A Conservative Adaptive Projection Method for the Variable Density Incompressible Navier-Stokes Equations *

Ann S. Almgren
John B. Bell
Phillip Colella
Louis H. Howell
Michael L. Welcome

Lawrence Berkeley National Laboratory
Berkeley, CA 94720

Abstract

In this paper we present a method for solving the equations governing time-dependent, variable density incompressible flow in two or three dimensions on an adaptive hierarchy of grids. The method is based on a projection formulation in which we first solve advection-diffusion equations to predict intermediate velocities, and then project these velocities onto a space of approximately divergence-free vector fields. Our treatment of the first step uses a specialized second-order upwind method for differencing the nonlinear convection terms that provides a robust treatment of these terms suitable for inviscid and high Reynolds number flow. Density and other scalars are advected in such a way as to maintain conservation, if appropriate, and free-stream preservation.

Our approach to adaptive refinement uses a nested hierarchy of logically-rectangular grids with simultaneous refinement of the grids in both space and time. The integration algorithm on the grid hierarchy is a recursive procedure in which coarse grids are advanced in time, fine grids are advanced multiple steps to reach the same time as the coarse grids and the data at different levels are then synchronized. The single grid algorithm is described briefly, but the emphasis here is on the time-stepping procedure for the adaptive hierarchy. Numerical examples are presented to demonstrate the algorithm's accuracy and convergence properties, and illustrate the behavior of the method. An additional example demonstrates the performance of the method on a more realistic problem, namely, a three-dimensional variable density shear layer.

*Support for this work was provided by the Applied Mathematical Sciences Program and the HPCC Grand Challenge Program of the DOE Office of Mathematics, Information, and Computational Sciences, and by the Defense Nuclear Agency under IACRO 95-2045, under contract DE-AC03-76SF00098.

1 Introduction

In this paper we develop a local adaptive mesh refinement algorithm for variable density, constant viscosity, incompressible flow based on a second-order projection method. The equations governing this flow are:

$$U_t + (U \cdot \nabla)U = \frac{1}{\rho}(-\nabla p + \mu \nabla^2 U + H_U), \tag{1}$$

$$\rho_t + \nabla \cdot (\rho U) = 0, \tag{2}$$

$$c_t + (U \cdot \nabla)c = k\nabla^2 c + H_c, \tag{3}$$

$$\nabla \cdot U = 0 \quad , \tag{4}$$

where $U = (u, v, w)$, ρ, c, and p represent the velocity, density, concentration of an advected scalar, and pressure, respectively, and $H_U = (H_x, H_y, H_z)$ represents any external forces. Here μ is the dynamic viscosity coefficient, k is the diffusive coefficient for c, and H_c is the source term for c. In general one could advect an arbitrary number of scalars, either passively or conservatively.

The development of the single grid second-order projection methodology for the incompressible Navier-Stokes equations is discussed in a series of papers by Bell, Colella and Glaz [6], Bell, Colella and Howell [7], and Almgren, Bell and Szymczak [4]. The method discussed here is an adaptive version of the algorithm presented by Almgren et al. [4], generalized to include finite amplitude density variation as originally discussed in Bell and Marcus [8]. The details of the single grid algorithm are discussed in Puckett et al. [23]. The basic methodology presented in those papers was motivated by a desire to apply higher-order upwind methods developed for hyperbolic conservation laws to incompressible flow. In particular, they use a specialized version of the unsplit second-order upwind methodology for the convective terms in Eqs. (1)–(3) that was introduced for gas dynamics by Colella [16]. The upwind methodology provides a robust discretization of the convective terms that avoids any stability restriction other than the CFL constraint for inviscid flow.

The focus of this paper is on developing a local adaptive mesh refinement (AMR) version of the basic projection methodology. This algorithm uses a hierarchical structured grid approach first developed by Berger and Oliger [10] for hyperbolic partial differential equations. In particular, AMR is based on a sequence of nested grids with successively finer spacing in both time and space. Increasingly finer grids are recursively embedded in coarse grids until the solution is sufficiently resolved. An error estimation procedure based on user-specified criteria evaluates where additional refinement is needed and grid generation procedures dynamically create or remove rectangular fine grid patches as resolution requirements change.

The approach to adaptive gridding used here has been demonstrated to be highly successful for gas dynamics by Berger and Colella [9] in two dimensions and by Bell et al. [5] in three dimensions. Steinthorsen et al. [27] generalized this approach to the compressible Navier-Stokes equations in two dimensions. Skamarock and Klemp [26] have successfully implemented an adaptive scheme with subcycling in time for the compressible formulation of the equations governing atmospheric flows.

For incompressible flow, Howell and Bell [17] presented a two-dimensional nonconservative adaptive algorithm, based on the Bell, Colella and Glaz projection formulation, which did not subcycle in time. This version used an exact projection which introduced substantial complication at coarse/fine boundaries because of local decoupling of the projection.

Minion [21] has developed an adaptive projection method for the two-dimensional incompressible Euler equations with constant density on locally refined grids. In this approach all grid levels are advanced with the same time step which is determined by the data at the finest level. Minion uses the treatment of the convection terms discussed in Bell, Colella and Howell [7] in which a MAC projection is used as an intermediate step in the convection algorithm in order to enforce incompressibility at the half-time level. He also uses an approximate cell-centered projection based on the MAC projection to enforce the divergence constraint at the end of the time step.

Almgren et al. [1, 3] developed a two-dimensional, variable density adaptive version of the approximate projection formulation developed by Almgren, Bell and Szymczak [4]. The methodology presented in these papers used nonconservative difference approximations of the convective terms, and did not incorporate an intermediate MAC projection. Since the treatment of convection was nonconservative a simplified synchronization between levels of refinement was used. Almgren et al. [2] present a generalization of this approach to three dimensions.

Clark and Farley [14] and Stevens [28, 29] present methods for solving the anelastic formulation of the equations governing the atmosphere on an adaptive hierarchy of grids. (The anelastic equations are analogous to the incompressible Navier-Stokes equations but with a different constraint, namely, $\nabla \cdot (\rho_0(z)U) = 0$, where ρ_0 is a given function of altitude that represents atmospheric stratification.) In [14], there is no temporal refinement; in [28, 29], an adaptive projection method is used with subcycling in time. Both of the above algorithms use a staggered representation of velocities, with arbitrary integer factors of refinement and different types of difference approximations than presented here.

In both methods there is two-way nesting, in that coarse grid data are used as boundary conditions for fine grid operations, then fine grid data are averaged down onto coarse grid data at the end of a time step. However, there is no elliptic synchronization step to enforce both the Dirichlet and Neumann matching conditions for the elliptic pressure solve at the coarse/fine interface. As documented in [29] and this paper, there is a loss of accuracy associated with not satisfying both matching conditions.

There are also a number of adaptive algorithms for incompressible flow based on an unstructured grid approach. The reader is referred to Ramamurti, Löhner and Sandberg [25], Ramamurti, Sandberg and Löhner [24] and the references cited therein for some discussion of this approach.

The methodology presented here is based on the approximate projection algorithm developed in Almgren, Bell and Szymczak [4]. The goal of this work is not simply to develop an adaptive algorithm for the incompressible Navier-Stokes equations, but to provide the basis for adaptive algorithms for more general low Mach number flow models. Examples of these types of models include the anelastic equations for atmospheric flow, with modules for moisture physics and radiation, and low speed combustion models, where the divergence constraint is inhomogeneous, with modules for reaction kinetics and thermal radiation and conduction. The characteristics of these more general low Mach number flows suggest additional desirable features for the flow algorithm which have influenced the design presented here. First, the nonlinearity of the additional physics modules, particularly as related to exothermic reactions, make conservation of advected species an important consideration. Second, the additional computational requirements of these more general models may require that not all regions of interest are refined to the finest available level. For this reason, we want the method to perform as well as possible on coarser levels.

As in our previous work [1, 3, 2] the method presented here uses subcycling in time; this allows all levels to be advanced at the same CFL number, where the performance of upwind advection algorithms is optimal. However, unlike earlier versions of the adaptive algorithm, we now use an intermediate MAC projection so that the advection velocity used in evaluating the convective terms in Eqs. (1)-(3) satisfies the divergence constraint (Eq. (4)). This permits conservative differencing to be used to advance advected quantities which guarantees conservation on each grid individually. In addition, we have paid special attention to the synchronization step of the algorithm so that the overall method is conservative for density (and other conservatively differenced scalar fields) and free-stream-preserving in the sense that constant scalar fields with no source terms remain constant independent of grid refinement patterns and the velocity field.

Before describing the adaptive algorithm we will briefly review the basic fractional step scheme for a single grid. In the third section we describe, in detail, the recursive time-stepping procedure for the adaptive algorithm and other aspects of the adaptive algorithm. The fourth section shows convergence results and presents computational examples illustrating the performance of the method.

2 Single Grid Projection Algorithm

In this section we review the basic fractional step scheme for the case of a single uniform grid. The reader is referred to [4] and [6] for a more detailed description. In this algorithm, velocity, density, and concentration are defined at cell centers at integer times and are denoted by $U^n_{i,j,k}$, $\rho^n_{i,j,k}$ and $c^n_{i,j,k}$, respectively. Pressure is specified at cell corners and is staggered in time; thus, pressure is denoted by $p^{n+1/2}_{i+1/2,j+1/2,k+1/2}$.

2.1 Advection-Diffusion Step

In the first step of the fractional step scheme, we solve the advection-diffusion equations Eqs. (2)–(3) for the updated density and concentration, and we compute an intermediate velocity field from Eq. (1) without strictly enforcing the divergence constraint on velocity. In the second step, we project this intermediate field onto the space of vector fields which approximately satisfy the divergence constraint.

For the advection-diffusion step we solve the conservative forms of Eqs. (1)–(3). This leads to a natural definition of face fluxes that are used to handle refluxing across coarse/fine grid boundaries in the adaptive algorithm. In particular, we solve

$$\frac{U^* - U^n}{\Delta t} = -[\nabla \cdot (UU)]^{n+1/2} + \frac{1}{\rho^{n+1/2}}(-\nabla p^{n-1/2} + \frac{\mu}{2}(\nabla^2 U^n + \nabla^2 U^*) + H_U^{n+1/2}), \quad (5)$$

$$\frac{\rho^{n+1} - \rho^n}{\Delta t} = -[\nabla \cdot (\rho U)]^{n+1/2} \tag{6}$$

and

$$\frac{c^{n+1} - c^n}{\Delta t} = -[\nabla \cdot (cU)]^{n+1/2} + H_c^{n+1/2} + \frac{k}{2}(\nabla^2 c^n + \nabla^2 c^{n+1}) \tag{7}$$

for the intermediate velocity U^* and the updated density ρ^{n+1} and concentration c^{n+1}. We note here that the same conservative discretization is used to represent convective and conservative differences because the advection velocities are discretely divergence-free. (This equivalence would not be true in the more general low Mach number case.) The method uses an unsplit second-order upwind predictor-corrector scheme for evaluating the advective derivatives in Eqs. (5)–(7). For this step the pressure gradient is evaluated at $t^{n-1/2}$ and is treated as a source term in Eq. (5), with $\rho^{n+1/2} \equiv \frac{1}{2}(\rho^n + \rho^{n+1})$. The forcing term H_U in the momentum equation and the source term H_c in the concentration equation are centered in time to preserve second-order accuracy.

In the predictor we first extrapolate the normal velocities to cell faces at $t^{n+1/2}$ using a second-order Taylor series expansion in space and time. The time derivative is replaced using Eq. (1). For face $(i + \frac{1}{2}, j, k)$ this gives

$$\begin{aligned} \tilde{u}^{L,n+1/2}_{i+1/2,j,k} &\approx u^n_{i,j,k} + \frac{\Delta x}{2} u_x + \frac{\Delta t}{2} u_t \\ &= u^n_{i,j,k} + (\frac{\Delta x}{2} - u^n_{i,j,k}\frac{\Delta t}{2})(u_x^{n,lim})_{i,j,k} + \frac{\Delta t}{2}(-(\widehat{vu_y})_{i,j,k} - (\widehat{wu_z})_{i,j,k} + \\ &\quad \frac{1}{\rho^n_{i,j,k}}(-(G_x p)^{n-\frac{1}{2}}_{i,j,k} + \mu \Delta^h u^n_{i,j,k} + H^n_{U,x,i,j,k})). \end{aligned} \tag{8}$$

extrapolated from (i, j, k), and

$$\begin{aligned} \tilde{u}^{R,n+1/2}_{i+1/2,j,k} &\approx u^n_{i+1,j,k} - \frac{\Delta x}{2} u_x + \frac{\Delta t}{2} u_t \\ &= u^n_{i+1,j,k} - (\frac{\Delta x}{2} + u^n_{i+1,j,k}\frac{\Delta t}{2})(u_x^{n,lim})_{i+1,j,k} + \frac{\Delta t}{2}(-(\widehat{vu_y})_{i+1,j,k} - (\widehat{wu_z})_{i+1,j,k} + \\ &\quad \frac{1}{\rho^n_{i+1,j,k}}(-(G_x p)^{n-\frac{1}{2}}_{i+1,j,k} + \mu \Delta^h u^n_{i+1,j,k} + H^n_{U,x,i+1,j,k})) \end{aligned} \tag{9}$$

extrapolated from $(i + 1, j, k)$. Here, Δ^h is a standard, five-point in 2-d, seven-point in 3-d, cell-centered approximation to the Laplacian and $G = (G_x, G_y, G_z)$ is a discretization of the gradient operator which defines a cell-centered gradient from a node-based pressure field.

Analogous formulae are used to predict values for $\tilde{v}^{F/B,n+1/2}_{i,j+1/2,k}$ and $\tilde{w}^{D/U,n+1/2}_{i,j,k+1/2}$ at the other faces of the cell. In evaluating these terms the first derivatives normal to the face (in this case $u_x^{n,lim}$) are evaluated using a monotonicity-limited fourth-order slope approximation [15]. The limiting is done on each component of the velocity at time n individually.

The transverse derivative terms ($\widehat{vu_y}$ and $\widehat{wu_z}$ in this case) are evaluated by first extrapolating all velocity components to the transverse faces from the cell centers on either side, then choosing between these states using the upwinding procedure defined below. In particular, in the y direction we define

$$\hat{U}^F_{i,j+1/2,k} = U^n_{i,j,k} + \left(\frac{\Delta y}{2} - \frac{\Delta t}{2} v^n_{i,j,k}\right)(U_y^{n,lim})_{i,j,k} \quad , \tag{10}$$

$$\hat{U}^B_{i,j+1/2,k} = U^n_{i,j+1,k} - \left(\frac{\Delta y}{2} + \frac{\Delta t}{2} v^n_{i,j+1,k}\right)(U_y^{n,lim})_{i,j+1,k} \quad . \tag{11}$$

Values are similarly traced from (i,j,k) and $(i,j,k+1)$ to the $(i,j,k+1/2)$ faces to define $\hat{U}^D_{i,j,k+1/2}$ and $\hat{U}^U_{i,j,k+1/2}$, respectively.

In this upwinding procedure we first define a normal advective velocity on the face (suppressing the $(i,j+\frac{1}{2},k)$ spatial indices on front and back states here and in the next equation):

$$\widehat{v}^{adv}_{i,j+\frac{1}{2},k} = \begin{cases} \widehat{v}^F & \text{if } \widehat{v}^F > 0, \;\; \widehat{v}^F + \widehat{v}^B > 0 \\ 0 & \text{if } \widehat{v}^F \leq 0, \widehat{v}^B \geq 0 \text{ or } \widehat{v}^F + \widehat{v}^B = 0 \\ \widehat{v}^B & \text{if } \widehat{v}^B < 0, \;\; \widehat{v}^F + \widehat{v}^B < 0 \; . \end{cases}$$

We now upwind $\widehat{U}$ based on $\widehat{v}^{adv}_{i,j+\frac{1}{2},k}$:

$$\widehat{U}_{i,j+\frac{1}{2},k} = \begin{cases} \widehat{U}^F & \text{if } \widehat{v}^{adv}_{i,j+\frac{1}{2},k} > 0 \\ 1/2(\widehat{U}^F + \widehat{U}^B) & \text{if } \widehat{v}^{adv}_{i,j+\frac{1}{2},k} = 0 \\ \widehat{U}^B & \text{if } \widehat{v}^{adv}_{i,j+\frac{1}{2},k} < 0 \end{cases}$$

After constructing $\widehat{U}_{i,j-\frac{1}{2},k}$, $\widehat{U}_{i,j,k+1/2}$ and $\widehat{U}_{i,j,k-1/2}$ in a similar manner, we use these upwind values to form the transverse derivatives in Eqs. (8) and (9):

$$(\widehat{vu_y})_{i,j,k} = \frac{1}{2\Delta y}(\widehat{v}^{adv}_{i,j+\frac{1}{2},k} + \widehat{v}^{adv}_{i,j-\frac{1}{2},k})(\widehat{u}_{i,j+\frac{1}{2},k} - \widehat{u}_{i,j-\frac{1}{2},k})$$

$$(\widehat{wu_z})_{i,j,k} = \frac{1}{2\Delta z}(\widehat{w}^{adv}_{i,j,k+1/2} + \widehat{w}^{adv}_{i,j,k-1/2})(\widehat{u}_{i,j,k+1/2} - \widehat{u}_{i,j,k-1/2}).$$

The normal velocity at each face is then determined by an upwinding procedure based on the states predicted from the cell centers on either side. The procedure is similar to that described above, i.e. (suppressing the $(i+1/2,j,k)$ indices)

$$\tilde{u}^{n+1/2}_{i+\frac{1}{2},j,k} = \begin{cases} \tilde{u}^{L,n+1/2} & \text{if } \tilde{u}^{L,n+1/2} > 0 \text{ and } \tilde{u}^{L,n+1/2} + \tilde{u}^{R,n+1/2} > 0 \\ 0 & \text{if } \tilde{u}^{L,n+1/2} \leq 0, \tilde{u}^{R,n+1/2} \geq 0 \text{ or } \tilde{u}^{L,n+1/2} + \tilde{u}^{R,n+1/2} = 0 \\ \tilde{u}^{R,n+1/2} & \text{if } \tilde{u}^{R,n+1/2} < 0 \text{ and } \tilde{u}^{L,n+1/2} + \tilde{u}^{R,n+1/2} < 0 \end{cases}$$

We follow a similar procedure to construct $\tilde{v}^{n+1/2}_{i,j+1/2,k}$ and $\tilde{w}^{n+1/2}_{i,j,k+1/2}$.

