp)^{(x),n-1/2}_{i,j}$ and $(Gp)^{(y),n-1/2}_{i,j}$ are difference-approximations to ∇p in the x and y directions respectively. Note that they are lagged—these come from the projection at the end of the previous timestep. See BCG for a discussion. A similar construction is done for the right states at the interface.

Finally, we do a Riemann solve (again, using the Burger's form of the Riemann problem) followed by upwinding to get the normal advective velocities. This is basically the $\mathcal{R}$ operation followed by $\mathcal{U}$. Together, it is:

$$u^{\rm adv}_{i+1/2,j} = \begin{cases} u^{n+1/2}_{i+1/2,j,L} & \text{if } u^{n+1/2}_{i+1/2,j,L} > 0, \quad u^{n+1/2}_{i+1/2,j,L} + u^{n+1/2}_{i+1/2,j,R} > 0 \\ \frac{1}{2}\left(u^{n+1/2}_{i+1/2,j,L} + u^{n+1/2}_{i+1/2,j,R}\right) & \text{if } u^{n+1/2}_{i+1/2,j,L} \le 0, \quad u^{n+1/2}_{i+1/2,j,R} \ge 0 \\ u^{n+1/2}_{i+1/2,j,R} & \text{otherwise} \end{cases} \quad (34)$$

and similar for $v^{\rm adv}_{i,j+1/2}$. These velocities are sometimes referred to as the MAC velocities.

3.2 MAC projection

We could simply use these time-centered advective velocities to construct the fluxes through the interfaces and update to the new time level. However BCH showed that such a method is unstable for CFL > 0.5. The fix is to enforce the velocity constraint on these advective velocities. This involves projecting the velocity field onto the space that is divergence free. This projection is usually called the MAC projection. Once the MAC-projected advective velocities are computed, we can reconstruct the interface states using this divergence-free velocity field.

The divergence of the MAC velocities is cell-centered and constructed as:

$$(DU)_{i,j} = \frac{u^{\rm adv}_{i+1/2,j} - u^{\rm adv}_{i-1/2,j}}{\Delta x} + \frac{v^{\rm adv}_{i,j+1/2} - v^{\rm adv}_{i,j-1/2}}{\Delta y} \quad (35)$$

We define a cell-centered ϕ. $G\phi$ will then be edge-centered on a MAC grid, and $L\phi = DG\phi$ is again cell-centered. Since $L = DG$, this makes the MAC projection an exact projection.

We solve

$$L\phi = DU \quad (36)$$

using multigrid V-cycles and then update the MAC velocities as:

$$u^{\rm adv}_{i+1/2,j} = u^{\rm adv}_{i+1/2,j} - \frac{\phi_{i+1,j} - \phi_{i,j}}{\Delta x} \tag{37}$$

$$v^{\rm adv}_{i,j+1/2} = v^{\rm adv}_{i,j+1/2} - \frac{\phi_{i,j+1} - \phi_{i,j}}{\Delta y} \tag{38}$$

3.3 Reconstruct interface states

Next we redo the construction of the interface states. This procedure is identical to that above—construct the interface states $\hat{u}_{L,R}$, $\hat{v}_{L,R}$ on all edges, upwind based on $\hat{u}^{\rm adv}$, $\hat{v}^{\rm adv}$, and use these to construct the full states (including transverse terms). Now however, we construct the interface states of u and v on both the x and y-interfaces (not just the normal component at each interface). Finally, instead of solving a Riemann problem to resolve the left and right states, we simply upwind using the MAC-projected $u^{\rm adv}$ and $v^{\rm adv}$. This results in the interface state $u^{n+1/2}_{i+1/2,j}$, $v^{n+1/2}_{i+1/2,j}$, $u^{n+1/2}_{i,j+1/2}$, $v^{n+1/2}_{i,j+1/2}$.

