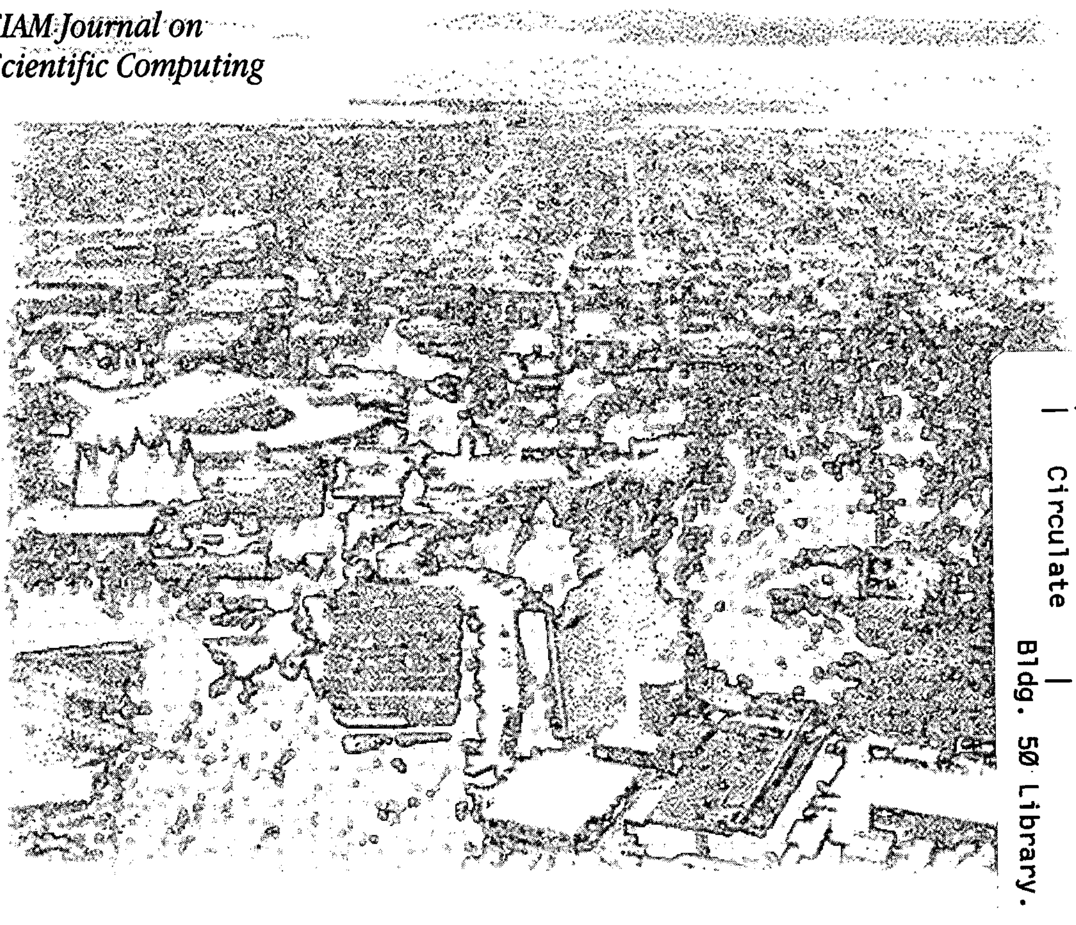
LBNL-39075
UC-405
Preprint

ERNEST ORLANDO LAWRENCE
BERKELEY NATIONAL LABORATORY

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

A.S. Almgren, J.B. Bell, P. Colella, L.H. Howell,
and M.L. Welcome
Computing Sciences Directorate

July 1996
To be submitted to
*SIAM Journal on
Scientific Computing*



LBNL-39075
UC-405

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

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

Computing Sciences Directorate
Ernest Orlando Lawrence Berkeley National Laboratory
University of California
Berkeley, California 94720

July 1996

This work was supported by the Applied Mathematical Sciences Program and the HPCC Grand Challenge Program of the U.S. Department of Energy Office of Mathematics, Information, and Computational Sciences, and by the Defense Nuclear Agency under IACRO 95-2045, under Contract No. DE-AC03-76SF00098.

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 grids 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 accuracy and convergence properties 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.

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

[*]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.

equations governing this flow are:

$$U_t + (U \cdot \nabla)U = \frac{1}{\rho}(-\nabla p + \mu\nabla^2 U + H_U), \tag{1.1}$$

$$\rho_t + \nabla \cdot (\rho U) = 0, \tag{1.2}$$

$$c_t + (U \cdot \nabla)c = H_c + k\nabla^2 c, \tag{1.3}$$

$$\nabla \cdot U = 0 \quad , \tag{1.4}$$

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. [22]. The basic methodology presented in those papers was motivated by a desire to apply higher-order upwind methods developed for gas dynamics to incompressible flow. In particular, they use a specialized version of the unsplit second-order upwind methodology for the convective terms in (1.1)-(1.3) that was introduced for gas dynamics by Colella [15]. 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.

This approach 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. [25] generalized this approach to the compressible Navier-Stokes equations in two dimensions. For incompressible flow, Howell and Bell [16] 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 [20] 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 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 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.

Stevens [26, 27] presents an adaptive projection algorithm with subcycling in time for the anelastic equations describing the atmosphere. (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.) Stevens' algorithm uses a staggered representation of velocities with arbitrary integer factors of refinement combined with a different type of difference approximation than is presented here. Coarse grid face-based velocities are used as boundary conditions for the fine grid face-based velocities. This leads to Neumann boundary conditions for pressure on fine grids; consequently, the overall algorithm conserves mass. However, he uses no additional synchronization between levels, which means that at coarse/fine interfaces there is a mismatch in the pressure field although the normal derivatives of pressure match. One of the examples presented in Section 4 indicates that failure to properly match the pressure at coarse/fine boundaries can lead to a loss of accuracy of the overall method.

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 [24], Ramamurti, Sandberg and Löhner [23] 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. The method uses subcycling in time; however, unlike earlier versions of the adaptive algorithm, we now use an intermediate MAC projection so that the advection velocities used in evaluating the convective terms satisfy the divergence constraint (1.4). Special attention is given 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 remain constant independent of grid refinement patterns and the velocity field.

Before describing the adaptive algorithm we will review, in the next section, 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. Other aspects of the adaptive algorithm are also discussed. 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 (1.2)-(1.3) for the updated density and concentration, and we compute an intermediate velocity field from (1.1) 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 (1.1)-(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

$$\frac{U^* - U^n}{\Delta t} = -[\nabla \cdot (UU)]^{n+1/2} + \frac{1}{\rho^{n+1/2}}(-\nabla p^{n-1/2} + \frac{\mu}{2}(\nabla^2 U^n + \nabla^2 U^*) + H_U^{n+1/2}), \tag{2.1}$$

$$\frac{\rho^{n+1} - \rho^n}{\Delta t} = -[\nabla \cdot (\rho U)]^{n+1/2} \tag{2.2}$$

and

$$\frac{c^{n+1} - c^n}{\Delta t} = -[\nabla \cdot (cU)]^{n+1/2} + H_c^{n+1/2} + \frac{k}{2}(\nabla^2 c^n + \nabla^2 c^{n+1}) \tag{2.3}$$