The normal velocities on cell faces are now centered in time and second-order accurate, but do not, in general, satisfy the divergence constraint. In order to enforce the constraint at this intermediate time, we apply the MAC projection (see [7]) to the face-based velocity field before construction of the conservative updates. The equation

$$D^{E\to C}(\frac{1}{\rho^n}G^{C\to E}\phi^{MAC}) = D^{E\to C}(\frac{\tilde{U}^{n+1/2}}{\Delta t/2}) \tag{12}$$

is solved for ϕ^{MAC}, with homogeneous Neumann boundary conditions on all physical boundaries except for outflow, where ϕ^{MAC} is set to zero to enforce the "no tangential acceleration" criterion. Here

$$D^{E\to C}(\frac{\tilde{U}^{n+1/2}}{\Delta t/2}) \equiv \frac{1}{\Delta t/2}\frac{\tilde{u}^{n+1/2}_{i+\frac{1}{2},j,k} - \tilde{u}^{n+1/2}_{i-\frac{1}{2},j,k}}{\Delta x} + \frac{\tilde{v}^{n+1/2}_{i,j+\frac{1}{2},k} - \tilde{v}^{n+1/2}_{i,j-\frac{1}{2},k}}{\Delta y} + \frac{\tilde{w}^{n+1/2}_{i,j,k+\frac{1}{2}} - \tilde{w}^{n+1/2}_{i,j,k-\frac{1}{2}}}{\Delta z}$$

and $G^{C\to E} = -(D^{E\to C})^T$ so that

$$(G_x^{C\to E}\phi^{MAC})_{i+1/2,j,k} = \frac{(\phi^{MAC}_{i+1,j,k} - \phi^{MAC}_{i,j,k})}{\Delta x}$$

with $G_y^{C\to E}$ and $G_z^{C\to E}$ defined analogously.

(In axisymmetric coordinates, $D^{E\to C}$ would be defined by

$$D^{E\to C}(\frac{\tilde{U}^{n+1/2}}{\Delta t/2}) \equiv \frac{1}{\Delta t/2}\frac{(r\tilde{u}^{n+1/2})_{i+1/2,j} - (r\tilde{u}^{n+1/2})_{i-1/2,j}}{r_i\Delta r} + \frac{\tilde{v}^{n+1/2}_{i,j+1/2} - \tilde{v}^{n+1/2}_{i,j-1/2}}{\Delta z},$$

where r is the distance from the axis of symmetry. $G^{C\to E}$ would be unchanged.)

The face-based advection velocity U^{ADV} is then defined by

$$U^{ADV} = \tilde{U}^{n+1/2} - \frac{\Delta t}{2\rho^n_{i+\frac{1}{2},j,k}}(G^{C\to E}_{norm}\phi^{MAC})$$

where $G^{C\to E}_{norm}$ is the gradient operator in the normal direction to each face. Here and in Eq. (12), ρ on the faces is averaged geometrically from the cell centers at time n. (We note also that incorporating $\Delta t/2$ into Eq. (12) defines ϕ as a pressure correction, which clarifies coarse/fine boundary conditions in the adaptive algorithm.)

At this point the predictor step is performed for the tangential velocity components, density and concentration. The extrapolation of the normal velocity components from cell centers to all cell faces has been described above; the tracing of density, concentration and tangential velocity components is analogous, with the time derivatives replaced using Eqs. (1)–(3).

Now let $S = \{U, \rho, c\}$. Time-centered values $\tilde{S}^{n+1/2}$ at each face (i.e. $\tilde{\rho}^{n+1/2}$, $\tilde{c}^{n+1/2}$, and $\tilde{U}^{n+1/2}$ including the normal velocity component) are determined by upwinding, as follows:

$$\tilde{S}_{i+\frac{1}{2},j,k} = \begin{cases} \tilde{S}^L & \text{if } u^{ADV}_{i+\frac{1}{2},j,k} > 0 \\ 1/2(\tilde{S}^L + \tilde{S}^R) & \text{if } u^{ADV}_{i+\frac{1}{2},j,k} = 0 \\ \tilde{S}^R & \text{if } u^{ADV}_{i+\frac{1}{2},j,k} < 0 \end{cases}$$

We define the conservative update terms in terms of the advective fluxes, $F_S^{adv} = U^{ADV}\tilde{S}^{n+1/2}$:

$$[\nabla\cdot(SU)]^{n+1/2}_{i,j,k} = D^{E\to C}(F_S^{adv}).$$

Using this approximation we now compute ρ^{n+1} from Eq. (6)

$$\rho^{n+1} = \rho^n - \Delta t\, D^{E\to C}(F_\rho^{adv}).$$

For later convenience, we define now the viscous and diffusive fluxes corresponding to our discretization of $\Delta^h = D^{E\rightarrow C}G^{C\rightarrow E}$:

$$F_U^{visc} = F_{U^n}^{visc} + F_{U^*}^{visc} = \frac{\mu}{2}(G_{norm}^{C\rightarrow E}U^{n,\ell} + G_{norm}^{C\rightarrow E}U^{*,\ell}),$$

$$F_c^{diff} = F_{c^n}^{diff} + F_{c^{n+1}}^{diff} = \frac{k}{2}(G_{norm}^{C\rightarrow E}c^{n,\ell} + G_{norm}^{C\rightarrow E}c^{n+1,\ell}),$$

Equations (5) and (7) require solution of parabolic equations for each component of the intermediate velocity U^*

$$(1 - \frac{\mu\Delta t}{2\rho^{n+1/2}}\Delta^h)U^* = U^n - \Delta t D^{E\rightarrow C}(F_U^{adv}) + \frac{\Delta t}{\rho^{n+1/2}}(-Gp^{n-1/2} + \frac{1}{2}D^{E\rightarrow C}(F_{U^n}^{visc}) + H_U^{n+1/2}),$$

and for the concentration c^{n+1} :

$$(1 - \frac{k\Delta t}{2}\Delta^h)c^{n+1} = c^n - \Delta t D^{E\rightarrow C}(F_c^{adv}) + \frac{\Delta t}{2}D^{E\rightarrow C}(F_{c^n}^{diff}) + \Delta t H_c^{n+1/2}.$$

These parabolic solves are described in more detail in Section 4.

The upwind method is an explicit difference scheme and, as such, requires a time-step restriction for stability. We use the standard CFL condition, modified to account for the case where the initial velocity is very small (or zero) but the accelerations may be large:

$$\Delta t \le \min\left(\min_{i,j,k}\left(\frac{\Delta x}{|u_{i,j,k}|}, \frac{\Delta y}{|v_{i,j,k}|}, \frac{\Delta z}{|w_{i,j,k}|}\right), \min_{i,j,k}\sqrt{\frac{2\Delta x}{|H_{U,i,j,k} - (Gp)_{i,j,k}|/\rho_{i,j,k}}}\right).$$

We note here that since the viscous terms are not included in defining the states used in the transverse derivatives (Eqs. (10)–(11)) there is an additional stability constraint on the time step for large μ or k which can require that the maximum CFL be reduced to 1/2 (see [20]). Also, we note that in three dimensions we have not included full corner coupling in the advection algorithm; consequently, we require CFL to be less than 0.8 in three dimensions.

The velocity field U^* computed using Eq. (5) does not, in general, satisfy the divergence constraint. The projection step, as described in the next subsection, approximately enforces this constraint.

2.2 Discretization of the Projection

In the projection step, a vector field decomposition is applied to $V = (U^* - U^n)/\Delta t$ to obtain the new velocity field, U^{n+1}, and an update for the pressure. In particular, if $\mathbf{P}$ represents the projection then

$$\frac{U^{n+1} - U^n}{\Delta t} = \mathbf{P}(V) \tag{13}$$

$$\frac{1}{\rho^{n+1/2}}\nabla p^{n+1/2} = \frac{1}{\rho^{n+1/2}}\nabla p^{n-\frac{1}{2}} + (\mathbf{I} - \mathbf{P})(V).$$

Note that the vector field V we project is not U^*, it is an approximation to U_t. This distinction is significant when the projection is not exact. Discretely, the projection is computed by solving for the appropriately weighted gradient component of V which we denote by $(1/\rho)G\phi$. We determine ϕ by solving

$$L_\rho^{n+1/2}\phi = DV$$

where D is a discrete nodal approximation to the divergence operator and $L_\rho^{n+1/2}\phi$ is a second-order accurate nodal approximation to $\nabla \cdot \left(\frac{1}{\rho^{n+1/2}}\nabla\phi\right)$.

In two dimensions the projection discretization can be derived directly from the variational form

$$\int \frac{1}{\rho}\nabla\phi(\mathbf{x}) \cdot \nabla\psi(\mathbf{x})\; d\mathbf{x} = \int V \cdot \nabla\psi(\mathbf{x})\; d\mathbf{x}, \qquad \forall\psi(\mathbf{x}) \tag{14}$$

where $d\mathbf{x}$ is the volume element $dx\ dy$ or $r\ dr\ d\theta$ as appropriate. If this variational form is used in conjunction with standard piecewise bilinear or piecewise linear (on a standard triangulation of a mesh) finite element basis functions, the resulting discrete problem corresponds to standard nine-point and five-point discretizations of $L_\rho^{n+1/2}$, respectively. (In this paper we use the nine-point discretization for all two-dimensional problems.) We then define

$$\frac{U^{n+1} - U^n}{\Delta t} = V - \frac{1}{\rho^{n+1/2}}\overline{G\phi}, \tag{15}$$

where $\overline{G\phi}$ is the cell average of $\nabla\phi$, and

$$p^{n+1/2} = p^{n-1/2} + \phi.$$

We note that this is not a discrete orthogonal projection; in fact, $DU^{n+1} \neq 0$. However, the projection as defined by Eqs. (13) and (14) is a discrete orthogonal projection onto a larger velocity space (in the finite element sense) which is then averaged onto the grid. The resulting approximate projection satisfies the divergence constraint to second-order accuracy and the overall algorithm is stable. The reader is referred to Almgren et al. [4] for a detailed discussion of this approximation to the projection.

In three dimensions a twenty-seven point discretization of the projection can be derived using trilinear basis functions; however, the derivation of an analog to the five-point scheme does not extend directly. Standard approaches to dividing a cube into tetrahedra lead to directional biases in the discretization which are undesirable. Instead, to avoid the computational work associated with the twenty-seven point discretization we use a standard seven-point finite difference analog to the five-point discretization in two dimensions to approximate $L_\rho^{n+1/2}$. The details of these stencils are given in the appendix.

2.3 Initialization of the Data

Specification of the problem must include values for U, ρ and c at time $t = 0$ and a description of the boundary conditions. The pressure is not initially prescribed, and must be calculated in an initial iterative step.

To begin the calculation, the initial velocity field is first projected to ensure that it satisfies the divergence constraint at t=0. Then an initial iteration is performed to calculate an approximation to the pressure at $t = \frac{\Delta t}{2}$. If this process were iterated to convergence and the projection were exact, then $U^1 \equiv U^*$ in the first step, because the pressure used in Eq. (5) would in fact be $p^{1/2}$, not $p^{-1/2}$. However, in practice we typically perform only a few iterations, since what is needed for second-order accuracy in Eq. (5) is only a first-order accurate approximation to $p^{n+1/2}$, which in a standard time step is approximated by $p^{n-\frac{1}{2}}$.

In each step of the iteration we follow the procedure described in the above two subsections. In the first iteration we use $p^{-1/2} = 0$. At the end of each iteration we have calculated a value of U^1 and a pressure $p^{1/2}$. During the iteration procedure, we discard the value of U^1, but define $p^{-1/2} = p^{1/2}$. Once the iteration is completed, we use the value of $p^{-1/2}$ in Eq. (5) along with the values of U^0, ρ^0 and c^0.

3 Adaptive Mesh Refinement

In this section we present the extension of the algorithm described above to an adaptive hierarchy of nested rectangular grids. In the first subsection we describe the creation of the grid hierarchy and the regridding procedure used to adjust the hierarchy during the computation; in the second we describe the initialization procedure used to begin a multilevel calculation. The third and fourth subsections contain an overview of, then the details of the time step algorithm for the grid system that subcycles in time, focusing on the synchronization between different levels of refinement. In the fifth and sixth subsections we discuss the spatial discretization of the single-level and multilevel elliptic operators used in the algorithm.

3.1 Creating and Managing the Grid Hierarchy

The grid hierarchy is composed of different levels of refinement ranging from coarsest ($\ell = 0$) to finest ($\ell = \ell_{max}$). Each level is represented as the union of rectangular grid patches of a given resolution. In this implementation, the refinement ratio is always even, with the same factor of refinement in each coordinate direction, i.e. $\Delta x^{\ell+1} = \Delta y^{\ell+1} = \Delta z^{\ell+1} = \frac{1}{r}\Delta x^{\ell}$, where r is the refinement ratio. (We note here that neither isotropic refinement nor uniform base grids are requirements of the fundamental algorithm; see the section on future work.) In the actual implementation, the refinement ratio, either 2 or 4, can be a function of level; however, in the exposition we will assume that r is constant. The grids are properly nested, in the sense that the union of grids at level $\ell + 1$ is contained in the union of grids at level ℓ for $0 \leq \ell < \ell_{max}$. Furthermore, the containment is strict in the sense that, except at physical boundaries, the level ℓ grids are large enough to guarantee that there is a border at least one level ℓ cell wide surrounding each level $\ell + 1$ grid. (Grids at all levels are allowed to extend to the physical boundaries so the proper nesting is not strict there.)

The initial creation of the grid hierarchy and the subsequent regridding operations in which the grids are dynamically changed to reflect changing flow conditions use the same procedures as were used by Bell et al. [5] for hyperbolic conservation laws. The construction of the grid hierarchy is based on error estimation criteria specified by the user to indicate where additional resolution is required. The error criteria are currently based on tracking features of the flow such as vorticity or density gradients; however, more sophisticated criteria based on estimating the error can be used (see, e.g., [9]). Given grids at level ℓ we use the error estimation procedure to tag cells where the criteria for further refinement are met. The tagged cells are grouped into rectangular patches using the clustering algorithm given in Berger and Rigoustsos [11]. These rectangular patches are refined to form the grids at the next level. The process is repeated until either the error tolerance criteria are satisfied or a specified maximum level is reached. The proper nesting requirement is imposed at this stage.

At $t = 0$ the initial data is used to create grids at level 0 through ℓ_{max}. (Grids have a user-specified maximum size, therefore more than one grid may be needed to cover the physical domain.) As the solution advances in time, the regridding algorithm is called every k_ℓ (also user-specified) level ℓ steps to redefine grids at levels $\ell+1$ to ℓ_{max}. Level 0 grids remain unchanged throughout the calculation. Grids at level $\ell+1$ are only modified at the end of level ℓ time steps, but because we subcycle in time, i.e. $\Delta t_{\ell+1} = \frac{1}{r}\Delta t_\ell$, level $\ell+2$ grids can be created and/or modified in the middle of a level ℓ time step if $k_{\ell+1} < r$.

When new grids are created at level $\ell+1$, the data on these new grids are copied from the previous grids at level $\ell+1$ if possible, otherwise interpolated in space from the underlying level ℓ grids.

We note here that while there is a user-specified limit to the number of levels allowed, at any given time in the calculation there may not be that many levels in the hierarchy, i.e. ℓ_{max} can change dynamically as the calculation proceeds, as long as it does not exceed the user-specified limit.

3.2 Initialization of the Multilevel Data

As in the single grid projection method, we must first project the given velocity field to approximately enforce the divergence constraint, and iterate with the initial data in order to define an initial pressure field. For accuracy, the initial projection is done as a full multilevel composite solve over all levels as described in Section 3.5. As a result, the velocity resulting from this projection satisfies the divergence constraint to $O(h^2)$ not only at each level but also at all the coarse/fine interfaces. After the projection all quantities other than pressure are averaged down from fine grids onto the coarser cells underlying them, to ensure that any level ℓ data, $0 \le \ell < \ell_{max}$, is the average of the finer values overlying it.

For the iteration used to define the initial pressure, we compute the time step on the finest level currently defined, and iterate all levels with that time step ($\Delta t^{\ell_{max}}$), i.e. without subcycling. Here, however, the velocity is advanced on each level without being projected at that level, i.e. $U^{*,\ell}$ but not $U^{1,\ell}$ is defined for $0 \le \ell \le \ell_{max}$. One multilevel composite projection is then done on the field $(U^* - U^0)/\Delta t^{\ell_{max}}$ to compute the pressure update on all levels simultaneously. Here again the constraint is approximately satisfied not only on each level but also at all the coarse/fine interfaces. As in the single grid case, during the iteration procedure the values of U^1 computed by the projection are discarded, and the new value of pressure is used for the next iteration. When the iteration is complete, the regular time-stepping procedure (i.e. with subcycling) is begun.

3.3 Overview of Time-Stepping Procedure

There are two approaches to solving a system of equations on a composite hierarchy of grids like those presented here. The first is to solve the system on the composite hierarchy at each time step using a combination of multilevel operations. This approach requires that every level be advanced with the same time step. The second approach is to advance each level independently at its own time step ($\Delta t^{\ell+1} = \frac{1}{r}\Delta t^{\ell}$), requiring no interlevel communication other than the supplying of Dirichlet data from a coarse level to be used as boundary conditions at the next finer level, then to synchronize the data at different levels at some specified interval. The algorithm is this paper is based on the latter approach for reasons detailed further in a later section.

The adaptive time-step algorithm can most easily be thought of as a recursive procedure, in which to advance level ℓ, $0 \leq \ell \leq \ell_{max}$ the following steps are taken:

- Advance level ℓ in time as if it is the only level. Supply boundary conditions for the velocity, density, concentration and pressure from level $\ell - 1$ if level $\ell > 0$, and from the physical domain boundaries.
- If $\ell < \ell_{max}$
  - Advance level $(\ell + 1)$ r times with time step $\Delta t^{\ell+1} = \frac{1}{r}\Delta t^{\ell}$. Use boundary conditions for the velocity, density, concentration and pressure from level ℓ, and from the physical domain boundaries.
  - Synchronize the data between levels ℓ and $\ell + 1$, and interpolate corrections to higher levels if $\ell + 1 < \ell_{max}$.

Before describing the steps of the synchronization in detail, we first discuss, in general terms, how to synchronize the data at different levels so that the solution as computed on each level sequentially can most closely approximate the solution which would be found using composite solves. The nature of the synchronization depends on the nature of the operator; however, during the advance of each level, for each operator we supply Dirichlet boundary data for the fine grids from the next coarser grid. Thus implies that the values at both levels are consistent, but the computed fluxes at the coarse/fine interfaces are not. It is this mismatch in fluxes which accounts for the discrepancy between solutions.

For hyperbolic equations the correction of flux discrepancies, which we summarize below, is discussed in detail in Berger and Colella [9]. For simplicity, we first assume that the coarse and fine grids use the same time step and that we have computed fluxes on the coarse grid, and on the fine grid using coarse grid data for boundary conditions. Because the operator is local and the discretization is explicit, the mismatch affects the solution only adjacent to the coarse/fine interface. We make the assumption that the fluxes as calculated on the fine grid are more accurate than those calculated on the coarse level. Thus, we replace the coarse grid flux at a coarse/fine interface with the average of the fine grid fluxes in the coarse grid update. This corrects the coarse grid values immediately adjacent to but "outside" the fine grids and represents a composite update to the solution. When generalized to the subcycling case, the fine grid fluxes used to update the coarse boundary cells are averaged in time as well.

For a self-adjoint elliptic operator, $\nabla \cdot \beta\nabla\phi$, there are also fluxes, $\beta\nabla\phi$, associated with each face which are differenced to discretize the operator. Again we solve on each level separately with only Dirichlet data from the coarse grid, which generates a mismatch between the coarse and fine level fluxes at the interface. In this case we are matching

Dirichlet data but allowing a mismatch in the Neumann data, whereas a composite solution would satisfy both matching conditions at the interface. Unlike the hyperbolic case, here we must solve an auxiliary elliptic equation with the flux mismatch as a source term in order to correct the solution. Furthermore, although the source is localized along the coarse/fine interface, the correction modifies the entire solution on both coarse and fine grids. In the context of the projection method with subcycling, we compute the temporal average of the fine grid fluxes to compute the flux mismatch that forms the source term for the elliptic correction. Similar considerations are used for parabolic equations which are solved implicitly in the method.