The only reason we need to do this step over, instead of using the interface states that we predicted previously is we want to ensure that they are consistent with the MAC-projected advective velocities (and therefore, consistent with the constraint).

3.4 Provisional update

Once we have the time-centered interface states that are consistent with the MAC-projected advective velocities, we can update the velocities to the new time by discretizing the advective terms ($U \cdot \nabla U$). We express the advective terms for u as $A^{(u),n+1/2}_{i,j}$ and those for v as $A^{(v),n+1/2}_{i,j}$. These have the form:

$$\begin{aligned} A^{(u),n+1/2}_{i,j} = \quad & \frac{1}{2}\left(u^{\rm adv}_{i-1/2,j} + u^{\rm adv}_{i+1/2,j}\right)\frac{u^{n+1/2}_{i+1/2,j} - u^{n+1/2}_{i-1/2,j}}{\Delta x} + \\ & \frac{1}{2}\left(v^{\rm adv}_{i,j-1/2} + v^{\rm adv}_{i,j+1/2}\right)\frac{u^{n+1/2}_{i,j+1/2} - u^{n+1/2}_{i,j-1/2}}{\Delta y} \end{aligned} \tag{39}$$

$$\begin{aligned} A^{(v),n+1/2}_{i,j} = \quad & \frac{1}{2}\left(u^{\rm adv}_{i-1/2,j} + u^{\rm adv}_{i+1/2,j}\right)\frac{v^{n+1/2}_{i+1/2,j} - v^{n+1/2}_{i-1/2,j}}{\Delta x} + \\ & \frac{1}{2}\left(v^{\rm adv}_{i,j-1/2} + v^{\rm adv}_{i,j+1/2}\right)\frac{v^{n+1/2}_{i,j+1/2} - v^{n+1/2}_{i,j-1/2}}{\Delta y} \end{aligned} \tag{40}$$

The normal update for u, v would include the Gp term and appear as:

$$u^{\star}_{i,j} = u^{n}_{i,j} - \Delta t A^{(u),n+1/2}_{i,j} - \Delta t (Gp)^{(x),n-1/2}_{i,j} \tag{41}$$

$$v^{\star}_{i,j} = v^{n}_{i,j} - \Delta t A^{(v),n+1/2}_{i,j} - \Delta t (Gp)^{(y),n-1/2}_{i,j} \tag{42}$$

Note that at this point, we don't have an updated p, so we use a lagged value from the previous step's projection.

Alternately, we can note that for an exact projection, Gp, is the gradient of a scalar and would be removed by the projection, so we can omit it in this update, giving an alternate provisional update:

$$u^{\star\star}_{i,j} = u^{n}_{i,j} - \Delta t A^{(u),n+1/2}_{i,j} \tag{43}$$

$$v^{\star\star}_{i,j} = v^{n}_{i,j} - \Delta t A^{(v),n+1/2}_{i,j} \tag{44}$$

Following the notation in Martin, we distinguish between these with an '$\star$' vs. '$\star\star$' [1].

[1] Note that these are identical to ABC [1] approximation projections (1) and (2) (a quick look at ABC might suggest the opposite, but note that their definition of $U^{\star}$ already includes a $-Gp^{n-1/2}$ term, so by explicitly adding it back in, you are dealing with the case where $U^{\star}$ was updated without any $Gp^{n-1/2}$ term, like the '$\star\star$' case above.)

3.5 Approximate projection

This provisional velocity field does not yet obey the constraint. To enforce the constraint, we need to do a projection. Here is where we have the flexibility on whether to include the $Gp^{n-1/2}$ term. If we were doing an exact projection, then adding the gradient of a scalar would not change the divergence-free velocity field, so there would be no need to add it.