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. The method uses an unsplit second-order upwind predictor-corrector scheme for evaluating the advective derivatives in (2.1)-(2.3) For this step the pressure gradient is evaluated at $t^{n-1/2}$ and is treated as a source term in (2.1), 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 (1.1). For face $(i + \frac{1}{2}, j, k)$ this gives

$$\tilde{u}^{L,n+1/2}_{i+1/2,j,k} \approx u^n_{i,j,k} + \frac{\Delta x}{2} u_x + \frac{\Delta t}{2} u_t$$

$$= u^n_{i,j,k} + (\frac{\Delta x}{2} - u^n_{i,j,k}\frac{\Delta t}{2})(u_x^{n,lim})_{i,j,k} + \frac{\Delta t}{2}(-(\widehat{vu_y})_{i,j,k} - (\widehat{wu_z})_{i,j,k} + \tag{2.4}$$
$$\frac{1}{\rho^n_{i,j,k}}(-(Gp)^{n-\frac{1}{2}}_{i,j,k} + \mu\Delta^{h,c}u^n_{i,j,k} + H^n_{x,i,j,k})).$$

extrapolated from (i, j, k), and

$$\tilde{u}^{R,n+1/2}_{i+1/2,j,k} \approx u^n_{i+1,j,k} - \frac{\Delta x}{2} u_x + \frac{\Delta t}{2} u_t$$

$$= u^n_{i+1,j,k} - (\frac{\Delta x}{2} + u^n_{i+1,j,k}\frac{\Delta t}{2})(u^{n,lim}_x)_{i+1,j,k} + \frac{\Delta t}{2}(-(\widehat{vu_y})_{i+1,j,k} - (\widehat{wu_z})_{i+1,j,k} + \tag{2.5}$$
$$\frac{1}{\rho^n_{i+1,j,k}}(-(Gp)^{n-\frac{1}{2}}_{i+1,j,k} + \mu\Delta^{h,c}u^n_{i+1,j,k} + H^n_{x,i+1,j,k}))$$

extrapolated from $(i + 1, j, k)$. Here, $\Delta^{h,c}$ is a standard, five-point in 2-d, seven-point in 3-d, cell-centered approximation to the Laplacian and G is a discretization of the gradient operator.

Analogous formulae are used to predict values for $\tilde{v}^{F/B,n+1/2}_{i,j+1/2,k}$ and $\tilde{w}^{U/D,n+1/2}_{i,j,k+1/2}$ at the other faces of the cell. In evaluating these terms the first derivatives normal to the face (in this case $u^{n,lim}_x$) are evaluated using a monotonicity-limited fourth-order slope approximation [14]. 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

$$\hat{U}^F_{i,j+1/2,k} = U^n_{i,j,k} + \left(\frac{\Delta x}{2} - \frac{\Delta t}{2}v^n_{i,j,k}\right)(U^{n,lim}_y)_{i,j,k} \quad , \tag{2.6}$$

$$\hat{U}^B_{i,j+1/2,k} = U^n_{i,j+1,k} - \left(\frac{\Delta x}{2} + \frac{\Delta t}{2}v^n_{i,j+1,k}\right)(U^{n,lim}_y)_{i,j+1,k} \quad . \tag{2.7}$$

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

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):

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

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

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

After constructing $\hat{U}_{i,j-\frac{1}{2},k}$, $\hat{U}_{i,j,k+1/2}$ and $\hat{U}_{i,j,k-1/2}$ in a similar manner, we use these upwind values to form the transverse derivatives in (2.4) and (2.5):

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

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

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)

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

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

$$D^{MAC}(\frac{1}{\rho^n}G^{MAC}\phi^{MAC}) = D^{MAC}(\tilde{U}^{n+1/2}) \tag{2.8}$$

is solved for ϕ^{MAC}, where

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

and $G^{MAC} = -(D^{MAC})^T$ so that

$$(G^{MAC}_x\phi^{MAC})_{i+1/2,j,k} = \frac{(\phi^{MAC}_{i+1,j,k} - \phi^{MAC}_{i,j,k})}{\Delta x}$$

with G^{MAC}_y and G^{MAC}_z defined analogously. The face-based advection velocities are then defined by

$$u^{ADV}_{i+\frac{1}{2},j,k} = \tilde{u}^{n+1/2}_{i+\frac{1}{2},j,k} - \frac{1}{\rho^n_{i+\frac{1}{2},j,k}}(G^{MAC}_x\phi^{MAC})_{i+1/2,j,k}$$

$$v^{ADV}_{i,j+\frac{1}{2},k} = \tilde{v}^{n+1/2}_{i,j+\frac{1}{2},k} - \frac{1}{\rho^n_{i,j+\frac{1}{2},k}}(G^{MAC}_y\phi^{MAC})_{i,j+1/2,k}$$

$$w^{ADV}_{i,j,k+\frac{1}{2}} = \tilde{w}^{n+1/2}_{i,j,k+\frac{1}{2}} - \frac{1}{\rho^n_{i,j,k+\frac{1}{2}}}(G^{MAC}_z\phi^{MAC})_{i,j,k+\frac{1}{2}}$$

where ρ on the faces is averaged geometrically from the cell centers at time n.

At this point the predictor step is performed for the tangential velocity components, density and concentration from cell centers to all cell faces. The extrapolation of the normal velocity components has been described above; the tracing of density, concentration and tangential velocity components is analogous, with the time derivatives replaced using (1.1)-(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:

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

The conservative update terms can now be defined by

$$[\nabla \cdot (SU)]^{n+1/2}_{i,j,k} = [\nabla \cdot (U^{ADV} \tilde{S}^{n+1/2})]_{i,j,k} = \frac{1}{\Delta x}(\tilde{S}_{i+\frac{1}{2},j,k} u^{ADV}_{i+\frac{1}{2},j,k} - \tilde{S}_{i-\frac{1}{2},j,k} u^{ADV}_{i-\frac{1}{2},j,k}) +$$

$$\frac{1}{\Delta y}(\tilde{S}_{i,j+\frac{1}{2},k} v^{ADV}_{i,j+\frac{1}{2},k} - \tilde{S}_{i,j-\frac{1}{2},k} v^{ADV}_{i,j-\frac{1}{2},k}) + \frac{1}{\Delta z}(\tilde{S}_{i,j,k+\frac{1}{2}} w^{ADV}_{i,j,k+\frac{1}{2}} - \tilde{S}_{i,j,k-\frac{1}{2}} w^{ADV}_{i,j,k-\frac{1}{2}})$$

Using this approximation we now compute ρ^{n+1} from (2.2). Equations (2.1) and (2.3) require solution of parabolic equations for each component of the intermediate velocity U^* and for the concentration c^{n+1}. For these solves the conservative updates and forcing terms are treated as explicit source terms. The parabolic solves are described in more detail in Section 3.

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:

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

We note here that since the viscous terms are not included in defining the states used in the transverse derivatives ((2.6)-(2.7)) there is an additional stability constraint on the time step for large μ (see [19]). If the viscous terms were included in (2.6)-(2.7) the time step control specified above would be sufficient.

The velocity field U^* computed using (2.1) 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 $(U^* - U^n)/\Delta t$ to obtain the new velocity field, U^{n+1}, and an update for the pressure. In particular, if $\mathbf{P}$ represents the projection then

$$\frac{U^{n+1} - U^n}{\Delta t} = \mathbf{P}\left(\frac{U^* - U^n}{\Delta t}\right) \tag{2.9}$$

$$\frac{1}{\rho^{n+1/2}} \nabla p^{n+1/2} = \frac{1}{\rho^{n+1/2}} \nabla p^{n-\frac{1}{2}} + (\mathbf{I} - \mathbf{P})\left(\frac{U^* - U^n}{\Delta t}\right).$$

Note that the vector field 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 $(U^* - U^n)/\Delta t$ which we denote by $(1/\rho)G\phi$. We determine ϕ by solving

$$L_\rho^{n+1/2}\phi = D\left(\frac{U^* - U^n}{\Delta t}\right)$$

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

$$\int \frac{1}{\rho}\nabla\phi(\mathbf{x}) \cdot \nabla\psi(\mathbf{x})\, d\mathbf{x} = \int \frac{U^* - U^n}{\Delta t} \cdot \nabla\psi(\mathbf{x})\, d\mathbf{x}, \qquad \forall\psi(\mathbf{x}) \tag{2.10}$$

where $d\mathbf{x}$ is the volume element $dx\ dy$, $r\ dr\ d\theta$ or $dx\ dy\ dz$ 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

$$\frac{U^{n+1} - U^n}{\Delta t} = \frac{U^* - U^n}{\Delta t} - \frac{1}{\rho^{n+1/2}}\overline{G\phi}, \tag{2.11}$$

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 (2.9) and (2.10) 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 resulting in a twenty-seven point stencil; 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 (2.1) 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 (2.1) 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 (2.1) 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. Then we present a brief description of the time step algorithm for the grid system that subcycles in time, and describe the initialization procedure used to begin a multilevel calculation. In the fourth and fifth subsections we discuss the spatial discretization of the single-level and multilevel elliptic operators used in the algorithm. Finally, we discuss the details of the adaptive time step procedure, focusing on the synchronization between different levels of refinement.