With these general principles in mind we can now discuss the specific sources of mismatch in the adaptive algorithm and briefly describe how each is corrected. The specific details of these corrections are described in the following subsection.

After the level $\ell+1$ data have been advanced to the same point in time as the level ℓ data, there are four mismatches in the composite solution which require correction in the synchronization step:

(M.1) The data at level ℓ that underlie the level $\ell+1$ data are not synchronized with the level $\ell+1$ data.

(M.2) The composite advection velocity computed from the MAC projection, defined as the time-averaged (over a level ℓ time step) level $\ell+1$ advection velocity on all level $\ell+1$ faces including the $\ell/(\ell+1)$ interface, and the level ℓ advection velocity on all other level ℓ faces, does not satisfy the composite divergence constraint at the $\ell/(\ell+1)$ interface. This mismatch results in spatially constant advected quantities with no source terms not remaining constant.

(M.3) The advective and diffusive fluxes from the level ℓ faces and the level $\ell+1$ faces do not agree at the $\ell/(\ell+1)$ interface, resulting in a loss of conservation.

(M.4) The composite new-time velocity, defined as the level $\ell+1$ new-time velocity on all level $\ell+1$ cells, and the level ℓ new-time velocity on all level ℓ cells not underlying level $\ell+1$ grids, does not satisfy the composite divergence constraint at the $\ell/(\ell+1)$ interface.

The aim of the synchronization steps is to correct the effects of each mismatch. As in the hyperbolic case (see, e.g., [10, 9]), (M.1) is easily corrected by averaging the level $\ell+1$ data onto the level ℓ data beneath. We denote this correction by (S.1). Velocity and scalar data at the new time are averaged from the fine grids onto the coarse level in a simple cell-centered averaging procedure. The level $\ell+1$ pressure is then averaged in time onto the level ℓ nodes, such that the pressure at a level ℓ node underlying a level $\ell+1$ node is defined to be the average over time of the r level $\ell+1$ values at that node defined within the single level ℓ time step just completed. This is consistent with the understanding, as described in the next subsection, that pressure is defined over a time interval rather than at a specific time.

The second mismatch, (M.2), is discretely manifest as a non-zero difference between the coarse and the effective time-averaged fine advection velocities at the coarse/fine interface. This difference results from not having satisfied the elliptic matching conditions at the coarse/fine interface during the MAC solve. As discussed earlier, an elliptic solve is necessary to correct for the mismatch. We perform a level ℓ "MAC sync solve" (S.2) for δe, with the right-hand-side defined as the divergence of the mismatch between the level ℓ and the time-averaged level $\ell+1$ advection velocities. The correction velocity field is defined as the inverse-density-weighted gradient of δe, and is used to re-advect velocity and scalars at level ℓ. These "re-advection corrections," as well as the interpolation of these corrections to all higher levels, are combined with the refluxing corrections to modify the solution as described below.

In the case of zero viscosity/diffusivity, the re-advection corrections described immediately above, and the correction for (M.3), which is simply the hyperbolic refluxing step described earlier, are added directly to the new-time solution. The refluxing corrections modify the solution only on the coarse grid cells immediately outside the fine grids; the re-advection corrections modify the solution at all cells at level ℓ and higher.

However, in the case of non-zero viscosity/diffusivity, the modification of the solution by the re-advection and refluxing corrections requires solving additional elliptic equations (S.3).

In the single grid projection algorithm, advective and diffusive fluxes are not added directly to the solution update; rather, they form part of the right-hand-sides for the parabolic solves associated with the Crank-Nicolson discretization of the diffusive terms in the velocity and scalar update equations. Similarly, in the synchronization step, the advective and diffusive flux mismatches, as well as the re-advection corrections, define the right-hand-sides for the refluxing solves. The solutions to the elliptic refluxing equations at level ℓ will modify the new-time velocity and scalar data on all grids at level ℓ, and the interpolation of these corrections will modify the new-time data on all grids at all higher levels. These corrections to the velocity field are not divergence-free, however, and must be projected before they can be added to the new-time solution.

The fourth mismatch, (M.4), arises from enforcing only Dirichlet conditions in the level projections. The mismatch is manifest as a non-zero composite residual at the interface found using a multilevel stencil which sees both the coarse and fine data adjacent to the interface. We can fix (M.4) using a composite (two-level) nodal projection, called the "sync projection," with the right-hand-side defined by the composite residual at the interface between levels ℓ and $\ell+1$; exactly how this residual is calculated will be discussed in the following section. Computationally, we take advantage of the linearity of the projection to combine the right-hand-side from the composite residual with the divergence of the corrections to the velocity field resulting from the re-advection and refluxing steps so that only one multilevel sync projection is necessary.

In the next subsection we will discuss the details of advancing a single level of data when it exists within an adaptive hierarchy of grids and then describe the quantities which must be accumulated on the coarse/fine interface over the level ℓ time step to capture the mismatches. Following that we will give the details of the synchronization steps described above.

3.4 Details of Time-Stepping Procedure

Assume now that we are advancing level ℓ, $0 \leq \ell \leq \ell_{max}$, one level ℓ time step. Let $U^{n,\ell}, \rho^{n,\ell}$, and $c^{n,\ell}$ be the velocity, density and concentration at time $n\Delta t^{\ell}$ on the level ℓ grid, where Δt^{ℓ} is the time step of the level ℓ grid. Let A^{ℓ} be the area of a face at level ℓ, and let Vol^{ℓ} be the volume of a grid cell at level ℓ. Let $\phi^{MAC,\ell}$ be ϕ^{MAC} as computed by the MAC projection on level ℓ, and $Gp^{n-1/2,\ell}$ be the lagged pressure gradient at level ℓ. Define $S = \{\rho, c\}$.

3.4.1 Advancing a single level

To advance the data on level ℓ one level ℓ time step, we follow the time-stepping procedure as described for the single grid algorithm in the previous section. We can distinguish two types of operations used to advance the data at a level: those that can be done one grid at a time, and those that must be done at all grids at a single level simultaneously. All advection operations other than the MAC projection are done grid by grid; the MAC projection, parabolic solves and nodal projection must be done on all grids at a level simultaneously. Boundary conditions for these projections and solves, and the interpolation and solution procedure for these equations, are discussed in Sections 3.5 and 3.6.

Boundary conditions for the explicit level ℓ operations are implemented by filling "ghost cells" of each fine grid. These ghost cells are filled by copying from other fine grids where possible, otherwise interpolating from underlying coarse grids or imposing physical boundary conditions, as appropriate.

When a coarse/fine boundary does not coincide with a physical domain boundary for the level ℓ advection step, level $\ell - 1$ velocity and scalar data are interpolated linearly in time and conservatively in space to fill the ghost cells outside the fine grids. (A three-cell-wide zone of ghost cells is needed to compute fourth-order slopes; otherwise just a one-cell-wide zone of ghost cells is needed.) A linear-in-time profile for velocity implies a piecewise-constant-in-time profile for pressure, since the pressure gradient, as a forcing term, correlates with change in velocity over time. Thus in the advection step, the lagged pressure gradient, $Gp^{n-1/2,\ell-1}$, is considered constant in time over the previous level $\ell - 1$ time step, and in the MAC solve, $\phi^{MAC,\ell-1}$ is considered constant over the level $\ell - 1$ time step and is interpolated spatially to provide boundary conditions for $\phi^{MAC,\ell}$ where necessary.

3.4.2 Computing the Coarse-Fine Mismatch

Over the course of a level ℓ time step, we must accumulate several quantities at the $\ell/(\ell+1)$ interface in order to correctly capture the mismatches at the end of the level ℓ time step. We refer to the face- or node-based data structures that contain these quantities as registers. The velocity and flux registers accumulate the mismatch between the level ℓ and level $\ell + 1$ face-based advection velocities and fluxes, respectively. The sync register accumulates the node-based composite residual which will be used in the right-hand-side for the sync projection.

These registers are defined only on the $\ell/(\ell + 1)$ interface and are indexed by level ℓ indices. Note that in d dimensions, one level ℓ face contains r^{d-1} level $\ell + 1$ faces; the sums over faces below should be interpreted as summing over all level $\ell + 1$ faces which are contained in the level ℓ face. The sums over k should be understood as summing over the r level $\ell+1$ time steps contained within a single level ℓ time step.

At the end of the level ℓ time step, the velocity register (δU^ℓ) holds the area-weighted difference between the MAC-projected advection velocity at level ℓ and the time average over one level ℓ time step of the space average over the area of the level ℓ face of the MAC-projected advection velocity at level $\ell+1$:

$$\delta U^\ell = -A^\ell U^{ADV,\ell} + \frac{1}{r}\sum_{k=1}^{r}\sum_{faces}(A^{\ell+1}U^{ADV,k,\ell+1}).$$

The advective flux registers for velocity ($\delta F_U^{adv,\ell}$) and scalars ($\delta F_S^{adv,\ell}$) contain the time step- and area-weighted difference between the advective fluxes calculated at level ℓ and the time average over the level ℓ time step of the space average over the area of the level ℓ face of the advective fluxes at level $\ell+1$:

$$\delta F_U^{adv,\ell} = \Delta t^\ell(-A^\ell F_U^{adv,\ell} + \frac{1}{r}\sum_{k=1}^{r}\sum_{faces}(A^{\ell+1}F_U^{adv,k,\ell+1}))$$

$$\delta F_S^{adv,\ell} = \Delta t^\ell(-A^\ell F_S^{adv,\ell} + \frac{1}{r}\sum_{k=1}^{r}\sum_{faces}(A^{\ell+1}F_S^{adv,k,\ell+1}))$$

The viscous/diffusive flux registers for velocity ($\delta F_U^{visc,\ell}$) and scalars ($\delta F_S^{diff,\ell}$) are defined analogously, but with the viscous/diffusive fluxes rather than advective fluxes:

$$\delta F_U^{visc,\ell} = \Delta t^\ell(-A^\ell F_U^{visc,\ell} + \frac{1}{r}\sum_{k=1}^{r}\sum_{faces}(A^{\ell+1}F_U^{visc,k,\ell+1}))$$

$$\delta F_c^{diff,\ell} = \Delta t^\ell(-A^\ell F_c^{diff,\ell} + \frac{1}{r}\sum_{k=1}^{r}\sum_{faces}(A^{\ell+1}F_c^{diff,k,\ell+1}))$$

We note here that the signs of the quantities added to the flux registers actually depend on the orientation of the normal facing away from the fine grid. We follow the convention below that the signs are given for the faces at which the fine grid is in the direction of the lower coordinate indices.

In accumulating the composite residual to be used in the right-hand-side for the sync projection, we must keep in mind that the nodal projection is not an exact projection. If we defined the composite residual as the composite divergence of U^{n+1} on levels ℓ and $\ell+1$ at the end of the level ℓ time step, then even if the composite divergence constraint had been satisfied exactly by the solution of the level projections, the residual would not be zero because of the approximate nature of the projection. Hence, in order to capture only the mismatch at the coarse/fine interface and not the "approximateness" of the projection, the composite residual is defined as a time-averaged residual that measures the extent to which the level projections fail to satisfy the equations defining the composite projection at the coarse/fine interface, but not the extent to which the projection is non-exact. The composite residual has the form

$$Res_{S-P}^\ell = D_{coarse}(V^\ell - \frac{1}{\rho^{n+1/2,\ell}}G\phi^\ell) + \frac{1}{r}\sum_{k=1}^{r}D_{fine}(V^{k,\ell+1} - \frac{1}{\rho^{n_k+1/2,\ell+1}}G\phi^{k,\ell+1}),$$

where the divergence operator D_{coarse} is defined to include only that contribution to the usual nodal divergence operator which comes from the level ℓ side of the $\ell/(\ell+1)$ interface, and D_{fine} is defined to include only that contribution to the divergence which comes from the level $\ell+1$ side of the interface. The fine grid contribution is computed along the fine nodes of the interface and averaged onto the coarse nodes. Here V^ℓ, as in the single grid projection, is the vector $(U^{*,\ell} - U^{n,\ell})/\Delta t^\ell$.

3.4.3 Synchronization of data

The first synchronization step, (S.1), was described in the previous subsection. Here we give the details of (S.2) - (S.4).

The mismatch, (M.2), is captured in the velocity register δU^ℓ; the divergence of δU^ℓ defines the right-hand-side for the level ℓ MAC sync solve (S.2). We solve

$$D^{E\to C}\left(\frac{A^\ell}{\rho^{n+1/2,\ell}} G^{C\to E}(\delta e^\ell)\right) = \widetilde{D}^{E\to C}(\delta U^\ell)$$

on all grids at level ℓ for the correction δe^ℓ. Recall δU^ℓ is defined only at the coarse/fine interface; here $\widetilde{D}^{E\to C}$ is defined to be the MAC divergence operator evaluated only on the level ℓ cells adjacent to the interface but not underlying any level $\ell+1$ grids. (At level ℓ cells underlying level $\ell+1$ grids the right-hand-side is zero.) Boundary conditions on physical no-flow boundaries are homogeneous Neumann ($\frac{\partial(\delta e)^\ell}{\partial n} = 0$); on outflow $\delta e^\ell = 0$. If $\ell > 0$, the boundary conditions for δe^ℓ are given as homogeneous Dirichlet conditions on the level $\ell - 1$ cells outside the level ℓ grids. We then define the correction velocity field from δe^ℓ :

$$U^\ell_{corr} = \frac{-1}{\rho^{n+1/2,\ell}} G^{C\to E}(\delta e^\ell).$$

We now use the correction velocity field to define flux corrections at all level ℓ faces. Because of memory considerations we do not store all the time-centered face states, so we must redefine these on all level ℓ faces. That is, we recreate $\tilde{U}^{n+1/2,\ell}$ and $\tilde{S}^{n+1/2,\ell}$ using $U^{ADV,\ell}$ for upwinding, identically to the procedure immediately following the level ℓ MAC projection. The flux corrections, $F_U^{corr,\ell} = U^\ell_{corr}\tilde{U}^{n+1/2,\ell}$ and $F_S^{corr,\ell} = U^\ell_{corr}\tilde{S}^{n+1/2,\ell}$, are then defined.

Because we must diffuse the re-advection and refluxing corrections before adding them to the new-time solution, and because even for inviscid flow the corrections to the new-time velocity field do not satisfy the divergence constraint, we do not add them directly to the solution. Rather, the divergence of the re-advection flux corrections is added to the advective and viscous/diffusive flux mismatches to define the cell-centered right-hand-sides for the refluxing solves (S.3):

$$\begin{aligned}
RHS^\ell_{V_{sync}} &= -D^{E\to C} F_U^{corr,\ell} - \frac{1}{\Delta t^\ell\, Vol^\ell}(\delta F_U^{adv,\ell} + \frac{1}{\rho^{n+1/2,\ell}} \delta F_U^{visc,\ell}) \\
RHS^\ell_{S_{sync}} &= -D^{E\to C} F_S^{corr,\ell} - \frac{1}{\Delta t^\ell\, Vol^\ell}(\delta F_S^{adv,\ell} + \delta F_S^{diff,\ell}).
\end{aligned}$$

Then, we solve for the correction to the solution, V^ℓ_{sync} and S^ℓ_{sync},

$$\begin{aligned}(1-\frac{\mu\Delta t}{2\rho^{n+1/2,\ell}}\Delta^h)V^\ell_{sync} &= RHS^\ell_{V_{sync}},\\ (1-\frac{k\Delta t}{2}\Delta^h)S^\ell_{sync} &= RHS^\ell_{S_{sync}},\end{aligned}$$

If $\ell > 0$, we must now modify the level $\ell-1$ velocity registers and flux registers to account for the corrections to the solution due to the re-advection corrections as well as the diffused corrections. This is analogous to the accumulation of advective and diffusive fluxes while advancing of a single level. To do this, we set

$$\begin{aligned}\delta U^{\ell-1} &:= \delta U^{\ell-1}+\frac{1}{r}\sum_{faces}(A^\ell U^\ell_{corr})\\ \delta F_U^{adv,\ell-1} &:= \delta F_U^{adv,\ell-1}+\frac{1}{r}\Delta t^{\ell-1}\sum_{faces}(A^\ell F_U^{corr,\ell}),\\ \delta F_U^{visc,\ell-1} &:= \delta F_U^{visc,\ell-1}+\frac{1}{r}\Delta t^{\ell-1}\sum_{faces}(\frac{\mu}{2}A^\ell G^{C\to E}V^\ell_{sync}),\\ \delta F_S^{adv,\ell-1} &:= \delta F_S^{adv,\ell-1}+\frac{1}{r}\Delta t^{\ell-1}\sum_{faces}(A^\ell F_S^{corr,\ell}),\\ \delta F_S^{diff,\ell-1} &:= \delta F_S^{diff,\ell-1}+\frac{1}{r}\Delta t^{\ell-1}\sum_{faces}(\frac{k}{2}A^\ell G^{C\to E}S^\ell_{sync}).\end{aligned}$$

We can now add the corrections to the scalar fields:

$$S^{n+1,\ell} := S^{n+1,\ell}+\Delta t^\ell S^\ell_{sync}$$

and if $\ell < \ell_{max}$, we interpolate the correction onto the fine grids at *all* finer levels, q, $\ell < q \le \ell_{max}$ using conservative interpolation:

$$S^{n+1,q} := S^{n+1,q}+\Delta t^\ell Interp_{cons}(S^\ell_{sync})$$

This completes the synchronization steps for scalar quantities.

The sync projection, (S.4), must account for the mismatch, (M.4), and also for the corrections now stored in V^ℓ_{sync}. In order to correct for just the mismatch, (M.4), we would solve

$$L_\rho^{n+1/2}\phi_1^{SP} = Res^\ell_{S-P}$$

where Res^ℓ_{S-P} is the field we have accumulated on level ℓ nodes by taking one-sided divergences at levels ℓ and $\ell+1$. Here we would do a composite solve on levels ℓ and $\ell+1$ to create a solution on both levels. We would then subtract $\frac{\Delta t^\ell}{\rho^{n+1/2}}G\phi_1^{SP}$ from the new-time velocity field at both levels.

To project the corrections stored in V^ℓ_{sync}, we would interpolate V^ℓ_{sync} to level $\ell+1$ to define $V^{\ell+1}_{sync}$, take a composite divergence, and solve

$$L_\rho^{n+1/2}\phi_2^{SP} = D(V_{sync})$$

for ϕ_2^{SP} on a composite grid. Then we would define $\Delta t^\ell V_{proj}$ as the contribution to U^{n+1}, where $V_{proj} = \mathbf{P} V_{sync} = V_{sync} - \frac{1}{\rho^{n+1/2}} G \phi_2^{SP}$.