BCH did an exact projection on a cell-centered grid. There, the divergence operator is:

$$(DU)_{i,j} = \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y} \tag{45}$$

This gives a cell-centered DU. If we want ϕ cell-centered, then the gradient, $G\phi$ must also be cell centered so $L\phi = DG\phi$ is cell-centered. This means that we must have

$$(G\phi)^{(x)}_{i,j} = \frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta x} \tag{46}$$

$$(G\phi)^{(y)}_{i,j} = \frac{\phi_{i,j+1} - \phi_{i,j+1}}{2\Delta y} \tag{47}$$

The resulting Laplacian would then be a 5-point stencil that skips over the immediate neighbors:

$$(L\phi)_{i,j} = \frac{\phi_{i+2,j} - 2\phi_{i,j} + \phi_{i-2,j}}{4\Delta x^2} + \frac{\phi_{i,j+2} - 2\phi_{i,j} + \phi_{i,j-2}}{4\Delta y^2} \tag{48}$$

This decomposes the domain into 4 distinct grids that are only linked together at the boundaries. While an exact projection, this decoupling can be undesirable.

Approximate projections relax the idea that $L = DG$. In an exact projection, when you apply the projection operator, P, in succession, the result is unchanged ($P^2 = P$). This is not the case for an approximate projection. As a result, exactly what form you project matters. For an approximate projection, we can use the standard 5-point stencil for the Laplacian,

$$(L\phi)_{i,j} = \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{\Delta x^2} + \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{\Delta y^2} \tag{49}$$

together with the cell-centered divergence above (Eq. 45).

Rider [12] and Almgren, Bell, and Crutchfield (ABC) [1] explore various forms of what to project when doing the approximate projection. For instance, do we include the $Gp^{n-1/2}$ term in the provisional velocity or not? Chorin noted that if viscosity is being modeled, then it is necessary to include it here to get second-order accuracy. Also, one can project the update to the velocity, $(U^\star - U^n)/\Delta t$ instead of just the new velocity, since U^n should already be divergence free. Rider argues that projecting the update is not desirable with approximate projections, since any error in U^n being divergence-free is carried forward to the new U^{n+1}. One issue is that a cell-centered approximate projection cannot remove all sources of divergence (see Rider and Martin's Phd thesis).

When projecting the new velocity, we scale by Δt to get a quantity that has dimensions of pressure. The procedure for the projection differs slightly depending on whether we project $U^\star$ or $U^{\star\star}$:

- case I: projecting $U^\star/\Delta t$.

  From the expression above, this looks like:

  $$\frac{U^\star}{\Delta t} = \frac{U^n}{\Delta t} - A^{(u),n+1/2} - (Gp)^{(x),n-1/2} \tag{50}$$

  Ideally, U^n is already divergence free, and $Gp^{n-1/2}$ is the gradient of a scalar, which will be removed, so the projection should pick out the divergence free portion of $A^{(u)}$. We solve:

  $$L\phi = D(U^\star/\Delta t) \tag{51}$$

using multigrid V-cycles. We then find the new, divergence free velocity field as:

$$U^{n+1} = U^\star - \Delta t G\phi \tag{52}$$

Since we already included $Gp^{n-1/2}$ in what we projected, $G\phi$ will be the correction,

$$G\phi = Gp^{n+1/2} - Gp^{n-1/2} \tag{53}$$

or

$$Gp^{n+1/2} = Gp^{n-1/2} + G\phi \tag{54}$$

(see Martin 2.5 or ABC). We store this for the next timestep.

- case II: projecting $U^{\star\star}/\Delta t$.

  From the expression above, this looks like:

$$\frac{U^{\star\star}}{\Delta t} = \frac{U^n}{\Delta t} - A^{(u),n+1/2} \tag{55}$$

  There is no explicit $Gp^{n-1/2}$ term. We solve:

$$L\phi = D(U^{\star\star}/\Delta t) \tag{56}$$

  using multigrid V-cycles. We then find the new, divergence free velocity field as:

$$U^{n+1} = U^{\star\star} - \Delta t G\phi \tag{57}$$

  Since there was no $Gp^{n-1/2}$ in what we projected, $p^{n+1/2} = \phi$, and

$$Gp^{n+1/2} = G\phi \tag{58}$$

  We store this for the next timestep.