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 error is above a given tolerance. 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 (k_ℓ is 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 Overview of Time-Stepping Procedure

The adaptive projection 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 $\ell + 1 < \ell_{max}$.

To advance the data at a single level requires evaluation of advective terms which are of hyperbolic character, solution of an elliptic equation (the MAC solve) to make the face-based advection velocities (U^{ADV}) divergence-free, solution of parabolic equations to account for the diffusive terms in the momentum and advected quantity equations, and solution of an elliptic equation to enforce the divergence constraint on the cell-centered new-time velocity (U^{n+1}) and define the update to pressure. The character of the equation determines the type of operation it requires in an adaptive framework.

As in the adaptive mesh technique for hyperbolic systems, the evaluation of the conservative derivatives in (2.1)-(2.3) can be performed one grid at a time, with boundary data copied from other fine grids, interpolated from underlying coarse grids, or supplied from physical boundary conditions. The parabolic and single-level elliptic solves require that the solution be computed on all grids at a level at one time, since these are no longer explicit operations. Boundary data for these solves are interpolated from underlying coarse grids or supplied from physical boundary conditions. The interpolation and solution procedure for these equations are discussed in Sections 3.4 and 3.5.

Once the level $\ell + 1$ data have been advanced to the same point in time as the level ℓ data, synchronization of the data between levels is required. For hyperbolic systems, averaging the fine data onto the coarse grids and refluxing across the coarse/fine interface comprise the synchronization step. For incompressible flows, we also average the data onto coarser levels and reflux across the coarse/fine interface; however, we must also account for the mismatch in the MAC-projected advection velocity (U^{ADV}) and the mismatch in the projected new-time velocity (U^{n+1}) which result from not having satisfied the full elliptic matching conditions at the coarse/fine interface. Namely, when performing the MAC and nodal projections we impose Dirichlet boundary conditions from level ℓ for the level $\ell + 1$ grids. Consequently, the fields computed in the elliptic solves match in value at coarse/fine interfaces but do not, in general, match normal derivatives.

In other words, in addition to the loss of conservation, which is corrected by refluxing, there is error in the multilevel solution due to each of the projections (MAC and nodal) which has occurred during the time step. The aim of the synchronization step is to correct each of those as much as possible. The correction is such that if the algorithm were run without subcycling in time, the synchronization step would be an exact fix to the Neumann mismatch resulting from the single-level elliptic solves. However, because the problem is nonlinear and multiple fine grid time steps are taken for each coarse grid time step, the fix cannot be exact.

The error resulting from the mismatch in MAC-projected velocities at the coarse/fine interface is such that the effective time-averaged advection velocity used on the coarse and fine grids does not satisfy what would be the composite MAC-divergence constraint at the coarse/fine interface. Similarly, the Neumann mismatch in the nodal single-level projections causes the new-time, composite velocity field (as represented on the fine grid where possible and on the coarse grid elsewhere) to not satisfy the composite nodal divergence constraint on the coarse/fine interface. As stated above, both of these result from having used Dirichlet boundary conditions interpolated from the coarse grid for the fine grid solves; this matches pressure at the coarse/fine boundary but does not match the normal derivative of pressure. Discretely, this inconsistency is manifest as a composite residual at the interface found using a multilevel stencil which sees both the coarse and fine data adjacent to the interface.

In order to correct for the mismatches, two additional elliptic solves are performed at the end of a level ℓ time step. The first is a level ℓ MAC sync solve 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 second is a composite (two-level) nodal 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 accumulated will be discussed in Section 3.6. The synchronization is further complicated because the corrections to the velocity field resulting from refluxing and from the MAC synchronization correction must also be projected to split them into divergence-free and gradient components.

Details of the spatial discretization and solution procedures for the level projection operators (MAC and nodal), the diffusion operators and synchronization projections are included in the next two subsections. Following that discussion we will give the exact details of the multilevel time-stepping procedure.

3.3 Initialization of the Multilevel Data

As in the single grid projection method, we must first project the given velocity field to 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 3.4. As a result, the velocity resulting from this projection satisfies the divergence constraint 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, $0 \leq \ell < \ell_{max}$, 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 $0 \leq \ell \leq \ell_{max}$. One multilevel composite solve 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 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.

3.4 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 projections on all levels at once. We compute these solutions using a multigrid algorithm adapted to the AMR grid hierarchy. The stencils at coarse/fine interfaces are determined by the finite-element derivation of the projection operator, in contrast to some other multilevel methods, e.g. FAC [18], which derive the relationships between coarse and fine grid data from the multigrid algorithm itself. Additional complications come from the need to support refinement ratios of 2 or 4 between levels.

Stencils at the coarse/fine interface are defined by equation (2.10), just as in the grid interiors. Figure 1 shows the spatial extent of the stencils for the 2-d 9-point discretization for a refinement ratio of 4; the 2-d 5-point and 3-d 7-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.

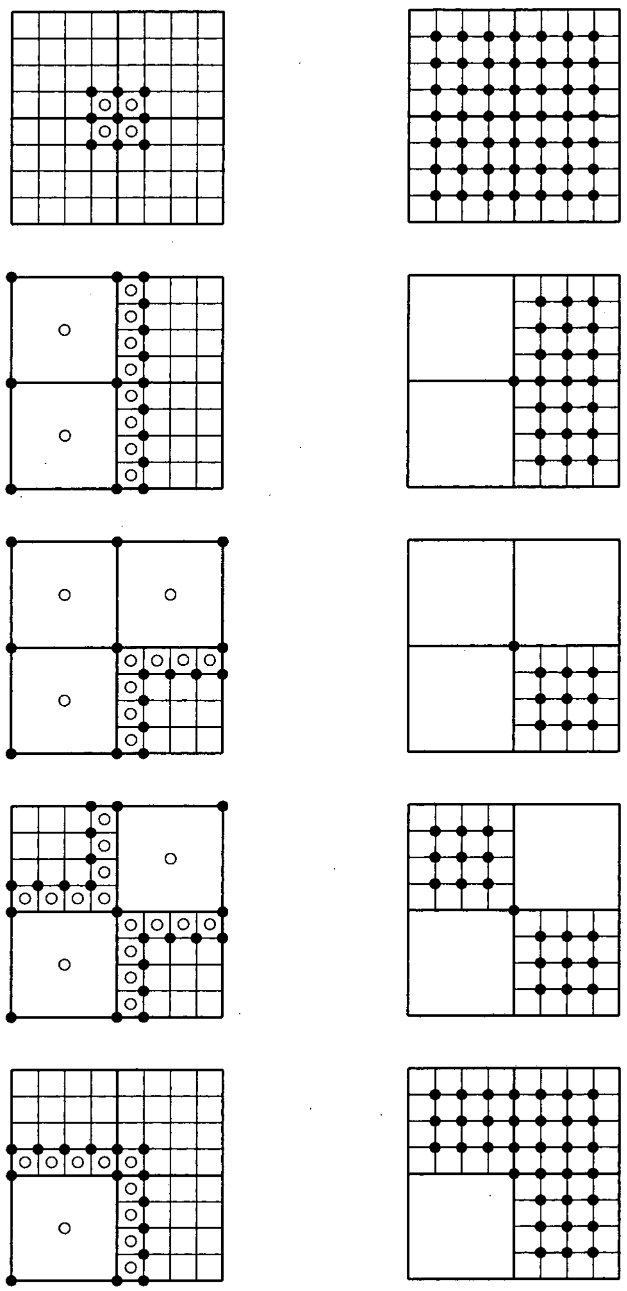


Figure 1: Stencils at grid edges and corners, shown for a refinement ratio of four. On the left, the stencil for $\nabla \cdot \frac{1}{\rho} \nabla \phi$ uses ϕ values defined at nodes (solid circles) and ρ values defined at cells (open circles). Also, the divergence stencil for $\nabla \cdot V$ uses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