Given that, in general, $Res_{S-P}^\ell \neq 0$, and the contributions in V_{sync} are not already divergence-free, we merge the above procedures, and we see that the field we want to add to the existing new-time velocity field is $\Delta t^\ell(-G\phi_1^{SP} + V_{proj})$, which is equivalent to adding $\Delta t^\ell V_{sync}$ and subtracting $\frac{\Delta t^\ell}{\rho^{n+1/2}} G(\phi_1^{SP} + \phi_2^{SP})$. We note that if we define $\phi_{sync} = \phi_1^{SP} + \phi_2^{SP}$, then

$$L_\rho^{n+1/2}(\phi_{sync}) = Res_{S-P}^\ell + D(V_{sync}),$$

and thus in practice we need not separate ϕ_1^{SP} from ϕ_2^{SP}. Rather we solve the above, and add the corrections to the velocity and pressure fields:

$$U^{n+1,\ell} := U^{n+1,\ell} + \Delta t^\ell (V_{sync}^\ell - \frac{1}{\rho^{n+1/2,\ell}} G \phi_{sync}^\ell)$$

$$U^{n+1,\ell+1} := U^{n+1,\ell+1} + \Delta t^\ell (V_{sync}^{\ell+1} - \frac{1}{\rho^{n+1/2,\ell+1}} G \phi_{sync}^{\ell+1})$$

$$p^{n+1/2,\ell} := p^{n+1/2,\ell} + \phi_{sync}^\ell$$

$$p^{n+1-\frac{1}{2r},\ell+1} := p^{n+1-\frac{1}{2r},\ell+1} + \phi_{sync}^{\ell+1}$$

In the above solution, $\rho^{n+1/2,\ell} \equiv 1/2(\rho^{n,\ell} + \rho^{n+1,\ell})$, and $\rho^{n+1/2,\ell+1}$ is the weighted average over the level ℓ time step of the density at level $\ell+1$.

If $\ell > 0$ we must account for the correction to the level ℓ velocity field in the composite residual for the $(\ell-1)$ / ℓ sync projection. We do this by adding a contribution to $Res_{S-P}^{\ell-1}$:

$$Res_{S-P}^{\ell-1} := Res_{S-P}^{\ell-1} + \frac{1}{r} D_{fine}(V_{sync}^\ell - \frac{1}{\rho^{n+1/2,\ell}} G \phi_{sync}^\ell)$$

where the contribution in D_{fine} comes only from the level ℓ grids and is defined only at nodes on the $(\ell-1)$ / ℓ interface which are not also at level $\ell+1$ (i.e. which are not at a physical boundary). This modification of the level ℓ data will be seen by the level $\ell-1$ data through the next level $(\ell-1)$ / ℓ sync projection.

If $\ell+1 < \ell_{max}$, we then interpolate the node-based pressure correction $\phi^{\ell+1}$ using bilinear interpolation, and interpolate the cell-based velocity correction, $V_{sync}^{\ell+1}$, using conservative interpolation, onto fine grids at *all* finer levels, q, $\ell+1 < q \le \ell_{max}$:

$$U^{n+1,q} := U^{n+1,q} + \Delta t^\ell Interp_{cons}(V_{sync}^{\ell+1} - \frac{1}{\rho^{n+1/2,\ell+1}} G \phi_{sync}^{\ell+1})$$

$$p^{n+1-\frac{1}{2r^{q-\ell}}} := p^{n+1-\frac{1}{2r^{q-\ell}}} + Interp_{bilin}(\phi_{sync}^{\ell+1}).$$

We note here that in previous work (see [2]) we had believed that solving the equations above on the level ℓ grids alone would be sufficiently accurate since both V_{sync} and Res_{S-P} are defined at coarse grid resolution. In a single-level solve, V_{sync} and ϕ_{sync} would be defined only at level ℓ and corrections at levels $\ell+1$ and higher would be defined by interpolation. While it is true that the source for the equation is at coarse grid resolution, if solved on a composite hierarchy the behavior of the solution on the fine grid away from the coarse/fine interface is not well represented on the coarse grid. As a result we have decided to use the composite grid solve despite the additional CPU expense.

Computational examples have borne out that in some but not all cases, the effect of using the multilevel rather than single-level solve is nontrivial. One can show analytically that the increased accuracy is most significant when there is significant variation in the right-hand-side for the level $\ell\ /(\ell+1)$ sync projection along the level $\ell+1$ boundaries. In one dimension the solution to Laplace's equation is linear, and hence linear interpolation of the solution from a coarse to fine grid is exact; for two and three dimensions the Green's function is proportional to the log or the inverse, respectively, of the distance from the source; these functions are not well approximated near the source by linear interpolation.

3.5 Details of the Nodal Projections

The AMR time-stepping scheme requires projection solutions on single levels ("level projection") and pairs of levels ("sync projection"), and for initialization it requires a projection on all levels at once. We compute these solutions using a multigrid algorithm adapted to the AMR grid hierarchy. The main complications involve the choice of coarse/fine interface stencils, and the need to support refinement ratios of 2 or 4 between levels.

A mathematical description of the projection operator is given in Section 2.2. The essential point is that we must solve

$$\nabla \cdot \left(\frac{1}{\rho} \nabla \phi \right) = RHS \tag{16}$$

for ϕ over some subset of the AMR levels, for some right-hand-side RHS determined by the needs of the time-stepping algorithm. In what follows we abbreviate the elliptic operator as L_ρ, and we often work in residual-correction form so that the equation to be solved becomes $L_\rho e = r$.

Equation (14) defines compact stencils for the elliptic operator in the grid interiors. The same finite-element integral provides the somewhat more complicated stencils used on the coarse/fine interfaces. This is in contrast to some other multilevel methods, e.g. FAC [19], which derive the relationships between coarse and fine grid data from the multigrid algorithm itself. An advantage to the finite-element formulation is that, if used consistently for both the elliptic operator L_ρ and the divergence D, the right-hand-side of (16) is always in the range of the elliptic operator. Thus no compatibility correction is required, even for singular problems with periodic or Neumann boundary conditions.

Figure 1 shows the spatial extent of the stencils for the 2-d nine-point discretization for a refinement ratio of 4; the 2-d five-point and 3-d seven-point discretizations are similar. In the interiors of the coarse and fine levels each finite element basis function is associated with a node of the mesh and extends over the four adjacent cells. On the interfaces the basis functions are associated with coarse nodes only, with values at the intermediate fine nodes linearly interpolated from the coarse nodes.

In the diagrams on the left side of the figure, a value at the central node is computed using values at the indicated surrounding nodes and cells. For divergence DV of a velocity field V, velocity values at cells marked with open circles are used. Likewise the linear operator expression $L_\rho\phi$ involves ρ at these same cells and ϕ at the nodes marked by solid

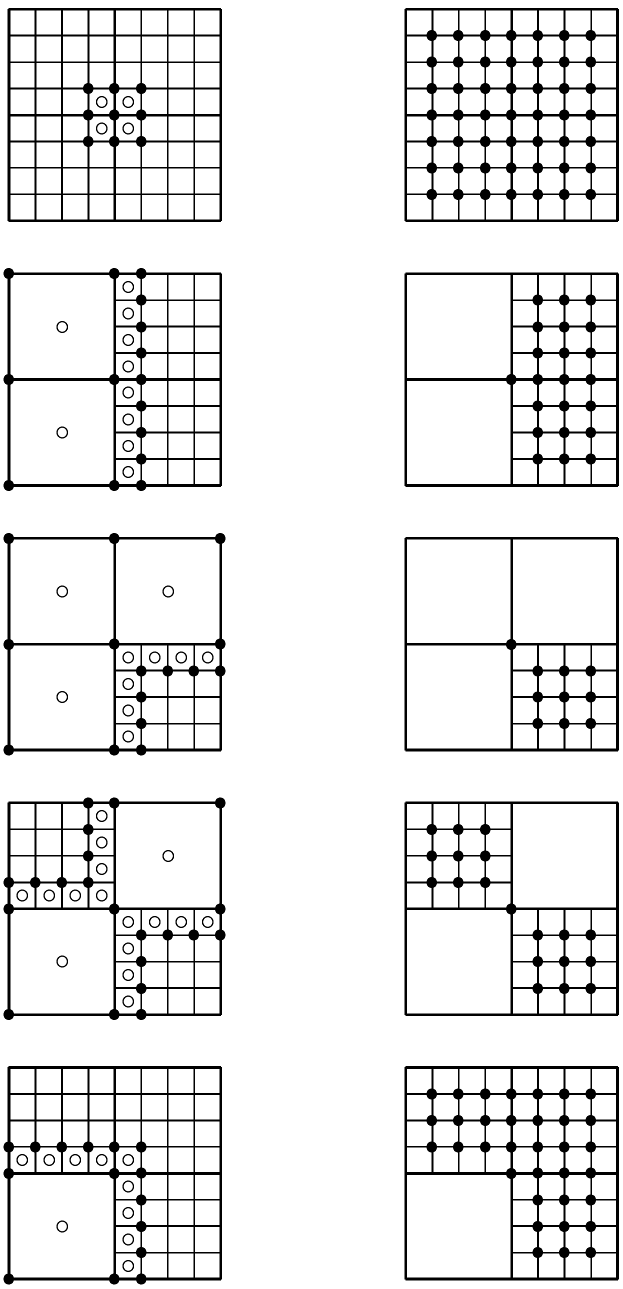


Figure 1: Stencils at grid edges and corners, shown for a refinement ratio of four. On the left, the stencil for $\nabla \cdot \frac{1}{\rho} \nabla \phi$ uses ϕ values defined at nodes (solid circles) and ρ values defined at cells (open circles). Also, the divergence stencil for $\nabla \cdot V$ uses V defined at these same (open) cell positions. On the right are the stencils for restricting residuals to the coarse grid.

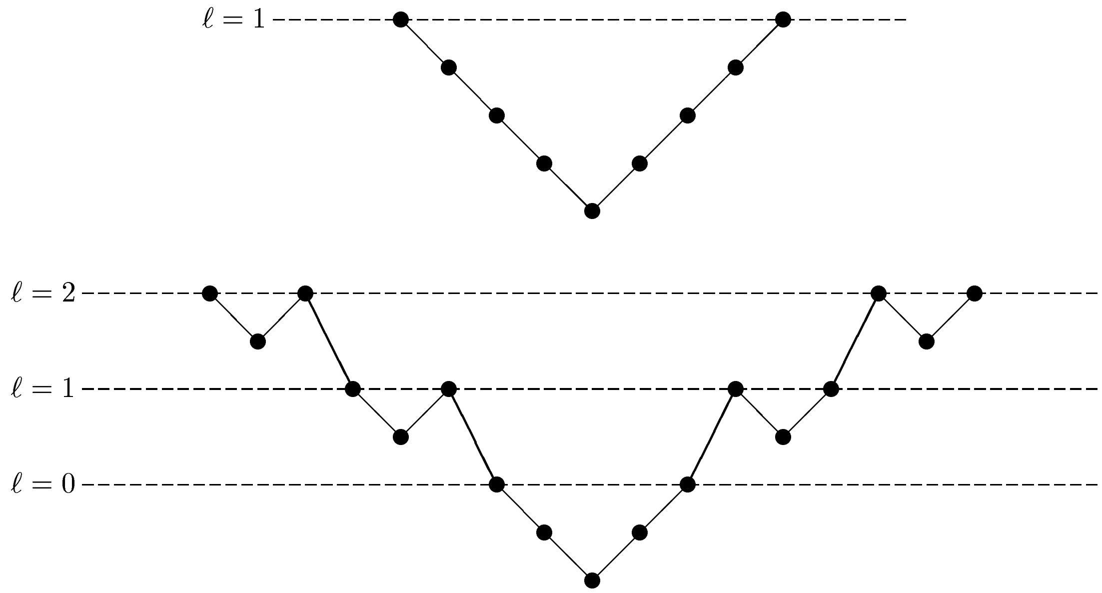


Figure 2: In the multigrid V-cycle (top), operations apply only to interior points of a level. In the multilevel cycle (bottom) two operations are defined that cross the coarse/fine interface—computing the residual, and restricting it to the coarse grid. Dotted lines show AMR levels, other levels are used only by the multigrid algorithm.

circles. The diagrams on the right side show the nodes involved when averaging residuals from the fine level down to the coarse level. A residual computed on an interface node represents a basis function with less area and hence less weight than a full coarse node. In the restriction step of a multigrid solve this value is combined with nearby fine grid values in order to produce a correctly-weighted coarse grid value.

Equations for the difference and restriction stencils are presented for both two and three dimensions in the appendix. In 2-d there are only the five basic geometric configurations shown, not counting rotations and reflections. In 3-d, however, the number is much larger, and a more general element assembly process becomes necessary.

Specifying the stencils at all points in the domain defines the linear system; now we consider the separate question of how to solve it. In preparation for a multigrid solve, we start with the levels of the AMR structure on which we want the solution, and construct new levels between and below (i.e., coarser than) the active AMR levels so that adjacent pairs of levels are related by a factor of 2. These new levels are for use by the multigrid solver alone; they do not participate in any other part of the adaptive algorithm. Each new level is created by coarsening the next finer level above it, and will not communicate with coarser AMR levels below it in any way.

Figure 2 may make the relationships between levels more clear. The top picture shows a multigrid V-cycle (cf. [30]) for a level projection—all coarse levels are obtained by coarsening the grid structure of the single active AMR level. The bottom picture shows a multilevel cycle involving three AMR levels with a factor of 4 refinement between each level.

We continue to denote AMR levels by ℓ, with a particular subset of levels $\ell_{lo} \leq \ell \leq \ell_{hi}$ being active in a given multilevel solve. We similarly denote the multigrid levels by m, $0 \leq m \leq m_{hi}$. Since the layout of multigrid levels depends on which AMR levels are currently active, it will typically be different for each invocation of the solver. Let $m = m(\ell)$ be the multigrid level corresponding to a given AMR level. Note that while $m(\ell_{hi}) = m_{hi}$, generally $m(\ell_{lo}) \neq 0$.

A multigrid V-cycle for the linear system $L_\rho^m e^m = r^m$, where m is either identical to or coarsened from an AMR level ℓ, has the following recursive form:

Begin V-cycle(L_ρ^m, e^m, r^m, m, ℓ, ν_1, ν_2):

> **If** $(m = 0)$ **then**
>
> > Solve(L_ρ^m, e^m, r^m)
>
> **Else if** $(\ell - 1 \geq \ell_{lo}$ and $m - 1 = m(\ell - 1))$
>
> > Relax(L_ρ^m, e^m, r^m, ν_2)
>
> **Else**
>
> > Relax(L_ρ^m, e^m, r^m, ν_1)
> > $r^{m-1} := I_m^{m-1}(r^m - L_\rho^m e^m)$
> > $e^{m-1} := 0$
> > V-cycle(L_ρ^{m-1}, e^{m-1}, r^{m-1}, $m-1$, ℓ, ν_1, ν_2)
> > $e^m := e^m + I_{m-1}^m e^{m-1}$
> > Relax(L_ρ^m, e^m, r^m, ν_2)
>
> **Endif**

End V-cycle

The "Relax" operation consists of two or more (ν) iterations of red-black Gauss-Seidel, while the "Solve" operation on the coarsest level uses a diagonally preconditioned conjugate gradient routine. All operations take place on the domain Ω^ℓ consisting of all grids at level ℓ. (Ω without a superscript represents the computational domain as a whole.) Boundary conditions on $\partial\Omega^\ell - \partial\Omega$ are Dirichlet conditions from level $\ell - 1$, while on $\partial\Omega^\ell \cap \partial\Omega$ they are physical boundary conditions for the edge of the computational domain. Before each relaxation or residual computation, it is necessary to update ghost nodes around the border of each grid from the boundary conditions or from neighboring fine grids. After relaxations it is also necessary to synchronize the nodes shared by adjacent grids. We perform these updates quickly using optimized grid-to-grid copy operations.

There are two motivations for including the coarse-level conjugate gradient "Solve" operation. One is that the coarsest multigrid level may consist of hundreds of cells spread over many grids, and thus may not be small enough to solve by Gauss-Seidel relaxations alone. The other is that for problems with large discontinuities in density, Gauss-Seidel relaxations may converge too slowly even on a small grid.

To complete the description of the multigrid scheme, we must specify the restriction and interpolation operators and show how the linear operator itself is applied on coarsened grids. Restriction is the simplest. We use a "full-weighting" method, where each fine node provides an equal contribution to the coarse grid residual. The residual thus behaves as if it were a conserved quantity in the multigrid system. In 2-d the stencil for this is

$$[I_m^{m-1}] = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

In the multilevel algorithm to follow, we also have to deal with restriction at coarse/fine interfaces. The details are more complicated, but the same conservation arguments apply. The actual stencils we use appear in the appendix.

Next we address the coarse grid operators themselves. For the sake of brevity in this section we will not present formulae for the various difference stencils here; those can be found in the appendix. What all of these stencils have in common is a dependence on a coefficient $1/\rho$ in the four cells surrounding each node (eight cells in 3-d). We call this coefficient σ, so that the elliptic operation becomes $\nabla \cdot \sigma \nabla \phi$. For axisymmetric ($r$-$z$) problems we can use $\sigma = r/\rho$ instead, which gives us the same stencils as in the Cartesian grid case except for a small (second-order) correction.

Since σ is analogous to conductivity, we coarsen it by doing an arithmetic average transverse to each "flux" and a harmonic average parallel to the flux. This gives us separate σ's for each coordinate direction on the coarser grids. For the x-direction in 2-d the result is

$$\sigma_{i/2,j/2}^{(x),m-1} = \frac{1}{\dfrac{1}{\sigma_{i,j}^m + \sigma_{i,j+1}^m} + \dfrac{1}{\sigma_{i+1,j}^m + \sigma_{i+1,j+1}^m}},$$

with an analogous expression for $\sigma_{i/2,j/2}^{(y),m-1}$ in the y-direction. For still coarser grids we use the same formula, using values of $\sigma^{(x),m-1}$ to compute $\sigma^{(x),m-2}$ and values of $\sigma^{(y),m-1}$ to compute $\sigma^{(y),m-2}$.

The linear operators on the coarsened grids then take the same form as the operators on the fine grids, using these coarsened coefficients. More elaborate coarsening strategies may be added to the algorithm in the future to give better performance with large discontinuities in density, but this one has provided adequate multigrid convergence for most of our present applications.