One pathology of this form of the projection is that $(DU)_{i,j}$ does not actually make use of the velocity field in zone (i, j). This decoupling from the local zone can result in a checkerboarding pattern in the projected velocity field.

4 Boundary conditions

For the advection portion of the algorithm, the boundary conditions on u and v are implemented in the usual way, using ghost cells. For the projection,

For a periodic domain, the boundary conditions on ϕ are likewise periodic. At a solid wall or inflow boundary, we already predicted the velocity that we want at the wall (in the advection step), and we do not want this value to change in the projection step. Since the correction is:

$$U^{n+1} = U^\star - \nabla\phi \tag{59}$$

we want $\nabla\phi \cdot n = 0$.

At outflow boundaries, we do not want to introduce any shear as we go through the boundary. This means that we do not want any tangential acceleration. Setting $\phi = 0$ on the boundary enforces $\nabla\phi \cdot t = 0$, where t is the unit vector tangential to the boundary.

See ABS for a discussion of the boundary conditions.

5 Bootstrapping

At step 0, we do not have a value of $Gp^{-1/2}$. To get an initial value for Gp, we run through the entire evolution algorithm starting with the initial data. At the end of a step, we reset u and v to the initial values and store the Gp at the end of this step as $Gp^{-1/2}$.

It is also common to precede this initialization by first projecting the velocity field to ensure it is divergence free. This way, we do not have to rely on the initial conditions to always set a divergence free velocity field.

6 Test problems

6.1 Convergence test

Minion introduced a simple test problem with an analytic solution. The velocity field is initialized as:

$$u(x,y) = 1 - 2\cos(2\pi x)\sin(2\pi y) \tag{60}$$
$$v(x,y) = 1 + 2\sin(2\pi x)\cos(2\pi y) \tag{61}$$

The exact solution at some time t is:

$$u(x,y,t) = 1 - 2\cos(2\pi(x-t))\sin(2\pi(y-t)) \tag{62}$$
$$v(x,y,t) = 1 + 2\sin(2\pi(x-t))\cos(2\pi(y-t)) \tag{63}$$

Minion also gives the pressure, but this is not needed for the solution. This is run on a doubly-periodic unit square domain. The main utility of this set of initial conditions is that we can use the analytic solution to measure the convergence behavior of the algorithm.

7 Extensions

- *Variable density incompressible*: Bell & Marcus [6] describe how to extend these methods to variable density flows. This means that the density in the domain may not be constant, but within a fluid element, the density does not change. This can arise, for instance, in modeling the Rayleigh-Taylor instability.

  The basic idea follows the method described above. Now the mass continuity equation is also evolved:

  $$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho U) = 0 \tag{64}$$

  The density is predicted to the interfaces follow the same procedure above and upwinded using the MAC velocities. For the projection, the decomposition is written as:

  $$U = U^d + \frac{1}{\rho}\nabla\phi \tag{65}$$

  and the elliptic equation is now a variable-coefficient equation:

  $$\nabla \cdot \frac{1}{\rho}\nabla\phi = \nabla \cdot U \tag{66}$$

- *Viscosity*: When viscosity is included in the system, our momentum equation becomes:

  $$U_t + U \cdot \nabla U + \nabla p = \epsilon \nabla^2 U \tag{67}$$

The solution process for this equation is largely the same. Following BCH, first the advective term is computed by predicting the velocities to the interfaces, doing the MAC projection, and then forming $A^{n+1/2}$. Now there is an explicit viscosity term (at time-level n) present in the prediction, as a source term. The provision velocity update is no longer a simple flux update, but instead requires solving two decoupled diffusion-like equations (one for u and one for v). These are differenced using Crank-Nicolson centering:

$$\frac{u^\star - u^n}{\Delta t} = -A^{(u),n+1/2} - \nabla p^{(x),n-1/2} + \frac{\epsilon}{2}\nabla^2(u^n + u^\star) \quad (68)$$

$$\frac{v^\star - v^n}{\Delta t} = -A^{(v),n+1/2} - \nabla p^{(y),n-1/2} + \frac{\epsilon}{2}\nabla^2(v^n + v^\star) \quad (69)$$

This involves two separate multigrid solves. Once $U^\star$ is found, the final projection is done as usual.