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 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 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 AMR 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. [28]) 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 3 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 $L^m e^m = r^m$, where m is either identical to or coarsened from an AMR level ℓ, has the following recursive form:

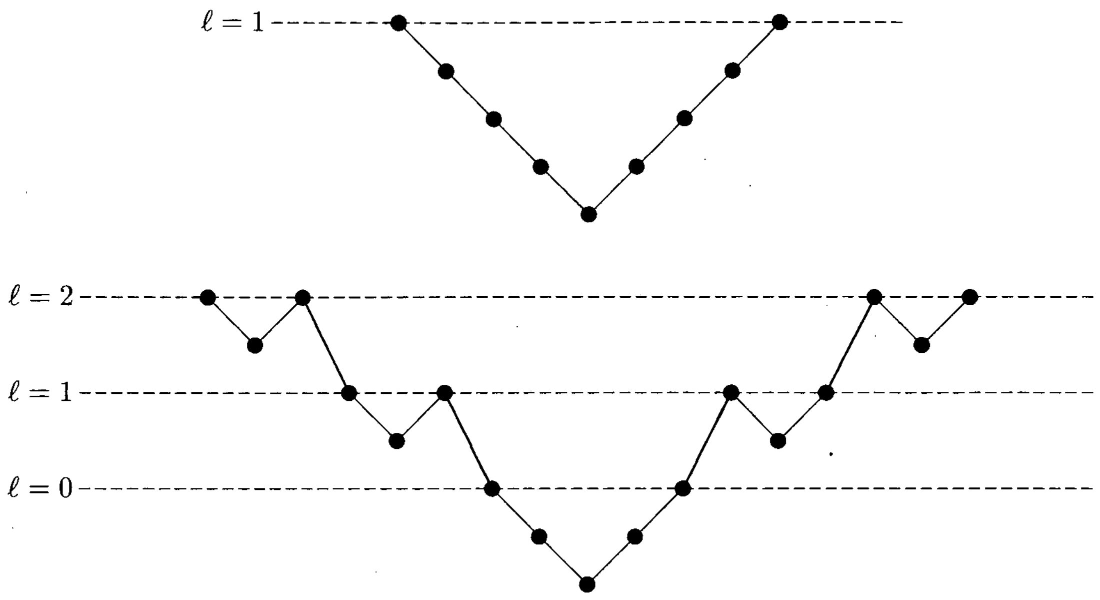


Figure 2: In the multigrid V-cycle (top), operations apply only to interior points of a level. In the multilevel cycle (bottom) two operations are defined that cross the coarse/fine interface—computing the residual, and restricting it to the coarse grid. Dotted lines show AMR levels, other levels are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

```
**Begin** V-cycle($L^m$, $e^m$, $r^m$, $m$, $\ell$, $\nu_1$, $\nu_2$):
    **If** ($m = 0$) **then**
        Solve($L^m$, $e^m$, $r^m$)
    **Else if** ($\ell - 1 \geq \ell_{lo}$ and $m - 1 = m(\ell - 1)$)
        Relax($L^m$, $e^m$, $r^m$, $\nu_2$)
    **Else**
        Relax($L^m$, $e^m$, $r^m$, $\nu_1$)
        $r^{m-1} := I_m^{m-1}(r^m - L^m e^m)$
        $e^{m-1} := 0$
        V-cycle($L^{m-1}$, $e^{m-1}$, $r^{m-1}$, $m - 1$, $\ell$, $\nu_1$, $\nu_2$)
        $e^m := e^m + I_{m-1}^m e^{m-1}$
        Relax($L^m$, $e^m$, $r^m$, $\nu_2$)
    **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.

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

$$[I_m^{m-1}] = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}. \tag{3.1}$$

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 formulas 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

$$\sigma_{i/2,j/2}^{(x),m-1} = \frac{1}{\dfrac{1}{\sigma_{i,j}^m + \sigma_{i,j+1}^m} + \dfrac{1}{\sigma_{i+1,j}^m + \sigma_{i+1,j+1}^m}}, \tag{3.2}$$

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 formulas work with both the 5-point and 9-point linear operators in 2-d, and an obvious extension applies to the 7-point operator in 3-d. (For simplicity, we present the formulas 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,

$$e^{m+1}_{2i-1/2,2j-1/2} = e^m_{i-1/2,j-1/2}, \tag{3.3}$$

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

$$e^{m+1}_{2i+1/2,2j-1/2} = \frac{(\sigma^{(x),m+1}_{2i,2j-1} + \sigma^{(x),m+1}_{2i,2j}) e^{m+1}_{2i-1/2,2j-1/2} + (\sigma^{(x),m+1}_{2i+1,2j-1} + \sigma^{(x),m+1}_{2i+1,2j}) e^{m+1}_{2i+1+1/2,2j-1/2}}{\sigma^{(x),m+1}_{2i,2j} + \sigma^{(x),m+1}_{2i,2j+1} + \sigma^{(x),m+1}_{2i+1,2j} + \sigma^{(x),m+1}_{2i+1,2j+1}}, \tag{3.4}$$

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

$$e^{m+1}_{2i+1/2,2j+1/2} = \tag{3.5}$$
$$\frac{\{(\sigma^{(x),m+1}_{2i,2j} + \sigma^{(x),m+1}_{2i,2j+1}) e^{m+1}_{2i-1/2,2j+1/2} + (\sigma^{(x),m+1}_{2i+1,2j} + \sigma^{(x),m+1}_{2i+1,2j+1}) e^{m+1}_{2i+1+1/2,2j+1/2} + (\sigma^{(y),m+1}_{2i,2j} + \sigma^{(y),m+1}_{2i+1,2j}) e^{m+1}_{2i+1/2,2j-1/2} + (\sigma^{(y),m+1}_{2i,2j+1} + \sigma^{(y),m+1}_{2i+1,2j+1}) e^{m+1}_{2i+1/2,2j+1+1/2}\}}{\sigma^{(x),m+1}_{2i,2j} + \sigma^{(x),m+1}_{2i,2j+1} + \sigma^{(x),m+1}_{2i+1,2j} + \sigma^{(x),m+1}_{2i+1,2j+1} + \sigma^{(y),m+1}_{2i,2j} + \sigma^{(y),m+1}_{2i+1,2j} + \sigma^{(y),m+1}_{2i,2j+1} + \sigma^{(y),m+1}_{2i+1,2j+1}}.$$

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 $L^\ell \phi^\ell = s^\ell$ is as follows:

```
**Begin** Multilevel cycle($L^\ell$, $\phi^\ell$, $s^\ell$, $r^{m(\ell+1)}$, $\ell$, $\nu_1$, $\nu_2$):
    $m := m(\ell)$
    $r^m := (s^\ell - L^\ell \phi^\ell)$
    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($L^\ell$, $e^m$, $r^m$, $m$, $\ell$, $\nu$, $\nu_2$)
    $\phi^\ell := \phi^\ell + e^m$
    If ($\ell > \ell^{min}$)
        $r^m := \begin{cases} (r^m - L^\ell e^m) & \text{on } \Omega^\ell \\ s^\ell - L^{\ell-1,\ell} \phi^{\ell-1,\ell} & \text{on } \partial\Omega^\ell - \partial\Omega \end{cases}$
        $\phi^{\ell-1}_{old} := \phi^{\ell-1}$
        Multilevel cycle($L^{\ell-1}$, $\phi^{\ell-1}$, $s^{\ell-1}$, $r^m$, $\ell-1$, $\nu_1$, $\nu_2$)
        $e^m := I^{\ell}_{\ell-1}(\phi^{\ell-1} - \phi^{\ell-1}_{old})$ on $\Omega^\ell + \partial\Omega^\ell$
        $\phi^\ell := \phi^\ell + e^m$ on $\Omega^\ell + \partial\Omega^\ell$
        $r^m := (r^m - L^\ell e^m)$
        $e^m := 0$
        V-cycle($L^\ell$, $e^m$, $r^m$, $m$, $\ell$, 0, $\nu_2$)
        $\phi^\ell := \phi^\ell + e^m$
    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.