Having introduced the directional σ's, we can now present the operator-dependent interpolation stencils required by the multigrid algorithm. Like the σ's themselves, these formulae work with both the five-point and nine-point linear operators in 2-d, and an obvious extension applies to the seven-point operator in 3-d. (For simplicity, we present the formulae as if we were computing $e^{m+1} := I_m^{m+1} e^m$.) We first inject the points that coincide with their coarse equivalents,

$$e_{2i-1/2,2j-1/2}^{m+1} = e_{i-1/2,j-1/2}^m,$$

then we weight the points offset in the x-direction using the coefficients for differences in that direction,

$$e_{2i+1/2,2j-1/2}^{m+1} = \frac{(\sigma_{2i,2j-1}^{(x),m+1} + \sigma_{2i,2j}^{(x),m+1}) e_{2i-1/2,2j-1/2}^{m+1} + (\sigma_{2i+1,2j-1}^{(x),m+1} + \sigma_{2i+1,2j}^{(x),m+1}) e_{2i+1+1/2,2j-1/2}^{m+1}}{\sigma_{2i,2j}^{(x),m+1} + \sigma_{2i,2j+1}^{(x),m+1} + \sigma_{2i+1,2j}^{(x),m+1} + \sigma_{2i+1,2j+1}^{(x),m+1}},$$

and use a similar formula for points offset in the y-direction. Finally, the points offset in both the x- and y-directions are defined by the composite formula

$$e^{m+1}_{2i+1/2,2j+1/2} = \frac{\begin{array}{r}\{(\sigma^{(x),m+1}_{2i,2j} + \sigma^{(x),m+1}_{2i,2j+1})e^{m+1}_{2i-1/2,2j+1/2} + (\sigma^{(x),m+1}_{2i+1,2j} + \sigma^{(x),m+1}_{2i+1,2j+1})e^{m+1}_{2i+1+1/2,2j+1/2} + \\ (\sigma^{(y),m+1}_{2i,2j} + \sigma^{(y),m+1}_{2i+1,2j})e^{m+1}_{2i+1/2,2j-1/2} + (\sigma^{(y),m+1}_{2i,2j+1} + \sigma^{(y),m+1}_{2i+1,2j+1})e^{m+1}_{2i+1/2,2j+1+1/2}\}\end{array}}{\sigma^{(x),m+1}_{2i,2j} + \sigma^{(x),m+1}_{2i,2j+1} + \sigma^{(x),m+1}_{2i+1,2j} + \sigma^{(x),m+1}_{2i+1,2j+1} + \sigma^{(y),m+1}_{2i,2j} + \sigma^{(y),m+1}_{2i+1,2j} + \sigma^{(y),m+1}_{2i,2j+1} + \sigma^{(y),m+1}_{2i+1,2j+1}}.$$

Since the interpolation stencils do not extend past the borders of each coarse cell, no special multilevel stencils at coarse/fine interfaces are required in the multilevel algorithm. We do, however, use ordinary linear interpolation instead of the operator-dependent stencils along the interfaces, since the interface stencils for $L_\rho\phi$ assume a linear profile between coarse nodes.

A multilevel cycle for the linear system $L^\ell_\rho\phi^\ell = RHS^\ell$ is as follows:

Begin Multilevel cycle(L^ℓ_ρ, ϕ^ℓ, RHS^ℓ, $r^{m(\ell+1)}$, ℓ, ν_1, ν_2):

$m := m(\ell)$
$r^m := (RHS^\ell - L^\ell_\rho\phi^\ell)$
If $(\ell < \ell_{hi})$ **then** $r^m := I^\ell_{\ell+1} r^{m(\ell+1)}$ on $\Omega^{\ell+1} + \partial\Omega^{\ell+1}$
If $(\ell = \ell_{lo})$ **then** $\nu := \nu_1$ **else** $\nu := 0$
$e^m := 0$
V-cycle(L^ℓ_ρ, e^m, r^m, m, ℓ, ν, ν_2)
$\phi^\ell := \phi^\ell + e^m$
If $(\ell > \ell^{min})$

$$r^m := \begin{cases} (r^m - L^\ell_\rho e^m) & \text{on } \Omega^\ell \\ RHS^\ell - L^{\ell-1,\ell}_\rho\phi^{\ell-1,\ell} & \text{on } \partial\Omega^\ell - \partial\Omega \end{cases}$$

$\phi^{\ell-1}_{old} := \phi^{\ell-1}$
Multilevel cycle($L^{\ell-1}_\rho$, $\phi^{\ell-1}$, $RHS^{\ell-1}$, r^m, $\ell-1$, ν_1, ν_2)
$e^m := I^\ell_{\ell-1}(\phi^{\ell-1} - \phi^{\ell-1}_{old})$ on $\Omega^\ell + \partial\Omega^\ell$
$\phi^\ell := \phi^\ell + e^m$ on $\Omega^\ell + \partial\Omega^\ell$
$r^m := (r^m - L^\ell_\rho e^m)$
$e^m := 0$
V-cycle(L^ℓ_ρ, e^m, r^m, m, ℓ, 0, ν_2)
$\phi^\ell := \phi^\ell + e^m$

Endif

End Multilevel cycle

This cycle is repeated as many times as necessary for convergence. All operations take place on Ω^ℓ, including points of the physical boundary $\partial\Omega$ but not including points of $\partial\Omega^\ell$ bordering the coarser level $\Omega^{\ell-1}$ unless otherwise noted.

3.6 Details of the Cell-Centered Level Solves

The cell-centered solves required by the adaptive projection algorithm as presented here are all single-level solves; the MAC projection, the MAC sync solve, and the parabolic solves done for the Crank-Nicolson representation of the diffusive terms, all involve the same type of spatial discretization. The discretization yields a cell-centered right-hand-side and a cell-centered solution (by contrast to the projections described in the previous subsection in which both the solution and the right-hand-side are defined at nodes). The construction of the right-hand-sides for the MAC projection and the parabolic solves has been defined in Section 2, and for for the MAC synchronization step in Section 3.4. Here we focus on the discretization of the operator and the solution procedure.

The goal in each case is to solve an equation of the form $(\alpha(\mathbf{x}) - \nabla \cdot (\beta(\mathbf{x})\nabla))\phi =$ RHS on the union of grids at a single level with boundary conditions for the union of grids given by physical boundary conditions on $\partial\Omega \cap \partial\Omega_\ell$, and by data from level $\ell - 1$ elsewhere. The discretization of the variable-coefficient elliptic operator uses a standard, five-point in 2-d, seven-point in 3-d, cell-centered finite difference approximation in the interior of the grids. In particular, the discretization can be viewed as computing the MAC divergence of face-based fluxes, $\beta(\mathbf{x})\nabla\phi$. The only complication in these solves, aside from performance issues, is that of maintaining sufficient accuracy at the boundary of the union of grids at a level. In this subsection we describe how the stencils at these boundaries are defined.

We solve this system using standard multigrid methods (V-cycles with red-black Gauss-Seidel relaxation and a conjugate gradient solver at the bottom of the V-cycle) as shown in Figure 2a. The restriction operator is volume-weighted averaging; the multigrid interpolation is piecewise constant.

At each level of the V-cycle (i.e. each multigrid level m), each red or black relaxation sweep is performed on all grids sequentially, with the boundary conditions effectively imposed once per sweep. For convenience, the boundary conditions are represented in the operator at any given point as Dirichlet values in the ghost cells immediately outside the fine grids. For a given fine grid, each ghost cell value can be copied from another fine grid, or defined using physical boundary conditions or the coarse grid data as well as interior data. In the latter two cases, interpolation or extrapolation of the data is usually required to define a value at the ghost cell location.

Physical boundary conditions are typically defined as either Neumann or Dirichlet data on $\partial\Omega$ (as opposed to at the center of the ghost cell just outside the domain). In the case of Dirichlet data, an extrapolation procedure is defined which fits a parabola through the value at the boundary and the two interior grid values along a line normal to the boundary. When used in the stencil for the elliptic operator this gives a second-order approximation to the normal derivative at the boundary. In Figure 3a, the linear operator at point (a) is evaluated using values at the cells marked with the small open or closed circles (the closed circles are legitimate fine grid values; the small open circle is the value at the ghost cell for (a)). The value in the ghost cell is evaluated by the extrapolation procedure defined above, using the data at the large open circle on the boundary as well as the data at the cell values marked by large open circles. For Neumann data, the extrapolation procedure defines a parabola passing through the two interior values and with the given normal derivative at the boundary in order to define a value in the ghost cell. This again gives a second-order accurate approximation to $\beta(\mathbf{x})\frac{\partial\phi}{\partial n}$; in both cases the second-order flux results in first-order local truncation error in the definition of the elliptic operator.

To supply boundary conditions from the coarse data before a red or black sweep, the data are interpolated onto the ghost cells immediately surrounding the fine grid. (See Figure 3b for a 2-d example; in this the thickest lines represent the boundaries of individual fine grids. Note that at the fine-fine interface shown, nothing special is done other than a copy from the other fine grid.) The interpolation is done in two stages: first the coarse grid data (the large closed circles in Figure 3b) are interpolated tangentially to the coarse/fine interface, so that coarse grid data are defined at points (the open circles in Figure 3b) which align with the fine grid points in all but the normal direction to the face. In two dimensions, the interpolation is done by defining, in each cell, first and second derivatives of the data in the tangential direction using centered differences, and then using those to interpolate the data from cell centers to the intermediate points. For example, in cell (b), $\phi_y \equiv (\phi^c - \phi^a)/(2\Delta y_{crse})$ and $\phi_{yy} \equiv (\phi^c + \phi^a - 2\phi^b)/\Delta y_{crse}^2$. However, in the cases where constructing the derivatives would require using coarse grid values which underlie a fine grid, the slopes are computed using a one-sided difference and the second derivatives are set to zero (e.g. for cell (c), $\phi_y \equiv (\phi^c - \phi^b)/\Delta y_{crse}$ and $\phi_{yy} \equiv 0$. If the coarse cells on both sides (in the tangential direction) are under fine grids, then both the first and second derivatives are set to zero and the interpolation scheme reduces to piecewise constant.

In three dimensions the procedure is similar, although the tangential interpolation is done in two directions simultaneously. Here the coarse grid data are used to define a biquadratic function which is used to interpolate to the intermediate points. Analogously to the two-dimensional algorithm, in the case where coarse data underlying a fine grid would be needed to compute a centered difference, the slope calculation in that direction reduces to a one-sided difference and the second derivative is set to zero. If even this is not possible, the first and second derivatives in that direction are set to zero. This is done for each tangential coordinate direction separately, testing only on the four nearest neighbor cells (e.g. $\phi_{i+1,j,k}, \phi_{i-1,j,k}, \phi_{i,j+1,k}, \phi_{i,j-1,k}$ would be used for $\phi_{i,j,k}$ along a face parallel to z = constant). The computation of the cross derivative (ϕ_{xy} in this case) requires the four neighbors along the diagonals (e.g. $(\phi_{xy})_{i,j,k} \equiv (\phi_{i+1,j+1,k} + \phi_{i-1,j-1,k} - \phi_{i-1,j+1,k} - \phi_{i+1,j-1,k})/(4\Delta x_{crse}\Delta y_{crse})$. If any of these values is in a cell underlying a fine grid then ϕ_{xy} is set to zero.

This stage of the interpolation is done at the beginning of the solve as opposed to at each relaxation sweep. At all but the finest level the coarse data is homogeneous because the residual-correction form is used within the multigrid solver, so the tangential interpolation is a trivial operation.

Before each sweep, the data already interpolated from the coarse data (the open circles in Figure 3b) are interpolated normal to each face to define values in the ghost cells (the squares and triangles in Figure 3b) analogously to the extrapolation used for the physical boundary values. Again, for each fine grid point next to a coarse/fine interface, a parabola is defined using the coarse grid value (the open circle) and the two interior values which align with the ghost cell being filled (as in Figure 3a). This polynomial is then evaluated at the location of the ghost cell. This normal interpolation procedure is identical in two and three dimensions.

Note that in the upper right corner of the coarse grid region in Figure 3b, the ghost cell

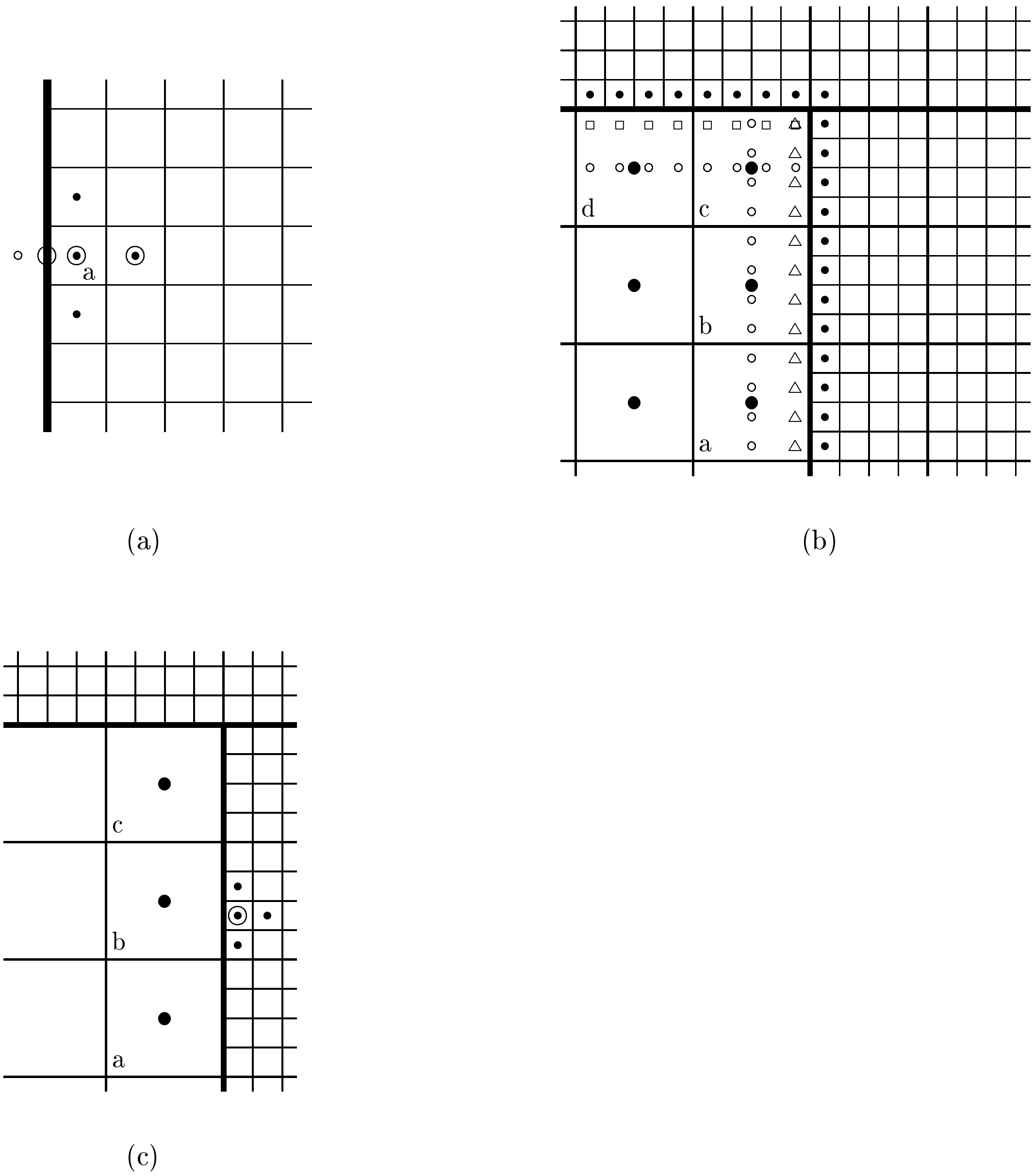


Figure 3: (a) At a physical boundary, interior and boundary values ($\bigcirc$'s) are used to extrapolate to the ghost cell ($\circ$); the ghost value and the other interior values ($\bullet$'s) are used to construct the Laplacian at (a). (b) Locations of coarse grid boundary conditions (●), tangentially interpolated values ($\circ$), fine grid cells ($\bullet$), and ghost cells ($\triangle$'s and $\square$'s). (c) Domain of dependence ($\bullet$'s and ●'s) of the Laplacian at a fine cell ($\bigcirc$) adjacent to the coarse/fine interface.

is marked with a square and a triangle. This illustrates that the ghost cell values are not unique; the square value will be used for computation of the operator immediately above that point, the triangle value will be used for computation of the operator immediately to the right of that point. Different coarse grid values are used to define the square and the triangle values.

The two-stage interpolation procedure described above in effect defines a specialized discretization of the elliptic operator which at the coarse/fine interface uses only interior fine grid data and coarse grid data which does not underlie any fine grids. In fact, the dependence of the ghost cell value on the value at which the elliptic operator is being evaluated changes the relaxation coefficient in the Gauss-Seidel relaxation sweeps. Interpolation of the coarse and fine data onto ghost cell locations is simply a convenience of implementation which allows greater efficiency in the relaxation sweeps and construction of the residual.

The driving concept for this special discretization is that the domain of dependence of the operator at a fine grid point adjacent to a coarse/fine interface should include only fine grid values and those coarse grid values which do not underlie any fine grids. This is shown in Figure 3c, where the points involved in the calculation of the operator at the large open circle are marked by closed circles. This is important because the coarse grid values underlying fine grids are defined as averages of the fine grid values, and using these to define the ghost cell values extends the domain of dependence of the elliptic operator inappropriately.

The resulting solution now satisfies a first-order approximation to the second-order linear operator at the fine cells adjacent to the coarse/fine interface where the stencil sees both the coarse and fine data; a first-order approximation at all physical boundaries, and a second-order approximation everywhere in the interior of the union of grids. However, since the first-order errors are localized at the boundary of the union of grids, the overall scheme is still second-order accurate because of the spectral properties of the discrete solution operator. In particular, these types of localized errors are well represented by eigenfunctions of the discrete elliptic operator that correspond to $O(h)$ eigenvalues of the solution operator.

4 Computational Results

In this section we first identify the questions we will address computationally, in terms of convergence rates and accuracy of adaptive solutions relative to uniform grid solutions. In the second subsection we present results from several convergence studies in two dimensions. (Because the algorithm is identical in two and three dimensions, and the cost of doing convergence studies in three dimensions is so much higher, we demonstrate the convergence behavior in two dimensions only.) In these studies, both the uniform grid and adaptive calculations are shown to be second-order accurate for smooth flows, and the importance of the sync projection is demonstrated.

In the following subsection, we show accuracy results for a more realistic problem, that of a bubble rising in a fluid 1000 times its density. The purpose of this example is to demonstrate that even for flows for which we don't expect second-order convergence because of the presence of steep gradients, if the grids are optimally placed the accuracy of the solution is comparable to that of the solution from a uniform fine grid calculation.

In the fourth subsection we show results from a variable density Navier-Stokes calculation in three dimensions, and comparison of these results with experimental data. Finally, in the fifth subsection, we present some timings of the algorithm in two and three dimensions on a DEC Alpha workstation, and discuss briefly several of the design issues.

4.1 Questions

There are a number of issues we would like to address in numerical testing of the algorithm presented in the previous sections. The first is the straightforward question about convergence rate of the solution,

(Q.1) Is the adaptive method second-order accurate for smooth flows, in the sense that as the base grid varies (with a fixed refinement ratio) both the composite solution on the entire domain, and the solution on the refined region only, converge with second-order accuracy to the exact solution?

In general, the determination of the asymptotic behavior of an adaptive algorithm does not address whether adaptivity actually improves the quality of the solution. Thus a second question is asked:

(Q.2) Does refinement of a region of the domain increase the accuracy of the solution in that region, without worsening the accuracy of the solution in the non-refined regions?

In addressing the first two questions, we will perform a simple test in which a patch is fixed in time in order to evaluate convergence of the solution in the absence of any effects due to the regridding procedure. This is not a typical use of the AMR algorithm; in practice, the flow dynamically evolves in time and the refinement adapts to the features of the flow. Thus, we ask:

(Q.3) When the refinement criteria are such that the refinement adapts to the features of the flow, how do the accuracy and convergence properties of the composite solution compare to those of a uniform fine grid solution?