- *Low Mach number combustion*: In low-Mach number combustion flows, the fluid is allowed to respond to local heat release by expanding. The velocity constraint is derived by differentiating the equation of state along particle paths, leading to the appearance of a source term:

$$\nabla \cdot U = S \quad (70)$$

Here, S, incorporates the compressibility effects due to the heat release and diffusion. This system is used when modeling burning fronts (flames). This type of flow can be thought of as linking two incompressible states (the fuel and the ash) by the expansion across the interface.

The solution technique largely follows that of the incompressible flow. One caveat, because the constraint now has a local heat source, S, doing the cell-centered divergence described above leads to a decoupling of DU from the local source, since the stencil of DU does not include the zone upon which it is centered.

This system is described in detail in [11, 9, 5].

- *Nodal projection*: instead of discretizing the final projection using a cell-centered ϕ, ABS use a node-centered ϕ. While this is still an approximate projection, this discretization couples in the zone we are centered on, and is said to be able to do a better job removing pathological divergent velocity fields that cell-centered projections stumble on.

References

[1] A. S. Almgren, J. B. Bell, and W. Y. Crutchfield. Approximate projection methods: Part I. Inviscid analysis. *SIAM J. Sci. Comput.*, 22(4):1139–59, 2000.

[2] A. S. Almgren, J. B. Bell, and W. G. Szymczak. A numerical method for the incompressible Navier-Stokes equations based on an approximate projection. *SIAM J. Sci. Comput.*, 17(2):358–369, March 1996.

[3] J. B. Bell, P. Colella, and H. M. Glaz. A Second Order Projection Method for the Incompressible Navier-Stokes Equations. *Journal of Computational Physics*, 85:257, December 1989.

[4] J. B. Bell, P. Colella, and L. H. Howell. An efficient second-order projection method for viscous incompressible flow. In *Proceedings of the Tenth AIAA Computational Fluid Dynamics Conference*, pages 360–367. AIAA, June 1991. see also: https://seesar.lbl.gov/anag/publications/colella/A_2_10.pdf.

[5] J. B. Bell, M. S. Day, C. A. Rendleman, S. E. Woosley, and M. A. Zingale. Adaptive low Mach number simulations of nuclear flame microphysics. *Journal of Computational Physics*, 195(2):677–694, 2004.

[6] J. B. Bell and D. L. Marcus. A second-order projection method for variable-density flows. *Journal of Computational Physics*, 101(2):334 – 348, 1992.

[7] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Math. Comp.*, 22:745–762, 1968.

[8] P. Colella. Multidimensional upwind methods for hyperbolic conservation laws. *Journal of Computational Physics*, 87:171–200, March 1990.

[9] M. S. Day and J. B. Bell. Numerical simulation of laminar reacting flows with complex chemistry. *Combust. Theory Modelling*, 4(4):535–556, 2000.

[10] D. F. Martin and P. Colella. A Cell-Centered Adaptive Projection Method for the Incompressible Euler Equations. *Journal of Computational Physics*, 163:271–312, September 2000.

[11] 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. *Comb. Sci. Tech.*, 140:123–168, 1998.

[12] W. J. Rider. Approximate projection methods for incompressible flow: Implementation, variants and robustness. Technical report, LANL UNCLASSIFIED REPORT LA-UR-94-2000, LOS ALAMOS NATIONAL LABORATORY, 1995.

[13] E. F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer, 1997.