3.5 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; the construction of the right-hand-side for the MAC synchronization step is described in greater detail in the next subsection. 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

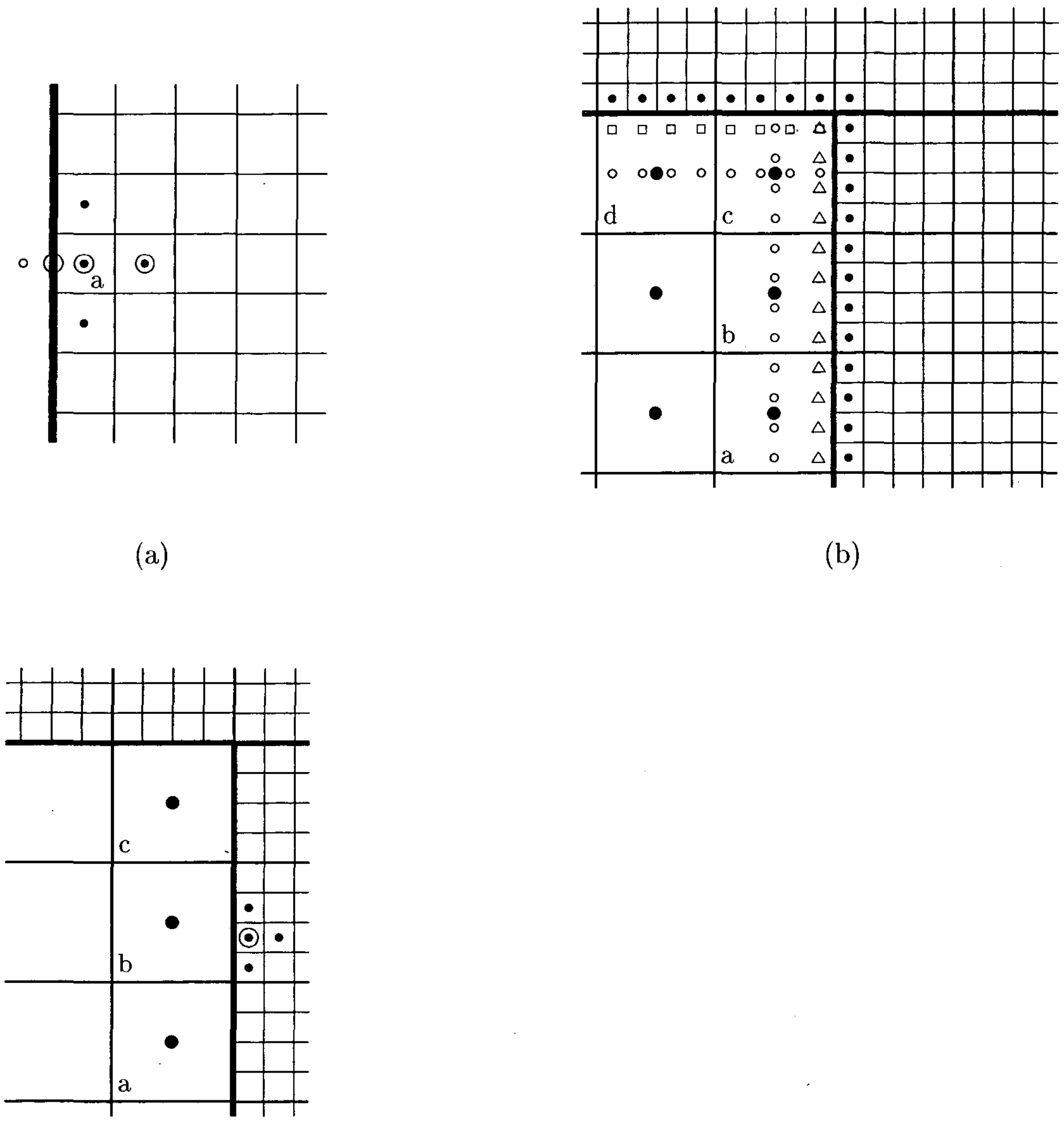


Figure 3: (a) At a physical boundary, interior and boundary values (○'s) are used to extrapolate to the ghost cell (∘); the ghost value and the other interior values (•'s) are used to construct the Laplacian at (a). (b) Locations of coarse grid boundary conditions (●), tangentially interpolated values (∘), fine grid cells (•), and ghost cells (△'s and □'s). (c) Domain of dependence (•'s and ●'s) of the Laplacian at a fine cell (○) adjacent to the coarse/fine interface.

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 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.

3.6 Details of Time-Stepping Procedure

In order to construct an algorithm which is both conservative and free-stream-preserving, we must store several additional quantities at coarse/fine interfaces. We refer to the face-based data structures that contain these quantities as registers. These include a velocity register (δU), advective and viscous flux registers for velocity ($\delta F_{U,adv}$ and $\delta F_{U,visc}$) and advective and diffusive flux registers for scalars ($\delta F_{S,adv}$ and $\delta F_{S,diff}$), where we now define $S = \{\rho, c\}$. The velocity register at level ℓ (δU^ℓ) contains the difference between the MAC-projected advection velocity at level ℓ ($U^{ADV,\ell}$) 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 $\ell + 1$. The advective flux registers at level ℓ ($\delta F^\ell_{U,adv}$ and $\delta F^\ell_{S,adv}$) contain the 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 $\ell+1$. The viscous/diffusive flux registers are defined analogously, but with the viscous/diffusive fluxes rather than advective fluxes. For convenience, these are weighted by the area of the level ℓ face and the level ℓ time step.

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.

3.6.1 Advancing a single level

Assume now that we are advancing level ℓ, $0 \le \ell \le \ell_{max}$, 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 (assumed the same in each coordinate direction) 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.

To advance the data on level ℓ itself, 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.

The extrapolation of normal velocities to faces and construction of the right-hand-side for the MAC projection are done one grid at a time, since these are completely explicit operations. The MAC projection, by contrast, requires solution of the elliptic equation for $\phi^{MAC,\ell}$ on all grids at level ℓ simultaneously, as described in the previous subsection. The boundary conditions for $\phi^{MAC,\ell}$ are homogeneous Neumann on all physical boundaries except for outflow, where $\phi^{MAC,\ell} = 0$. If $\ell > 0$, the values of $\phi^{MAC,\ell-1}$, considered constant in time over the $\ell - 1$ time step, supply the remaining boundary conditions. In order to keep the correct scaling of $\phi^{MAC,\ell}$ relative to $\phi^{MAC,\ell-1}$, we solve a re-scaled version of (2.8):

$$D^{MAC}\left(\frac{1}{\rho^{n,\ell}} G^{MAC}\left(\frac{\Delta t^\ell}{2} \phi^{MAC,\ell}\right)\right) = D^{MAC}(\tilde{U}^{n+1/2,\ell}).$$

We then define the advection velocity on cell faces in each level ℓ grid

$$U^{ADV,\ell} = \tilde{U}^{n+1/2,\ell} - \frac{1}{\rho^{n,\ell}} G^{MAC}\left(\frac{\Delta t^\ell}{2} \phi^{MAC,\ell}\right).$$

Now we complete the predictor step by tracing all components of velocity and scalars to faces, and upwinding using the advection velocity. The creation of the advective update terms for velocity ($[\nabla \cdot (U^{ADV,\ell}\, \tilde{U}^{n+1/2,\ell})]_{ijk}$) and scalars ($[\nabla \cdot (U^{ADV,\ell}\, \tilde{S}^{n+1/2,\ell})]_{ijk}$) is again composed of grid-by-grid operations.