Second-order accuracy of the solution is only a relevant goal when the flow is sufficiently smooth; even on a single grid the method has slope limiters that locally reduce the advection step to first-order when the gradients are too steep, in order to eliminate oscillations. Thus, for non-smooth flows, we pose the following questions:

(Q.4) Does an appropriate refinement improve the accuracy of the solution even when the flow is not smooth? Are the rates of convergence of the adaptive solution comparable to those of the uniform grid solution even when those rates are less than second-order?

A final question,

(Q.5) Is the sync projection really necessary for accuracy?

is also addressed in the next section. This question is posed here because two of the methods presented in the literature (see [14, 29]) describe adaptive methods with elliptic solves on coarse and fine levels separately, but no elliptic synchronization. In this algorithm great attention is paid to the elliptic synchronization, and we show in the next subsection the reasons why.

4.2 Convergence Studies

The calculations we present here demonstrate second-order convergence of the adaptive method on problems with smooth initial data. These include a problem in which we fix the refinement in time for the purpose of evaluating the convergence of the solution, and another for which the refinement more naturally follows the features of the flow.

Both problems are unsteady. The first is constant density inviscid flow in a doubly periodic box, for which we also demonstrate the importance of doing the sync projection. The second is a viscous vortex merger problem, which demonstrates that the dynamic creation and destruction of grids again maintains the convergence properties of the algorithm.

In summary, with the first problem we address (Q.1)-(Q.2) and (Q.5) for a simple inviscid case; with the second problem we address (Q.1)-(Q.3) for a more interesting viscous case. Question (Q.4) will be addressed in the next subsection.

Problem 1

For the first problem, we compute the errors and convergence rates of the solution for uniform grids, base grids with one level of factor r refinement, $r = 2$ or $r = 4$, and base grids with two levels of factor two refinement. Here we include the latter case to demonstrate that the convergence rate is consistent as grids are recursively nested.

For the adaptive cases, we consider two different measures of the error: first, the error of the full composite grid solution, i.e., the error of the solution over the entire domain, and second, the error of the solution solely in the region of maximum refinement. The importance of each measure depends on the nature of the calculation and purpose of refinement. If the aim is purely to resolve a specific region or feature of the flow as well as possible, and one only cares about the rest of the solution insofar as it affects the refined region, then the latter measure is most relevant. If, however, one's goal is to compute the entire solution using adaptivity to improve the accuracy where errors are large then the former error is more relevant.

The $r = 2$ and $r = 4$ cases were studied in [21] using the L_2 norm of the error on the patch only; the norms and rates obtained here are comparable, and the conclusions identical. As in [21], for these calculations only we do not use the slope limiters for the advective derivatives.

The initial data for the first problem are given on the periodic unit square by

$$u(x, y) = 1 - 2\cos(2\pi x)\sin(2\pi y),$$

$$v(x, y) = 1 + 2\sin(2\pi x)\cos(2\pi y).$$

The exact solution for these initial conditions is

$$u(x, y, t) = 1 - 2\cos(2\pi(x - t))\sin(2\pi(y - t)),$$

$$v(x, y, t) = 1 + 2\sin(2\pi(x - t))\cos(2\pi(y - t)),$$

$$p(x, y, t) = -\cos(4\pi(x - t)) - \cos(4\pi(y - t)).$$

A total of twelve calculations are presented in Table 1; each has a base grid of $n \times n$ cells. The row labeled "Uniform" represents uniform grid calculations; the rows labeled

$r = 2$ or $r = 4$ represent single level adaptive calculations, with refinement ratio of 2 or 4, respectively, and the row labeled $r = 2, 2$ represents the two levels of factor two refinement. In each adaptive case there is a single patch at each level, and the finest resolution covers the square from (0.25,0.25) to (0.5,0.5) in physical space.

The calculations are run to $t = 0.5$; the time step is determined each level 0 time step using the CFL number 0.75. In Table 1 we present the L_2 and L_∞ norms of the errors and convergence rates of u as calculated both on the region (0.25,0.25) to (0.5,0.5) (labeled "Patch") and on the entire domain (labeled "Domain"). The error is defined as the difference from the exact solution; the L_2 norm of the error e in u is defined by

$$||e||_2 = \sqrt{\sum_{i,j} h^2 e_{i,j}^2}.$$

Note that when calculating norms of the error over the entire domain, we do not include those coarse cells covered by fine cells, and the h above depends on level. The rate between the two columns of error norms is defined as $\log_2(E_l/E_r)$, where E_l and E_r are the errors shown in the columns on the left and right sides, respectively.

Patch	L_2					L_∞				
	$n = 32$	Rate	$n = 64$	Rate	$n = 128$	$n = 32$	Rate	$n = 64$	Rate	$n = 128$
Uniform	2.668e-3	2.04	6.505e-4	2.01	1.620e-4	2.024e-2	2.04	4.923e-3	2.02	1.211e-3
$r = 2$	2.290e-3	1.92	6.048e-4	1.93	1.591e-4	1.843e-2	1.90	4.933e-3	1.88	1.344e-3
$r = 2, 2$	2.580e-3	1.99	6.481e-4	1.98	1.641e-4	2.432e-2	1.94	6.339e-3	2.04	1.539e-3
$r = 4$	2.368e-3	1.92	6.265e-4	1.91	1.663e-4	2.726e-2	2.04	6.621e-3	1.96	1.698e-3

Domain	L_2					L_∞				
	$n = 32$	Rate	$n = 64$	Rate	$n = 128$	$n = 32$	Rate	$n = 64$	Rate	$n = 128$
Uniform	3.280e-3	2.02	8.073e-4	2.00	2.021e-4	2.120e-2	2.10	4.958e-3	2.03	1.213e-3
$r = 2$	3.245e-3	2.00	8.123e-4	2.00	2.033e-4	2.559e-2	1.82	7.254e-3	2.03	1.772e-3
$r = 2, 2$	3.153e-3	2.02	7.790e-4	2.00	1.947e-4	2.692e-2	1.92	7.125e-3	1.90	1.903e-3
$r = 4$	3.305e-3	2.00	8.263e-4	2.01	2.054e-4	2.776e-2	1.90	7.413e-3	1.88	2.016e-3

Table 1: L_2 and L_∞ errors and convergence rates for u in constant density inviscid calculation. Here, slope limiters are off. The upper table shows errors only in the region of maximum refinement; the lower shows composite errors over the entire domain.

From this table we draw two conclusions:

- The calculated solution is converging to the exact solution with second-order accuracy in the L_2 and L_∞ norms for the uniform grid and the adaptive calculations.
- When the refined grid is placed non-optimally, as in these calculations, the error in the refined patch is comparable to the error at the resolution of the base grid. The use of refinement is not improving the accuracy.

The second observation is not unexpected. For this problem the refined patch is placed very poorly. At the final time flow has entered the patch and passed out the other side; no feature of the flow has been resolved over the finer grid. The adaptive algorithm cannot recover the accuracy that has been lost on the coarser grid. The second example provides a more realistic assessment of the performance of the methodology.

In order to answer (Q.5), we ran two of the above adaptive cases again, but with no sync projection. These results are presented in Table 2. As we can see, the $r = 2$ solution generated without a sync projection reduces to first-order accurate in the L_2 norm as the resolution increases, and the L_∞ norm reduces to $O(1)$; for $r = 4$ the solution actually becomes less accurate in both norms in going from a base grid of $n = 64$ to $n = 128$. Not surprisingly, the maximum error is located inside the refined region adjacent to the coarse/fine interface. We conclude that the sync projection is necessary to maintain second-order accuracy of the overall adaptive method. (We note here that the loss of accuracy associated with the lack of synchronization between levels is also documented in [29], though for a less dramatic case.)

Patch	L_2					L_∞				
	$n = 32$	Rate	$n = 64$	Rate	$n = 128$	$n = 32$	Rate	$n = 64$	Rate	$n = 128$
$r = 2$	1.324e-2	1.80	3.796e-3	1.01	1.886e-3	3.716e-2	1.09	1.740e-2	-.23	2.041e-2
$r = 4$	2.249e-2	1.67	7.049e-3	-1.41	1.880e-2	5.005e-2	0.51	3.524e-2	-3.23	3.309e-1

Domain	L_2					L_∞				
	$n = 32$	Rate	$n = 64$	Rate	$n = 128$	$n = 32$	Rate	$n = 64$	Rate	$n = 128$
$r = 2$	1.366e-2	1.96	3.508e-3	1.80	1.009e-3	3.716e-2	1.09	1.740e-2	-.23	2.041e-2
$r = 4$	1.447e-2	1.89	3.902e-3	-.54	5.664e-3	5.005e-2	0.51	3.524e-2	-3.23	3.309e-1

Table 2: L_2 and L_∞ errors and convergence rates for u in constant density inviscid calculation. Here slope limiters are off and no sync projection is used.

Problem 2

The first example with a single fixed refinement patch demonstrates the convergence properties of the method without the complexities associated with regridding operations. However, it is also atypical because the refinement is not adapting to the flow. Our next example provides a more realistic demonstration of the methodology. This problem is a constant density four-vortex calculation. To initialize the velocity field for this problem, we first place four vortices in the unit square, centered at (0.5,0.5), (0.59,0.5), $(0.455, 0.5 + .45 * \sqrt{3})$, and $(0.455, 0.5 - .45 * \sqrt{3})$. The first vortex has strength -150, the "outer" three vortices each have strength 50, and the profile for each, centered around (x_i, y_i), is

$$\tfrac{1}{2}(1 + \tanh(100(0.03 - r_i))$$

where $r_i = \sqrt{(x - x_0)^2 + (y - y_0)^2}$. Here and in all further calculations the slope limiters are used.

We use the vorticity as the source term for a Poisson equation, in which the field being solved for is the stream function. (We can use the same multilevel projection methodology already in place for the velocity projections to solve this Poisson equation.) The stream function is computed with homogeneous Dirichlet boundary conditions on the unit square, and the velocity field is calculated from the stream function. This velocity field is then projected to ensure that it approximately satisfies the discrete divergence constraint.

The calculations are run to $t = 0.25$ with a CFL number of 0.9. For the adaptive calculations, the refinement criterion is that the magnitude of vorticity be greater than 5% of its maximum. The viscosity is set to $\mu = 0.0001$.

The convergence results from a total of three uniform grid calculations and three adaptive calculations are shown in Table 3. The uniform grid calculations have base grids of 64×64, 128×128, and 256×256. The adaptive calculations each have a single level of factor four refinement, and have base grids of 16×16, 32×32 and 64×64. A time sequence of the adaptive calculation with a base grid of 64×64 is shown in Figure 4 showing both the evolution of the flow as the outer vortices orbit the inner vortex and the dynamic adaptation of the grids to follow the evolving vortical structure.

Here there is no exact solution, so we use the solution computed on a 1024×1024 uniform grid as a reference solution, the difference from which defines the error of the coarser calculations. Because the gridding is different in each calculation, we present only the error as defined over the entire domain.

Domain	L_2					L_∞				
	64	Rate	128	Rate	256	64	Rate	128	Rate	256
Uniform	1.558e-2	1.98	3.940e-3	1.89	1.067e-3	2.701e-1	2.00	6.739e-2	2.72	1.020e-2

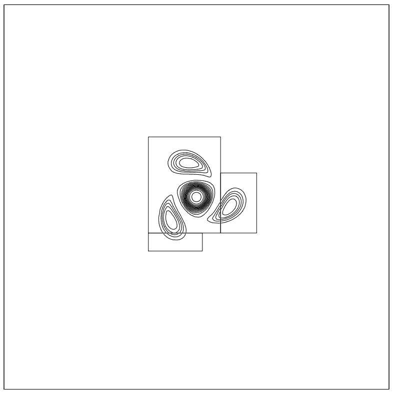
Domain	L_2					L_∞				
	16	Rate	32	Rate	64	16	Rate	32	Rate	64
$r = 4$	1.560e-2	1.98	3.944e-3	2.04	9.607e-4	2.648e-1	2.00	6.623e-2	2.21	1.433e-2

Table 3: L_2 and L_∞ errors and convergence rates for u in constant density viscous vortex merger calculation.

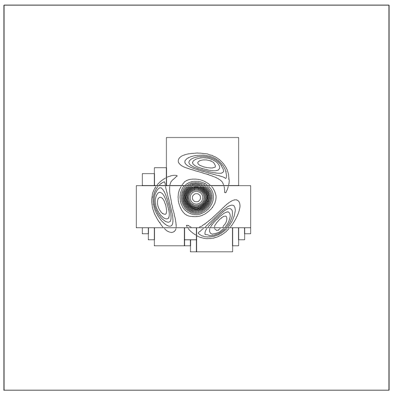
This problem clearly demonstrates second-order convergence of the algorithm. Furthermore, when the refinement criteria allow the method to adapt to the flow, the algorithm can essentially recover the accuracy of the equivalent fine grid.

4.3 Axisymmetric Bubble Rise

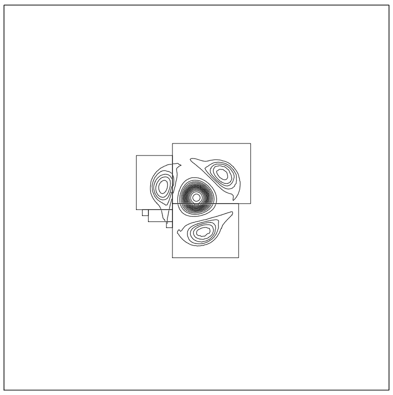
The next set of calculations compares the errors of a composite grid solution with the errors of a uniform fine grid solution for an under-resolved flow with a very strong density contrast. The extreme density contrast and large density gradients were chosen so that the flow is not in the asymptotic, second-order regime for the basic projection algorithm. The goal here is to show that with the adaptive algorithm one can achieve accuracy comparable to a uniform fine grid even for flow that is not well-resolved at the finest level of refinement. We present data from axisymmetric calculations of a light bubble rising under gravity in a constant density background. The regridding criterion, which flags coarse grid cells for refinement whenever the density is below a critical value, is such that all of the bubble is always at the finest resolution.



(a) $t = 0.05$



(b) $t = 0.15$



(c) $t = 0.25$

Figure 4: Contour plots of vorticity for the four-vortex calculation with a base grid of 64×64 and one level of factor 4 refinement at times t= 0.0, 0.15, 0.25.

The initial conditions are a zero velocity field and a density field

$$\rho(r,z,t=0) = \frac{\rho_1+\rho_2}{2} + \frac{\rho_1-\rho_2}{2}\tanh(5000(\sqrt{r^2+(z-1)^2}-R_0))$$

in a .01 × .02 domain where $R_0 = .0025$ is the radius of the bubble. Here $\rho_1 = 999.2$ and $\rho_2 = 1.225$, the densities of water and air, respectively, in MKS units. The viscosity $\mu = 0.0011377$ is that of water. Contour plots of the time evolution of the density field from the calculation with a base grid of 16 × 32 and one level of $r = 4$ are shown in Figure 5; these show the rise of the bubble and the moving fine grid. Contour plots from the uniform grid calculation at the same times are indistinguishable.

Here again we do not have an exact solution; instead the calculation is first done on a uniform 256 × 512 grid, and this is taken as the reference solution, the difference from which defines the error. Shown in Table 4 are the L_2 norms of the errors in ρ, u and v, evaluated on the entire domain, from several calculations. These calculations include three uniform grid calculations (with base grids of 16 × 32, 32 × 64, and 64 × 128), and three adaptive calculations, $r = 2$ using base grids of 16 × 32 and 32 × 64, and $r = 4$ using a base grid of 16 × 32. Note that the calculations along the same diagonal have the same resolution at the finest level. Only the error relative to the uniform 256 × 512 calculation is shown here. As expected because of the steep density gradient, the solution is not converging with second-order accuracy.

	ρ			u			v		
	16 × 32	32 × 64	64 × 128	16 × 32	32 × 64	64 × 128	16 × 32	32 × 64	64 × 128
Uniform	.3377	.2281	.1330	6.025e-5	2.921e-5	1.116e-5	8.460e-5	2.967e-5	1.083e-5
$r = 2$	.2315	.1343		2.921e-5	1.116e-5		2.936e-5	1.081e-5	
$r = 4$	.1362			1.126e-5			1.135e-5		

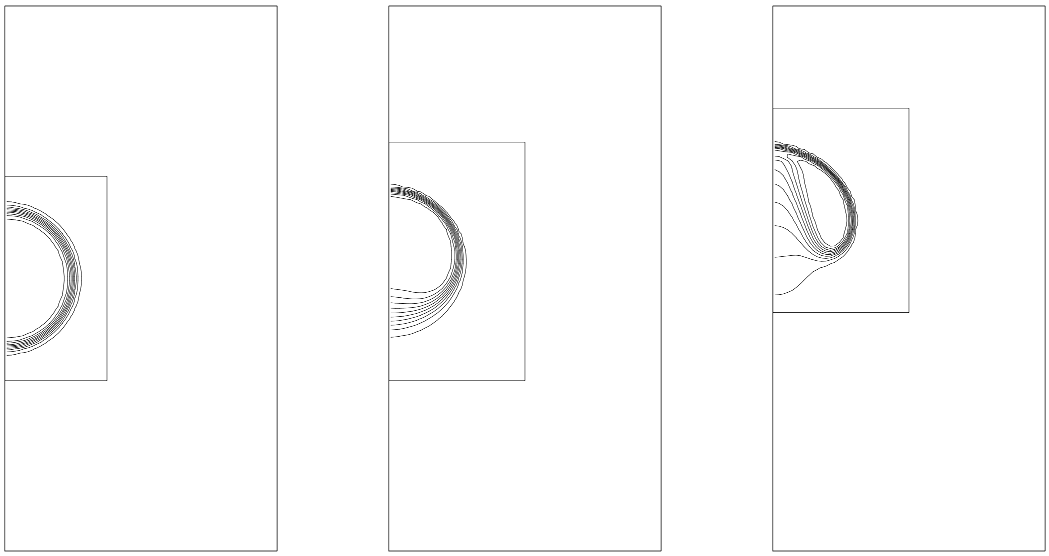
Table 4: L_2 errors in ρ, u and v on the entire domain as calculated using a uniform 256 × 512 grid as a reference solution.

We first note that for this flow the density is converging at less than first-order accuracy and the velocity is converging at approximately $O(h^{1.3})$ for the uniform grids. Nevertheless, the adaptive computations are resolving the flow with the same global accuracy as the uniform grids of the same resolution as the finest grids in the refinement.

4.4 Three-Dimensional Shear Layer

Finally, we present a three-dimensional variable density shear layer calculation. The data for the problem were chosen to model the conditions studied by Brown and Roshko [12] and Konrad [18] who were studying the effects of density variation on low speed shear layers. Although the experimental shear layer was unforced, we have added forcing using frequencies taken from Monkewitz and Huerre [22] as was done by Chien et al. [13] for their two-dimensional simulations of shear layers.

The calculation was performed in a box with dimensions 512 × 128 × 384. The base grid was 32 × 8 × 24, and there were two levels of refinement, the first by a factor of 4 and the second by a factor of 2, for an effective resolution at the fine level of 256 × 64 × 96, with



(a) t = 0.0　　(b) t = 0.01　　(c) t = 0.022

Figure 5: Contour plots of density for the bubble calculation with a base grid of 16×32 and one level of factor 4 refinement at times t= 0.0, 0.01, 0.022.