If $\ell < \ell_{max}$, we initialize the level ℓ velocity and advective flux registers (defined only at the faces on the ℓ / $\ell+1$ interface and indexed by level ℓ indices) by

$$\delta U^{\ell} := -A^{\ell} U^{ADV,\ell}$$

$$\delta F^{\ell}_{U,adv} := -\Delta t^{\ell} A^{\ell} (U^{ADV,\ell} \tilde{U}^{n+1/2,\ell})$$

$$\delta F^{\ell}_{S,adv} := -\Delta t^{\ell} A^{\ell} (U^{ADV,\ell} \tilde{S}^{n+1/2,\ell}).$$

If $\ell > 0$, we then update the level $\ell - 1$ velocity and advective flux registers (defined only at the faces on the $\ell - 1$ / ℓ interface and indexed by level $\ell - 1$ indices) by

$$\delta U^{\ell-1} := \delta U^{\ell-1} + \frac{1}{r} A^{\ell-1} \sum_{faces} U^{ADV,\ell}$$

$$\delta F^{\ell-1}_{U,adv} := \delta F^{\ell-1}_{U,adv} + A^{\ell-1} \sum_{faces} \Delta t^{\ell} (U^{ADV,\ell} \tilde{U}^{n+1/2,\ell})$$

$$\delta F^{\ell-1}_{S,adv} := \delta F^{\ell-1}_{S,adv} + A^{\ell-1} \sum_{faces} \Delta t^{\ell} (U^{ADV,\ell} \tilde{S}^{n+1/2,\ell}).$$

Note that one level $\ell - 1$ face contains r^2 (in two dimensions it would be r) level ℓ faces; the sums above should be interpreted as summing over all level ℓ faces which are contained in the level $\ell - 1$ face. Note that the $\frac{1}{r}$ weighting is necessary for the velocity register but not for the flux registers because the flux registers incorporate Δt^{ℓ} directly.

Having completed the predictor, which defines the advective update terms for both U and S, we must now do the viscous solve for U and the diffusive solve for c. These have the form of solving for $U^{*,\ell}$ in (a rewritten version of (2.1))

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

and for $c^{n+1,\ell}$ in (a rewritten version of (2.3))

$$(1 - \frac{k \Delta t^{\ell}}{2} \nabla^2) c^{n+1,\ell} = c^{n,\ell} - \Delta t^{\ell} [\nabla \cdot (U^{ADV,\ell} \tilde{c}^{n+1/2,\ell})] + \Delta t^{\ell} H_c^{n+1/2,\ell} + \frac{k \Delta t^{\ell}}{2} \nabla^2 c^{n,\ell}.$$

Boundary conditions for these solves are supplied from the physical boundaries where appropriate. Where coarse grid data are needed for boundary conditions they are linearly interpolated in time.

If $\ell < \ell_{max}$, we now initialize the level ℓ viscous flux registers (defined only at the level ℓ / $\ell + 1$ interface) as below. The quantity put into the viscous flux register is constructed by multiplying the normal gradient across the face of the time-averaged quantity by the area of that face, the time step, and the diffusive coefficient.

$$\delta F^{\ell}_{U,visc} := -\mu \Delta t^{\ell} A^{\ell} G^{MAC}_{norm} (\frac{U^{n,\ell} + U^{*,\ell}}{2})$$

$$\delta F^{\ell}_{c,diff} := -k\Delta t^{\ell} A^{\ell} G^{MAC}_{norm}(\frac{c^{n,\ell}+c^{n+1,\ell}}{2})$$

If $\ell > 0$, we then update the level $\ell-1$ flux viscous registers by

$$\delta F^{\ell-1}_{U,visc} := \delta F^{\ell-1}_{U,visc} + A^{\ell-1} \sum_{faces} \mu \Delta t^{\ell} G^{MAC}_{norm}(\frac{U^{n,\ell}+U^{*,\ell}}{2})$$

$$\delta F^{\ell-1}_{c,diff} := \delta F^{\ell-1}_{c,diff} + A^{\ell-1} \sum_{faces} k \Delta t^{\ell} G^{MAC}_{norm}(\frac{c^{n,\ell}+c^{n+1,\ell}}{2}).$$

Again the summing convention is that contributions are summed from all level ℓ faces contained in the level $\ell-1$ face.

Finally, to complete the level ℓ time step, we perform the level ℓ nodal level projection, which requires solving

$$L^{n+1/2}_{\rho} \phi^{\ell} = D(V^{\ell})$$

on level ℓ for ϕ^{ℓ}, where $V^{\ell} = \frac{U^{*,\ell}-U^{n,\ell}}{\Delta t^{\ell}}$. Boundary conditions from $\phi^{\ell-1}$ are considered piecewise constant in time over the level $\ell-1$ time step.

We then define $U^{n+1,\ell}$ and $p^{n+1/2,\ell}$ by

$$U^{n+1,\ell} = U^{*,\ell} - \frac{1}{\rho^{n+1/2,\ell}}(\overline{G\phi})^{\ell}$$

and

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

Residuals from the level projections must be accumulated to define the right-hand-side for the sync projection. Because the projection is not exact, if we defined the right-hand-side for the sync projection by taking 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 in each level projection the right-hand-side would not be zero. Hence, in order to capture only the mismatch at the coarse/fine interface and not the "approximateness" of the projection, we define the right-hand-side for the sync projection as the time-averaged composite residual. This contains a measure of the extent to which the level projections fail to satisfy the equations which define the composite projection at the coarse/fine interface, but not the extent to which the projection is non-exact. Therefore, at the end of each level ℓ time step, if $\ell < \ell_{max}$, we define $RHS^{\ell}_{S-P} := D_{coarse}(V^{\ell} - \frac{1}{\rho^{n+1/2,\ell}} G\phi^{\ell})$ where the divergence operator D_{coarse} is defined to include only that contribution to the usual nodal divergence operator which comes from the coarse side of the coarse/fine interface. This would be equivalent to zeroing the coefficients $1/\rho$ on the fine grid and applying the standard nodal divergence operator.

If $\ell > 0$, we set $RHS^{\ell-1}_{S-P} := RHS^{\ell-1}_{S-P} + \frac{1}{r} D_{fine}(V^{\ell} - \frac{1}{\rho^{n+1/2,\ell}} G\phi^{\ell})$ where here the contribution to the divergence comes only from the fine side of the coarse/fine interface; this contribution is computed on the fine nodes along the interface and averaged onto the coarse nodes.

3.6.2 Synchronization of data

If $\ell < \ell_{max}$ and r level $\ell+1$ time steps have just been completed, the level ℓ and $\ell+1$ data must be synchronized.

First, we average $U^{n+1,\ell+1}$ and $S^{n+1,\ell+1}$ down onto the level ℓ grids wherever possible. This is a simple cell-centered averaging procedure, where for $r=2$ in 3-d, e.g., the level ℓ value becomes the average of the eight level $\ell+1$ values occupying that volume.

We then coarsen the level $\ell+1$ pressure down onto the level ℓ grids but in a time-averaged manner, such that the pressure at a level ℓ node underlying a level $\ell+1$ grid is defined to be the average over time of the r level $\ell+1$ values defined within the single level ℓ time step just completed.

In order to do the refluxing, we now define the cell-centered

$$V^{\ell}_{sync} = -\frac{1}{\Delta t^{\ell}\, Vol^{\ell}}(\delta F^{\ell}_{U,adv} + \frac{1}{\rho^{n+1/2,\ell}}\delta F^{\ell}_{U,visc})$$

$$S^{\ell}_{sync} = -\frac{1}{\Delta t^{\ell}\, Vol^{\ell}}(\delta F^{\ell}_{S,adv} + \delta F^{\ell}_{S,diff})$$

on cells in the rows of level ℓ cells immediately *outside* the level $\ell+1$ grids and set $V_{sync} = S_{sync} = 0$ elsewhere. This part of the correction to $U^{n+1,\ell}$ and $S^{n+1,\ell}$ will make the scheme conservative again; however, we can not correct $U^{n+1,\ell}$ directly because the correction is not divergence-free. Instead we store the correction in V_{sync} and project this correction field as part of the multilevel sync projection.

However, before we use V^{ℓ}_{sync} and S^{ℓ}_{sync} to correct the solution, we want to account for the fact that the advection velocities at level ℓ ($U^{ADV,\ell}$) and the time average over the level ℓ time step of the advection velocities at level $\ell+1$ do not necessarily agree at the coarse/fine interface. The resulting difference in fluxes is correctly accounted for by refluxing; however, even with refluxing, this mismatch means that quantities were effectively advected with a velocity field which does not satisfy the composite divergence constraint. In order to correct for this mismatch, which is captured in the velocity register δU, we must perform another level ℓ solve. We solve

$$D^{MAC}\left(\frac{1}{\rho^{n+1/2,\ell}}G^{MAC}(\delta e^{\ell})\right) = \tilde{D}^{MAC}(\delta U^{\ell})$$

on all grids at level ℓ for the correction δe^{ℓ}. Here $\tilde{D}^{MAC}$ is defined to be the MAC divergence operator evaluated only on the level ℓ cells immediately outside the level $\ell+1$ grids. 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 coarse grid points outside the fine grids.

We must now use the correction to the advection velocity to re-adjust the fluxes at all level ℓ faces. Because of memory considerations we did not store all the time-centered face states, so we now 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 project.

We define the correction velocity field

$$U^{\ell}_{corr} = \frac{1}{\rho^{n+1/2,\ell}}G^{MAC}(\delta e^{\ell})$$

and compute new fluxes, $U^\ell_{corr}\tilde{U}^{n+1/2,\ell}$ and $U^\ell_{corr}\tilde{S}^{n+1/2,\ell}$.

Again, the corrections to the velocity will not necessarily satisfy the divergence constraint, so we add the corrections here also to V^ℓ_{sync} and S^ℓ_{sync}. The velocity corrections will be projected in the sync projection. The scalar corrections could be added immediately, but are stored in S^ℓ_{sync} for convenience. That is,

$$V^\ell_{sync} := V^\ell_{sync} + \nabla \cdot (U^\ell_{corr}\, \tilde{U}^{n+1/2,\ell}),$$

$$S^\ell_{sync} := S^\ell_{sync} + \nabla \cdot (U^\ell_{corr}\, \tilde{S}^{n+1/2,\ell}).$$

Note that if $\ell > 0$, then we must modify the level $\ell - 1$ velocity registers and flux registers to account for the fact that we will be adding S^ℓ_{sync} to the level ℓ scalar fields and the projected component of V^ℓ_{sync} to the level ℓ velocity field. To do this, we set

$$\delta U^{\ell-1} := \delta U^{\ell-1} + \frac{1}{r} A^{\ell-1} \sum_{faces} U^\ell_{corr}$$

$$\delta F^{\ell-1}_{U,adv} := \delta F^{\ell-1}_{U,adv} + A^{\ell-1} \sum_{faces} \Delta t^\ell (U^\ell_{corr}\tilde{U}^{n+1/2,\ell})$$

$$\delta F^{\ell-1}_{S,adv} := \delta F^{\ell-1}_{S,adv} + A^{\ell-1} \sum_{faces} \Delta t^\ell (U^\ell_{corr}\tilde{S}^{n+1/2,\ell}),$$

using the same summing convention.

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, $\ell < q \leq \ell_{max}$ using conservative interpolation:

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

At this point there are two types of corrections we need to make to $U^{n+1,\ell}$: 1) that due to the fact that the level projections were done one level at a time, and therefore the composite divergence constraint is not satisfied at the coarse/fine interface, and 2) that due to the refluxing of velocity described above. The former is represented as a non-zero composite residual at the coarse/fine interface which represents the correction field which must be subtracted. Simply, if there were no refluxing we would just solve

$$L^{n+1/2}_\rho \phi^{SP}_1 = RHS^\ell_{S-P}$$

where RHS^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 $\frac{\Delta t^\ell}{\rho^{n+1/2}} G\phi^{SP}_1$ from the new-time velocity field at both levels.

The second part of the correction, that from the refluxing, is at this point in the algorithm carried in V^ℓ_{sync}, and is defined only on level ℓ since we did the MAC sync solve only on one level. If this field already satisfied the divergence constraint then we could simply add it to the velocity field; since there is no guarantee that it does, we want to add only the projected part of this field. To find the divergence-free part to add, we would first interpolate V^{ℓ}_{sync} to level $\ell+1$ and then take its composite divergence. We would solve

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

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

Combining these contributions, we see that the field we now 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 now note that $L_{\rho}^{n+1/2}(\phi_1^{SP} + \phi_2^{SP}) = RHS_{S-P}^{\ell} + D(V_{sync})$, and thus in practice we need not separate ϕ_1^{SP} from ϕ_2^{SP}. Rather, we define a single multilevel field ϕ_{sync} by solving $L_{\rho}^{n+1/2}\phi_{sync} = RHS_{S-P}^{\ell} + D(V_{sync})$, and once that solve is completed, we add the corrections to the velocity and pressure fields:

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

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

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

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

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 $RHS^{\ell-1}_{S-P}$:

$$RHS^{\ell-1}_{S-P} := RHS^{\ell-1}_{S-P} + \frac{1}{r} D_{fine}(V^{\ell}_{sync} - \frac{1}{\rho^{n+1/2,\ell}} G\phi^{\ell}_{sync})$$

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, $V^{\ell+1}_{sync}$, using conservative interpolation, onto fine grids at *all* finer levels, q, $\ell+1 < q \leq \ell_{max}$:

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

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

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 RHS_{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. As demonstrated in the next section, this additional expense is minimal and the effects can be nontrivial.

4 Computational Results

In this section we present several sets of results. In the first subsection we present convergence and accuracy results on a variety of two-dimensional problems, both constant and variable density, in Cartesian and axisymmetric geometries. (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 the second subsection we show results from a variable density Navier-Stokes calculation in three dimensions. Finally, in the third subsection, we present some timings of the algorithm in two and three dimensions on a DEC Alpha workstation.

4.1 Convergence Studies

The first set of calculations we present here is to demonstrate second-order convergence of the adaptive method on problems in which the data are smooth enough to achieve second-order accuracy on uniform grids. These include constant density inviscid flow in a box, for which we also demonstrate the importance of doing the sync projection; and variable density viscous flow in a box, with which we show that the method does retain its convergence properties when applied to the variable density Navier-Stokes equations. The second set of calculations compares the errors on the fine patch of a composite grid with the errors on the same patch in physical space of a uniform fine grid for a more realistic problem with dynamic regridding.

We note here that in Table 1 we contrast data from calculations in which the sync projection was not used, when it was done in a single-level form, and when it was done as a multilevel projection. In all later cases the calculations were done using a multilevel sync projection.

In Table 1 we present the L_1, L_2, and L_∞ norms of the errors in velocity, as well as the convergence rates of the errors, for constant density inviscid flow in a 1×1 box. The initial conditions for this problem are that of a vortex centered in the middle of the box:

$$u(x, y, t = 0) = -U \sin(\theta), \quad v(x, y, t = 0) = U \cos(\theta)$$

where

$$U = \begin{cases} 1 - a(r - r_0)^2 & \text{if } r < r_{vort} \\ \frac{b}{r} & \text{if } r \geq r_{vort} \end{cases}$$

with $r_0 = \frac{2}{3} r_{vort}$, $a = \frac{9}{4 r_{vort}}$, and $b = \frac{3 r_{vort}^2}{4}$, where $r_{vort} = 0.2$ and $r = \sqrt{(x - .5)^2 + (y - .5)^2}$. This problem has the property that both U and $\frac{\partial U}{\partial r}$ are continuous at $r = r_{vort}$. This analytically specified velocity field is discretely projected at $t = 0$ to make it approximately divergence-free before beginning the time-stepping. The boundary conditions are slip walls on all sides.