$\Delta x_{finest} = 2$. The boundary conditions were: inflow at $x = 0$, outflow at $x = 512$, slip walls at $y = 0, 128$, and no-slip walls at $z = -192, 192$.

The computations presented here were performed at Reynolds number 2×10^4 based on the mean flow rate and the length of the computational domain. The flow was initialized to be $U(x, y, z, t = 0) = (u_0(z), 0, 0)$ with

$$u_0(z) = \frac{U_1 + U_2}{2}(1 + \lambda_v \tanh(\frac{2z}{\delta_0}))$$

and $\lambda_v = \frac{U_1 - U_2}{U_1 + U_2}$, where $U_1 = 1.451, U_2 = .549$ and $\delta_0 = 6$. The density was initialized in the domain to be $\rho(x, y, z, t = 0) = (1 + .02R)\rho_0(z)$ where

$$\rho_0(z) = \frac{\rho_1 + \rho_2}{2}(1 - \lambda_r \tanh(\frac{2z}{\delta_0})),$$

with $\lambda_r = \frac{\rho_2 - \rho_1}{\rho_1 + \rho_2}$, where $\rho_1 = 1$, $\rho_2 = 7$ and R was a random fluctuation from -1 to 1 intended to break the inherent symmetries in the flow. These profiles are the same profiles as were used by Chien et al. [13] except for the inclusion of the random perturbation in the density field.

The inflow velocity profile as a function of time was

$$U(x = 0, y, z, t) = (1 + \Sigma_{i=1}^{10} m_i \sin(f_i t))u_0(z - z_{pert}),$$

where $z_{pert} = 0.1 \sin(f_1 t) \sin(.22089323y)$. The frequencies were $f_1 = .219$, and $f_i = f_{i-1}/i$ for $2 \le i \le 10$, and the magnitudes were $m_1 = .01$, $m_2 = .75m_1$, $m_3 = .55m_1$, $m_4 = .44m_1$, and $m_i = 1.7m_1/i$ for $5 \le i \le 10$. The density of the fluid flowing in through $x = 0$ was defined to be $\rho(0, y, z, t) = \rho_0(z - z_{pert})$. These inflow profiles are also taken from the work of Chien et al.; however, we have added a transverse perturbation in the form of z_{pert} to introduce three-dimensional structure into the flow. This perturbation of the inflow data is intended to mimic a mild "flutter" of the splitter plate used in the experiments. For this computation the flutter oscillated in time with a zero mean and had a maximum deflection of 5% of the finest mesh spacing. Although this perturbation is small, we found that without the introduction of some three-dimensional perturbation the flow evolved to an essentially two-dimensional configuration with very small transverse velocities for the size of computational region considered here.

The flow requires about 100 coarse grid time steps for the initial perturbations to pass through the domain and the pattern of vortex formation to become established. For these initial cycles we adjusted the error criteria so that no level 2 grids were formed and so that level 1 grids followed the structures. We then set the error criteria so that level 2 grids would be formed in the region where the two fluids were mixing, and ran the computation for an additional 225 level 0 time steps. By step 200 all of the vortical structures in the problem had been resolved on the finest level mesh from their inception.

In Figures 6a-d we show a time sequence of density in the x-z cross-section centered spanwise in the domain. These "snapshots" are taken at intervals of 10 level 0 time steps; the times are: (a) $n = 205, t = 1585$, (b) $n = 215, t = 1630$, (c) $n = 225, t = 1673$, (d) $n = 235, t = 1715$. Recall that eight level 2 time steps are taken for each level 0 time step. Figures 6e-f show spanwise averages of the density at $n = 205$ and $n = 225$ in order to calculate the spreading rate. Although it is difficult to precisely define an envelope around the spreading shear layer, the visual spreading rate, δ_{vis}, calculated from these profiles is consistent with the experimentally computed value of 21% [12] superimposed on our data in these figures. In Figure 7 we show a 3-d rendering of the magnitude of vorticity at $n = 205$ to demonstrate the spanwise structure of the flow. The region shown in this figure covers the full distance in x but not the full z-extent. We note that although the transverse inflow perturbations are quite small substantial three dimensional structures do develop.

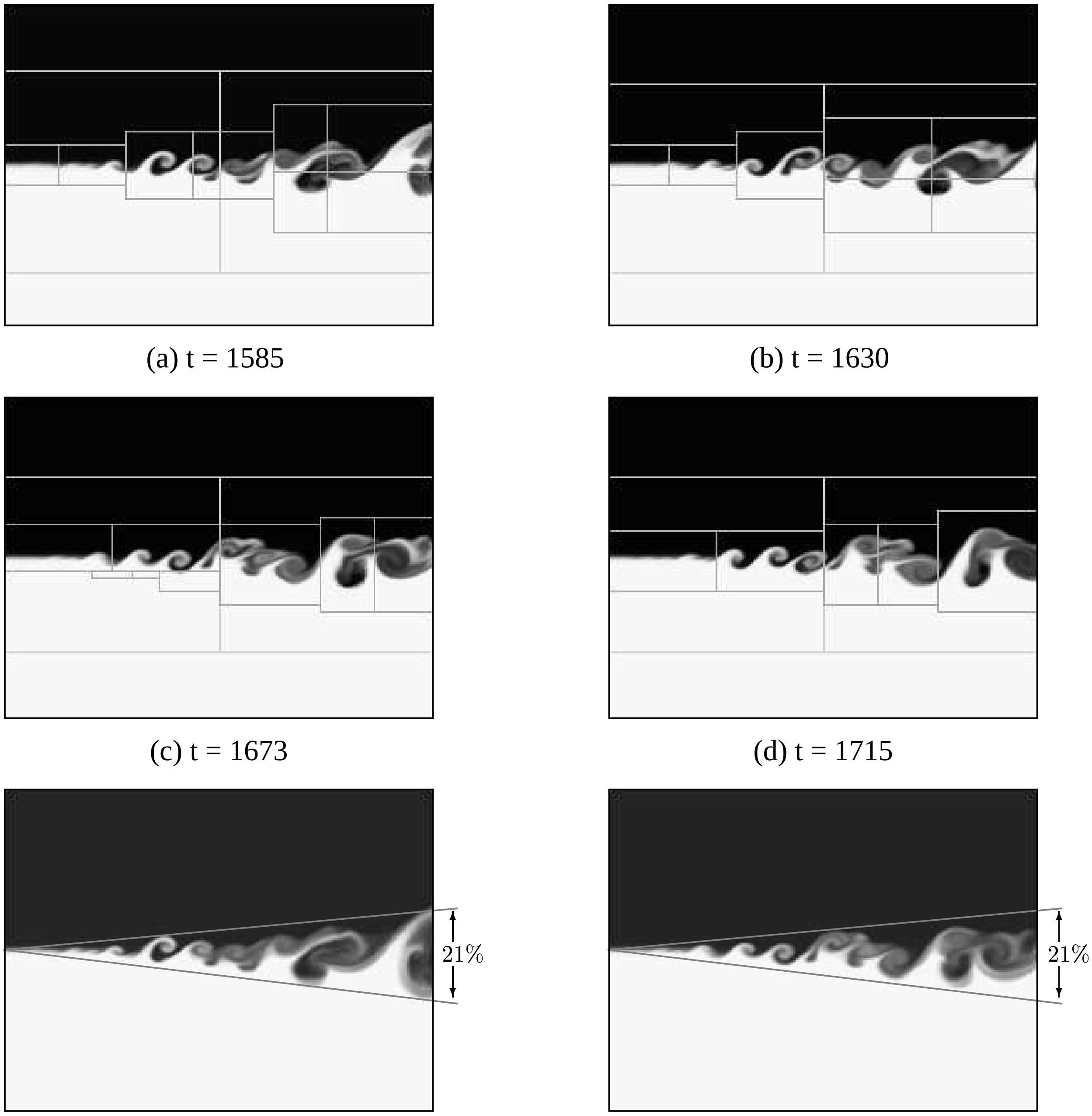
Finally, in order to have a more quantitative comparison with the experimental data we accumulated flow statistics from $n = 200$ to $n = 325$. In particular, we computed time- and spanwise-averaged values of the mean x-velocity and density and the density perturbation. (The spanwise averaging was across the entire domain.) We note that although the time interval corresponds to 1000 steps on the finest grid we only accumulated data at the end of coarse grid time steps so that the statistics include only 125 time samples. These are scaled by U_1 for the mean velocity and ρ_2 for the mean density and density perturbation. In Figure 8 the profiles are plotted on the same axes as the experimental data from [12] (for the mean values) and [18] (for the fluctuating density); the experimental values are shown as mean values with error bars which represent the observed spread in the data. The velocity profile matches the experimental data well. The density profile lies above the experimental data in the center of the profiles. By examining the statistical data earlier in the computation we found that the density profile was converging more slowly over time than the velocity profile. Consequently this disagreement may be an artifact of the small number of temporal samples. (Chien et al. [13] report needing several thousand samples to compute accurate statistics.) The perturbational density profile matches the overall structure of the experimental profile. The peak is well approximated and the overall shape of the profile is correct, including the flattening and subsequent drop-off to the right of center although the values in the center are somewhat high.

4.5 Performance

In this subsection we look at performance data and associated issues for the adaptive algorithm discussed above. In the first part of the section we present timings for one of the test problems shown in Section 4.2. In the second part we discuss the impact of some of our design choices on the accuracy and performance of the algorithm.

Timings

Here we present timings on a single processor of a four-processor DEC Alpha for the constant density four-vortex problem presented in Section 4.2. In addition to the uniform 256×256 calculation and the calculation with a 64×64 base grid and $r = 4$, we present timings from two additional calculations: 128×128 base grid with $r = 2$, and 64×64 base grid with two refined levels with $r = 2$. Each calculation has the same resolution at the finest level. Presented in Table 5 are the number of CPU seconds required to complete the calculation, the number of total cells advanced, and the CPU time per cell advanced as measured in μs / cell. The number of total cells advanced is the sum over all levels of the number of cells advanced at that level. The ratio of CPU time per cell advanced is interesting for evaluating how the cost of the calculation scales with the size of the problem and refinement strategy.



(a) t = 1585 (b) t = 1630

(c) t = 1673 (d) t = 1715

(e) t = 1585 (f) t = 1673

Figure 6: (a)-(d): Time sequence of density in x-z cross-section at y=64 with the level 1 and level 2 grids superimposed. (e)-(f): Spanwise average of density; superimposed for comparison is the experimentally observed visual spreading rate, $\delta_{vis} = 21\%$. In each figure the lighter fluid is on top.

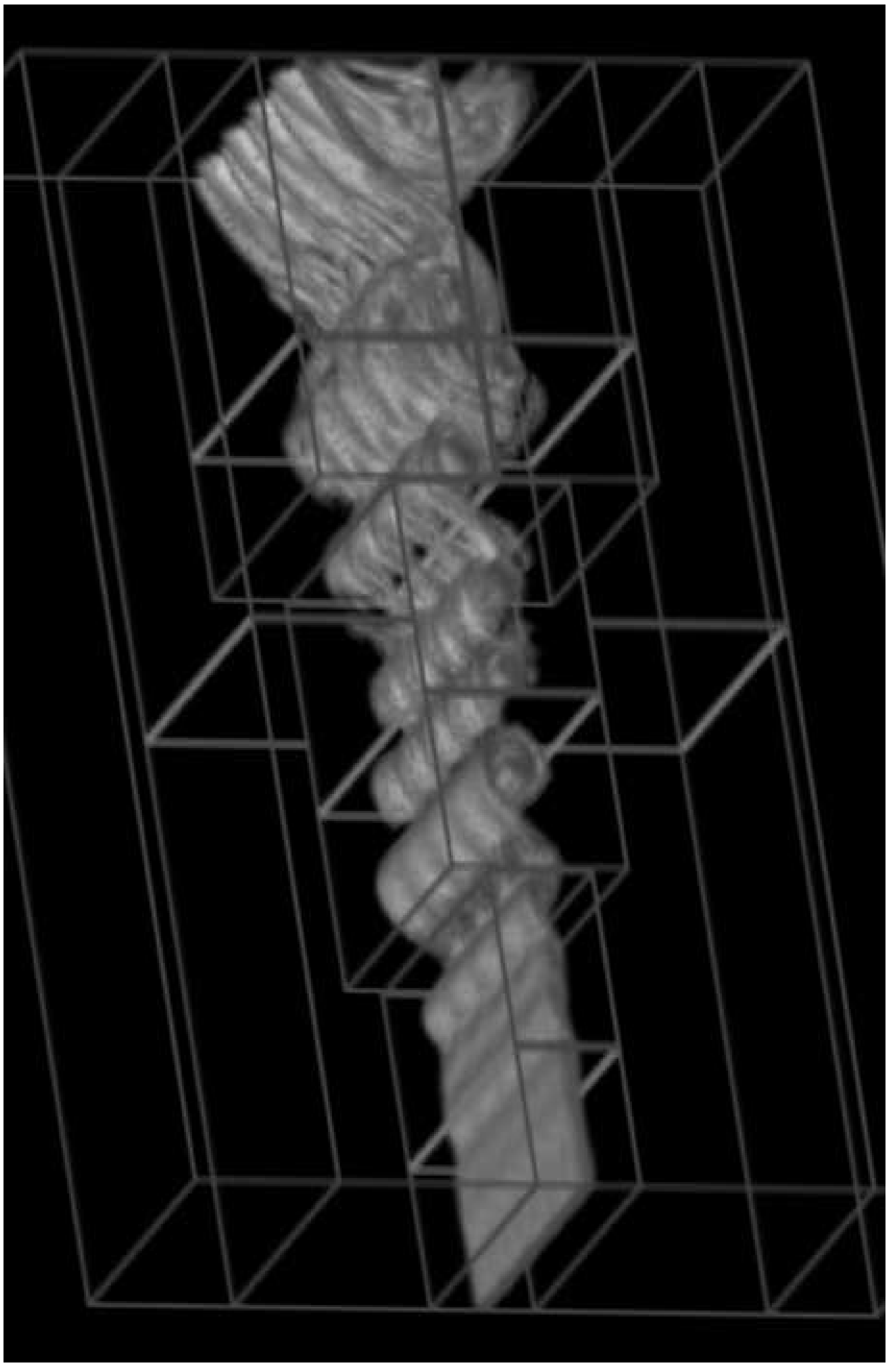


Figure 7: Three-dimensional rendering of vorticity at $t = 1585$ with the level 1 and level 2 grids superimposed. The domain is cropped slightly in the z-direction.

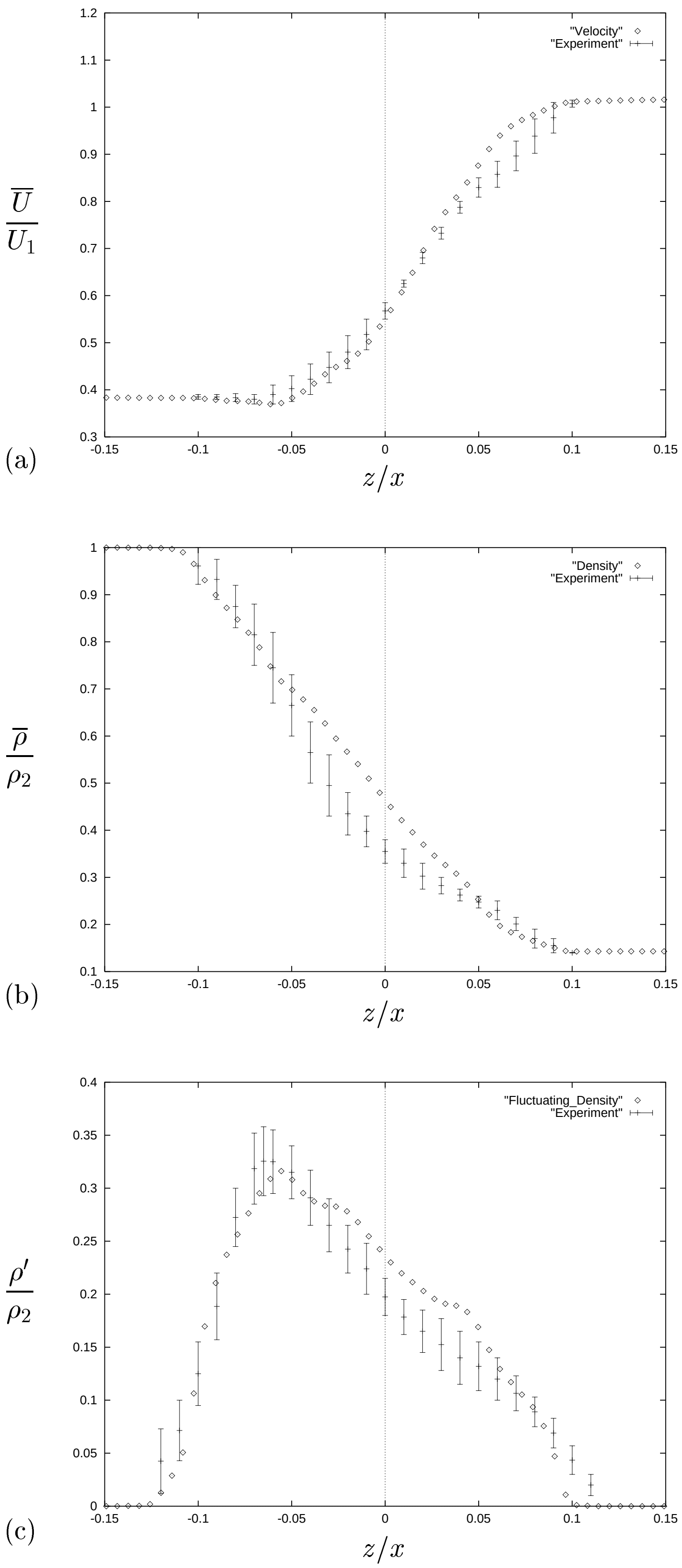


Figure 8: Mean velocity and density profiles and fluctuating density profile for the three-dimensional shear layer. The error bars on the mean profiles denote the experimental data band from Brown and Roshko [12]; the error bars on the fluctuating density profile denote the experimental band from Konrad [18].

	CPU Time		Cells Advanced
	Total(s)	μs/cell	Number
Uniform 256×256	1389.	161	8650752
Base Grid $128 \times 128, r = 2$	378.6	211	1794496
Base Grid $64 \times 64, r = 2, 2$	256.1	236	1085056
Base Grid $64 \times 64, r = 4$	241.3	195	1239232

Table 5: Timings for uniform grid and refined grid calculations on a single processor of a four-processor DEC Alpha for the four-vortex problem presented in Section 4.2.

The timings presented above show that the adaptive version of the algorithm with a single level of $r = 4$ increases the cost per cell by approximately 20%; the single level of $r = 2$ increases the cost per cell by approximately 30%. Roughly speaking, this suggests that if less than 80% of the problem domain requires the highest resolution, using a single level of $r = 4$ refinement is faster in overall time than a uniform fine grid calculation (as well as requiring less total memory).

It is interesting that the three-level calculation, although it advances the fewest number of cells, takes longer than the $r = 4$ calculation. (There are fewer total cells advanced because the regridding algorithm allows for less total area at the finest resolution using two factors of refinement by 2 than one factor 4.) The reason this occurs is that although both calculations spend approximately the same amount of time in the MAC projection and nodal level projection, the three-level calculation spends almost twice as long in the sync projections and sync solves. In both cases, for N time steps at level 0, there are N level 0-1 sync projections and level 0 MAC sync solves; for the three-level case, however, there are additionally $2N$ level 1 sync projections and sync solves. While the level 0-1 sync projection is less costly for $r = 2$ than for $r = 4$, the expense of the additional level outweighs this factor.

The efficiency of the multigrid solvers is the single largest factor in determining the overall efficiency of the code. In the timings discussed above, the multigrid tolerances were specified to be 10^{-12} for the MAC projections, 10^{-8} for the MAC sync solves, 10^{-12} for the nodal level projections, and 10^{-10} for the nodal sync projections. These were the ratio of the final residual to original residual in the L_∞ norm for the elliptic equation to be considered solved. The difference between the tolerances for the level solves and those for the sync solves is dictated by the requirements of solvability; similarly, in practice the tolerances for the sync solves at levels $\ell > 0$ must be smaller than those at level 0 for the level 0 sync solves to be solvable on a domain with Neumann or periodic boundaries.

Design Issues

In this section we address the question of why subcycling in time is worthwhile in terms of accuracy and efficiency of the algorithm. In the case of no-subcycling, such as used by Minion[21], there are no elliptic synchronization operations necessary because the elliptic solves are done as composite solves.

A direct comparison of approaches is not practical. We would require substantial additional implementation effort and software redesign to implement a single step algorithm.

Also, since our implementation has substantial additional capability compared to Minion's code, a direct timing comparison with his code is not possible. We can, however, make a reasonable estimate of the relative performance of our algorithm versus a single step version based on the same methodology by comparing timings for the nodal projection. (A full multi-level nodal projection is used for initialization; hence we can measure its performance.)

Before discussing the detailed timing it is worth reviewing the pattern of nodal projections during a time step with subcycling. For a three-level computation, during a coarse time step the algorithm requires one level 0 nodal projection, r level 1 nodal projections, and r^2 level 2 nodal projections. The synchronization requires r multilevel solves over levels 1 and 2 and one multilevel solve over levels 0 and 1. A single time step algorithm requires one multilevel time step over all three levels for each time step; however, r^2 steps are required to reach the same final time. We note also that because of the complication of the stencils at coarse–fine interfaces, the multilevel solves are more costly than solves at a single level.

The problem we have chosen for comparison is a three-dimensional shear-layer similar to the example of the previous section with a fixed pattern of grids. For this the base grid was $40 \times 8 \times 24$ with two levels of $r = 4$ refinement. From the initialization we measured the time for a single 0-1-2 projection as 258 seconds. Thus, sixteen steps of a single step algorithm would require 4128 seconds for the nodal projections. For a single step of the subcycling, the code required 2606 seconds for nodal projections. Thus, the nodal projections for a nonsubcycling version of the algorithm would require approximately a 60% increase in computational time compared to the subcycling version. Since the execution pattern is similar for the MAC projections and the advection steps, we would expect this comparison to be a reasonable estimate for a complete non-subcycling version of the algorithm.

We also note that the use of subcycling versus nonsubcycling has a non-negligible effect on accuracy for the types of advection schemes we are using. In particular, these types of explicit upwind methods perform best at CFL numbers approaching one. For one-dimensional advection tests, dropping the time step by a factor of four increases the error by approximately 20% for a discontinuous profile and approximately 50% for a smooth profile on very coarse grids. On finer grids the disparity is increased so that for a discontinuous profile the error is a factor of two larger and for a smooth profile the error is larger by a factor of four. Thus, smaller time steps lose sufficient accuracy that a factor of two finer spatial grid is required to achieve the same accuracy in the solution. Additional reductions in the CFL further reduce the accuracy of the scheme. Although this effect may be reduced when additional physical processes are introduced, we nevertheless expect subcycling to produce better results on advectively dominated flows than a nonsubcycling version. In particular, as illustrated in the previous section, on unrefined portions of the domain our algorithm retains (at least) the accuracy of a uniform coarse grid solution at a comparable resolution. Without subcycling we would expect a degradation in the unrefined portions of the flow.

5 Conclusions and Future Work

We have described here a conservative adaptive projection method for time-dependent incompressible flow which conserves advected quantities and maintains free-stream preservation across coarse/fine interfaces. The levels in the adaptive mesh hierarchy are refined in both space and time. The fractional step character of the projection algorithm requires that we solve hyperbolic, parabolic and elliptic PDE's in an adaptive framework at different stages of the algorithm. The ability of the methodology to handle these prototype equations allows for generalization to a much broader set of equations governing low Mach number flows with more realistic physics.

Future directions for this work include the following: improvement to the multigrid solvers for greater efficiency in the case of strongly varying coefficients (i.e. density jumps) and optimization for running on vector and/or parallel computing platforms; development of partial refinement strategies, in which the grid is refined only in one or two coordinate directions; and extension to a quadrilateral grid framework for body-fitted gridding. In addition, we are exploring the development of automatic error estimation techniques that can effectively control the grid placement. This methodology will also be used to study and validate numerical models of sub-grid-scale processes in various physical applications in order to understand their scale-dependence.

Finally, we are extending the algorithm to include a more general divergence constraint $\nabla \cdot (\sigma U) = Q$. This will provide a framework for modeling more general low Mach number flows such as low speed combustion and the anelastic model of the atmosphere.

Acknowledgements

We would like to thank William Crutchfield, Charles Rendleman and Marcus Day for their development of the cell-centered multigrid software. We would also like to thank Rick Pember for his patient alpha testing of the code described herein.

Appendix: Projection Stencils

Restriction, divergence, and the elliptic operator $L_\rho \phi$ are all defined at nodes of the grid and thus involve complicated stencils at coarse/fine interfaces. We build up these stencils using a finite-element assembly process involving the coarse and fine cells immediately surrounding each interface node. The spatial extent and general form of the stencils in 2-d is shown in Figure 1 for a refinement ratio $r = 4$.

The finite-element basis functions for the 2-d nine-point stencils have a bilinear form on each cell, the 2-d five-point stencils are linear on each of two triangles in each cell, and the 3-d seven-point stencils do not have an obvious geometric interpretation—they are essentially an extension of the 2-d five-point stencils into three dimensions. The details of integration within each cell, however, are not of primary interest to us here. What we need is the contribution each cell makes to a difference stencil, and a procedure for assembling the contributions with the appropriate weights for each interface node.

The divergence stencils in 2-d for uniform parts of the grid look like

$$[D] = \left(\frac{1}{2\Delta x} \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}, \frac{1}{2\Delta y} \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \right),$$

where the velocity components are defined on the four cells surrounding each node. (The gradient stencil $[\,\overline{G}\,]$ is just the transpose of this, taking a scalar quantity on nodes and returning a vector quantity on cells. Because the gradient stencil covers only one cell it, like interpolation, does not require special treatment at interfaces.) Breaking $[D]$ up into individual cells is trivial. The contribution of the cell value to the lower right of the node, for example, is

$$[D_{+-}] = \left(\frac{1}{2\Delta x} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \frac{1}{2\Delta y} \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix} \right).$$

With the contributions from the other three cells we then have

$$[D] = [D_{--}] + [D_{-+}] + [D_{+-}] + [D_{++}]$$

in the grid interiors. These stencils for divergence are correct for both the bilinear and linear triangular basis functions, and extend to 3-d in a straightforward manner. For axisymmetric calculations a small correction is required; this will be presented at the end of this appendix.

At coarse/fine interfaces we define divergence only at coarse nodes. The sum over the cells covered by the basis function can be expressed as follows,

$$[D]_o = \frac{1}{w_o} \left[\sum_{c \in S_c(o)} [D_c]_o + \frac{1}{r^d} \left(\sum_{f \in S_f(o)} [D_f]_o + \sum_{p \in F(o)} \chi(p) \sum_{f \in S_f(p)} [D_f]_p \right) \right].$$

Here d is the spatial dimension (2 or 3), o represents the coarse interface node (at (o_x, o_y) in 2-d) where we are evaluating the stencil, and $F(o)$ is the set of fictitious fine nodes p (at (p_x, p_y) in 2-d) on the coarse/fine interface between o and its neighboring coarse nodes.

$S_f(o)$ and $S_c(o)$ are the sets of fine and coarse cells surrounding o, respectively. In 2-d the weight function

$$\chi(p) = \frac{(r - |p_x - o_x|)(r - |p_y - o_y|)}{r^2}$$

captures the linear form of the basis function along each interface edge. (Note that this weight function resembles but is not identical to the bilinear basis function ψ_o. $\psi_o = \max(\chi, 0)$ in the coarse grid and along the interface, but within the fine grid ψ_o drops to 0 across the width of a single fine cell.) In 3-d the weight function has a similar form:

$$\chi(p) = \frac{(r - |p_x - o_x|)(r - |p_y - o_y|)(r - |p_z - o_z|)}{r^3}.$$

The factor w_o represents a weight for the entire stencil. In 2-d it is the integral of the basis function ψ_o normalized so that a coarse node has weight 1:

$$w_o = \sum_{c \in S_c(o)} \frac{1}{4} + \sum_{f \in S_f(o)} \frac{1}{4r^2} + \sum_{p \in F(o)} \chi(p) \sum_{f \in S_f(p)} \frac{1}{4r^2}.$$

In 3-d, analogously,

$$w_o = \sum_{c \in S_c(o)} \frac{1}{8} + \sum_{f \in S_f(o)} \frac{1}{8r^3} + \sum_{p \in F(o)} \chi(p) \sum_{f \in S_f(p)} \frac{1}{8r^3}.$$

For the linear operator L_ρ we have several different stencils. Bilinear basis functions in 2-d give a nine-point stencil derived from integrals over four adjacent cells. Each of these four integrals gives a similar "unit cell" contribution, such as the following for the lower right cell:

$$[(D\sigma G)_{+-}] = \frac{\sigma^{(x)}_{+-}}{6\Delta x^2} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & 2 \\ 0 & -1 & 1 \end{bmatrix} + \frac{\sigma^{(y)}_{+-}}{6\Delta y^2} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & -1 \\ 0 & 2 & 1 \end{bmatrix}.$$

As with divergence, we build up the stencil in uniform parts of the mesh by simply adding four unit cells together:

$$[(D\sigma G)] = [(D\sigma G)_{--}] + [(D\sigma G)_{-+}] + [(D\sigma G)_{+-}] + [(D\sigma G)_{++}].$$

On coarse/fine interfaces we again sum over the cells where the basis function for an interface node has support:

$$[(D\sigma G)]_o = \frac{1}{w_o} \left[\sum_{c \in S_c(o)} [(D\sigma G)_c]_o + \frac{1}{r^d} \left(\sum_{f \in S_f(o)} [(D\sigma G)_f]_o + \sum_{p \in F(o)} \chi(p) \sum_{f \in S_f(p)} [(D\sigma G)_f]_p \right) \right].$$

Note that the distinction between $\sigma^{(x)}$ and $\sigma^{(y)}$ is only relevant on coarser levels of a multigrid structure, as these directional coefficients were introduced as part of the multigrid coarsening scheme. In other cases, including all applications of the interface stencils, we have $\sigma^{(x)} = \sigma^{(y)} = \sigma = 1/\rho$.

The five-point formula obtained from linear basis functions over triangles differs from the nine-point formula only in having a different unit cell:

$$[(D\sigma G)_{+-}] = \frac{\sigma^{(x)}_{+-}}{2\Delta x^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \frac{\sigma^{(y)}_{+-}}{2\Delta y^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

The extension of this formula to 3-d to obtain the seven-point stencil is straightforward.

Full-weighting restriction in a uniform part of the mesh assigns to each coarse node a weighted sum of the values at all fine nodes within the 4 surrounding coarse cells (8 in 3-d). For each coarse node at the coarse/fine interface there is a smaller set of nearby fine nodes, with the remainder of the coarse value coming from the interface node itself. If we define $G(o)$ as the set of fine nodes—not interface nodes—not more than r fine cells away from o in any direction, we can express the restriction of a quantity s defined on level ℓ as

$$(I_\ell^{\ell-1} s)_o = w_o s_o + \frac{1}{r^d}\sum_{p\in G(o)} \chi(p) s_p.$$

Note that letting s be a constant provides an alternate definition of w_o.

For axisymmetric problems in 2-d we scale the difference stencils by r (which here denotes radius rather than refinement ratio) to put them in conservative form. Correction terms are required to correctly account for the r-dependence in the finite element integrals. Here we present only the form of the stencils associated with bilinear elements.

Divergence becomes

$$[rD] = \left(\frac{1}{2\Delta r}\begin{bmatrix} -r_- & r_+ \\ -r_- & r_+ \end{bmatrix}, \frac{1}{2\Delta z}\begin{bmatrix} r_- & r_+ \\ -r_- & -r_+ \end{bmatrix} + \frac{\Delta r}{12\Delta z}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\right),$$

where r_- and r_+ are defined at cell centers. Note that except for the correction term this stencil is the same as would be obtained by replacing the vector field V by rV. The stencil breaks up into individual cells for assembly at the coarse/fine interface just as the Cartesian grid stencils do.

Gradient also requires a correction term.

$$[\,\overline{G}\,] = \left(\frac{1}{2\Delta r}\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}, \frac{1}{2\Delta z}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} + \frac{\Delta r}{12 r\Delta z}\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}\right)^{\mathrm{T}},$$

where $[\,\overline{G}\,]$ and r are both defined at a cell center.

For the linear operator L_ρ we redefine σ as r/ρ and use the same multigrid coarsening formulae as before. The stencil then looks like the Cartesian grid stencil with a small correction term, the contribution from the lower right cell now being

$$[(rD(1/\rho)G)_{+-}] = \frac{\sigma^{(r)}_{+-}}{6\Delta r^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & 2 \\ 0 & -1 & 1 \end{bmatrix} + \frac{\sigma^{(z)}_{+-}}{6\Delta z^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & -1 \\ 0 & 2 & 1 \end{bmatrix} + \frac{\sigma^{(z)}_{+-}\Delta r}{12 r_+ \Delta z^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix}.$$

The other unit cells have a similar correction, except for the signs. In the upper right cell, like this one, the correction opposes the main difference term, while in both left cells the correction has the same sign as the main difference. In all four cases the effect is to shift the effective r for the cell towards the node where the stencil is being evaluated.

REFERENCES

[1] A. S. Almgren, J. B. Bell, P. Colella, and L. H. Howell. An adaptive projection method for the incompressible Euler equations. In *11th AIAA Computational Fluid Dynamics Conference*, Orlando, FL, July 6–9 1993.

[2] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. Welcome. A high-resolution adaptive projection method for regional atmospheric modeling. In *Proceedings of the U.S. EPA NGEMCOM Conference*, August 1995.

[3] A. S. Almgren, J. B. Bell, L. H. Howell, and P. Colella. An adaptive projection method for the incompressible Navier-Stokes equations. In *Proceedings of the IMACS 14th World Conference*, Atlanta, GA, July 11–15, 1994.

[4] 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), March 1996.

[5] J. B. Bell, M. J. Berger, J. S. Saltzman, and M. Welcome. Three dimensional adaptive mesh refinement for hyperbolic conservation laws. Technical Report UCRL-JC-108794, LLNL, Dec. 1991.

[6] 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, December 1989.

[7] J. B. Bell, P. Colella, and L. H. Howell. An efficient second-order projection method for viscous incompressible flow. In *10th AIAA Computational Fluid Dynamics Conference*, Honolulu, June 24–27, 1991.

[8] J. B. Bell and D. L. Marcus. A second-order projection method for variable-density flows. *J. Comput. Phys.*, 101:334–348, 1992.

[9] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82:64–84, 1989.

[10] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.

[11] M. J. Berger and I. Rigoustsos. An algorithm for point clustering and grid generation. Technical Report NYU-501, New York University-CIMS, 1991.

[12] G. L. Brown and A. Roshko. On density effects and large structure in turbulent mixing layers. *J. Fluid Mech.*, 64(4):775–816, 1974.

[13] K.-Y. Chien, R. E. Ferguson, A. L. Kuhl, H. M. Glaz, and P. Colella. Inviscid dynamics of two-dimensional shear layers. *Comp. Fluid Dyn.*, 5:59–80, 1995.

[14] Terry L. Clark and R. D. Farley. Severe downslope windstorm calculations in two and three spatial dimensions using anelastic interactive grid nesting: A possible mechanism for gustiness. *J. Atmos. Sci.*, 41(3):329–350, 1984.

[15] P. Colella. A direct Eulerian MUSCL scheme for gas dynamics. *SIAM Journal on Computing*, 6:104–117, January 1985.

[16] P. Colella. A multidimensional second order Godunov scheme for conservation laws. *J. Comput. Phys.*, 87:171–200, 1990.

[17] L. H. Howell and J. B. Bell. An adaptive-mesh projection method for viscous incompressible flow. *SIAM J. Sci. Comput.*, to appear.

[18] J. H. Konrad. *An Experimental Investigation of Mixing in Two-Dimensional Turbulent Shear Flows with Applications to Diffusion-Limited Chemical Reactions.* PhD thesis, California Institute of Technology, 1977.

[19] S. F. McCormick. *Multilevel Adaptive Methods for Partial Differential Equations.* SIAM, Philadelphia, 1989.

[20] M. L. Minion. On the stability of Godunov-projection methods for incompressible flow. *J. Comput. Phys.*, accepted for publication,1996.

[21] M. L. Minion. A projection method for locally refined grids. *J. Comput. Phys.*, accepted for publication,1996.

[22] P. A. Monkewitz and P. Huerre. Influence of the velocity ratio on the spatial instability of mixing layers. *J. Physics of Fluids*, 25:1137–1143, 1982.

[23] Eldridge G. Puckett, Ann S. Almgren, John B. Bell, Daniel L. Marcus, and William G. Rider. A second-order projection method for tracking fluid interfaces in variable density incompressible flows. *J. Comput. Phys.*, 130:269–282, 1997.

[24] W. Sandberg R. Ramamurti and R. Löhner. Simulation of torpedo launch using a 3-D incompressible finite element solver and adaptive remeshing. Technical Report 95-0086, AIAA, 1995.

[25] R. Ramamurti, R. Löhner, and W. Sandberg. Evaluation of a Three-Dimensonal finite element incompressible flow solver. Technical Report 94-0756, AIAA, 1994.

[26] William C. Skamarock and Joseph B. Klemp. Adaptive grid refinement for two-dimensional and three-dimensional nonhydrostatic atmospheric flow. *Monthly Weather Review*, 121:788–804, March 1993.

[27] E. Steinthorsson, D. Modiano, W. Y. Crutchfield, J. B. Bell, and P. Colella. An adaptive semi-implicit scheme for simulations of unsteady viscous compressible flow. In *12th AIAA Computational Fluid Dynamics Conference*, Orlando, FL, June 1995.

[28] David E. Stevens. *An Adaptive Multilevel Method for Boundary Layer Meteorology.* PhD thesis, Dept. of Applied Mathematics, University of Washington, 1994.

[29] David E. Stevens and Christopher S. Bretherton. A forward-in-time advection scheme and adaptive multilevel flow solver for nearly incompressible atmospheric flow. *J. Comput. Phys.*, 129:284–295, 1996.

[30] P. Wesseling. *An Introduction to Multigrid Methods.* Wiley, New York, 1992.