The calculations are run to $t = 0.9$, with a fixed time step per calculation. The coarse grid time step for a calculation with coarse grid spacing $1/n$ is $\Delta t = 0.9/n$, effectively equivalent to a CFL number of .9 since $U \leq 1$ in the domain.

For this table, as well as the ensuing ones, the headings of the columns such as $32 - 64$ represent the resolution of the two calculations used to compute the resulting error. These numbers are $1/h_c - 1/h_f$, where h_c and h_f are the mesh spacings at the *finest* level of each of the two calculations. The rate between the two columns of errors 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.

The rows of Table 1 describe the region refined ("Uniform" = all of the grid refined, "Centered" = (.25,.25) to (.75,.75) refined, "Offset" = (0.,0.) to (.5,.5) refined) and the type of sync projection used ("MLSP" = multilevel sync projection, "SLSP" = single-level sync projection, "NO SP" = no sync projection at all). For all of the calculations in Table 1 there was a single level of factor two refinement.

	L_1			L_2			L_∞		
Case	32-64	Rate	64-128	32-64	Rate	64-128	32-64	Rate	64-128
Uniform Grid	2.422	2.10	.5663	6.121	2.04	1.485	40.67	1.88	11.01
Centered MLSP	2.725	2.03	.6675	6.054	2.01	1.501	40.12	1.87	11.00
Centered SLSP	4.767	2.10	1.116	8.144	2.08	1.928	42.43	1.95	11.01
Centered NO SP	9.924	2.40	1.886	16.28	2.36	3.171	66.99	2.07	15.99
Offset MLSP	9.759	2.11	2.254	24.69	2.13	5.637	148.35	1.95	37.60
Offset SLSP	9.395	2.09	2.206	24.57	2.10	5.745	140.2	1.79	40.6
Offset NO SP	12.07	1.64	3.868	34.44	1.63	11.11	270.8	1.38	104.29

Table 1: Composite grid errors ($\times 10^3$) and convergence rates for x-velocity in constant density inviscid calculations.

We conclude from Table 1:

- For data which are smooth enough to be second-order accurate on a uniform grid, calculating the solution on an adaptive hierarchy, with the algorithm as presented in the text, is also second-order accurate, whether the adaptivity is placed as one would expect (around the vortex) or in a non-optimal manner (with a corner of the grid at the center of the vortex).

- The effect of the sync projection is non-trivial. As we see from the case of the offset patch without the sync projection, eliminating that step reduces the order of accuracy of the method to 1.6 in the L_1 and L_2 norms. Even for the centered patch, although the rates are not reduced, the errors themselves are noticeably larger when the sync projection is not done.

- There is some indication that doing the multilevel sync projection reduces the error for a multilevel calculation, but the evidence is not conclusive; the error reduction can be seen in the case of the centered patch, but not with the offset patch.

In regard to this last point, we mention another result which makes the case for a multilevel sync projection more strongly. A two-level, r-z constant density Navier-Stokes ($Re = 100$) calculation was run for 2100 coarse grid time steps with the multilevel sync projection and then again with the single-level sync projection. The domain was 1×30, and the base grid was 4×120, with one grid at level 1 covering $z \leq 1.5$ with a factor of two refinement. The level 1 grid was fixed in time, and the CFL number was 0.5. The initial velocity field was uniform, $U = (1, 0)$; the boundary conditions were inflow at $z = 0$, outflow at $z = 30$ and a no-slip wall at $r = 1$. The density and viscosity were $\rho = .01134, \mu = .0002268$, respectively.

The diagnostic used to evaluate the sync projection in this calculation was the volume-weighted sum in the r-direction of v^n, i.e. the net velocity flux in the z-direction. This sum should be the same at all z to the precision of the projection for a single-level calculation. For a multilevel calculation, the degree to which the sum is non-constant across the coarse/fine interface is the degree to which the flow is not divergence-free there. In the calculation described above, the variation in the sum when the multilevel sync projection was used was $O(10^{-8})$. When the single-level sync projection was used, the variation in the sum was $O(10^{-2})$ after 2100 coarse grid steps. The fact that the solution does not satisfy the composite divergence constraint at the interface results, in this case, from the fact that the elliptic problem has been solved on the coarse grid and the solution then interpolated to the fine grid. This conservative interpolation (not piecewise constant) does not preserve the diagnostic sum.

One can show analytically that the increased accuracy resulting from a multilevel as opposed to single-level sync projection is most significant when there is significant variation in the right-hand-side for the level ℓ $/(\ell+1)$ 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. The variation across the z =constant coarse/fine interface in the calculation above results from the viscous boundary layer forming on the $r = 1$ edge of the domain.

In Table 2 we show results similar to those in Table 1, but here we include errors and rates from a calculation in which the grids are dynamically destroyed and created throughout the calculation ("Moving Offset"). Specifically, letting T be the period of the calculations, the fine patch covers the lower left quadrant from $t = 0$ to $t = T/4$; it then covers the lower right quadrant, the upper right quadrant, and the upper left quadrant, from $t = T/4$ to $T/2$, $t = T/2$ to $3T/4$, and $t = 3T/4$ to T, respectively. The errors and rates from this calculation are contrasted with those from the uniform grid calculation and the offset fixed grid calculation. For all of the calculations in Table 2 there was a single refined level with a factor of two refinement.

This type of regridding, i.e., that in which new fine grids are created solely in regions which have been coarsely resolved thus far, requires that all data in the new fine patches be interpolated from the coarse grid. As a result we don't expect better accuracy than on a uniform coarse patch, but we note that we do retain second-order convergence despite the non-optimal regridding.

In Table 3 we show additional results from adaptive calculations of the problem described

	L_1			L_2			L_∞		
Case	32-64	Rate	64-128	32-64	Rate	64-128	32-64	Rate	64-128
Uniform Grid	2.422	2.10	.5663	6.121	2.04	1.485	40.67	1.88	11.01
Fixed Offset	9.759	2.11	2.254	24.69	2.13	5.637	148.35	1.95	37.60
Moving Offset	9.841	2.05	2.382	24.74	2.02	6.111	191.1	1.84	53.38

Table 2: Composite grid errors ($\times 10^3$)and convergence rates for x-velocity in constant density inviscid calculations. Here we contrast a fixed patch vs. a moving patch.

above; here, however, we contrast the error from calculations in which the resolution is achieved in different ways: uniform fine grid, a factor of two refinement, a factor of four refinement, or two factors of two refinement. In all cases the center quarter of the domain is at the same effective resolution, and the data from these calculations (effectively 32×32 or 64×64) are differenced with the data from a uniform 256×256 calculation, which for the purposes here is considered to be the exact solution. The solution is differenced and the error computed only over the center quarter of the domain. We see from this table that using a single factor of two refinement, or two factors of two, the accuracy is very close to that of the uniform fine grid calculation. The L_1 norm of the error is noticeably larger for the factor four refinement than for the other three cases (.014 vs. .009), although the L_2 and L_∞ norms are comparable. However, the L_1 norm is still significantly lower than that of the coarser uniform grid case (.00396 vs. .00953).

We conclude that when the refined patch is optimally placed, i.e., in the region where most of the error in the calculation is likely to occur, the actual error of the calculation is very close to that which one would get by placing a uniform fine grid over the domain. The fine grid results have not been significantly "contaminated" by their nesting within a coarser grid.

The second problem we consider in studying convergence of the adaptive projection method is a variable density Navier-Stokes calculation. The initial profile is again a vorticity distribution centered in a 1×1 box, but with a slightly different profile:

$$u(x, y, t = 0) = -\sin(2\pi(y - .5))\sin^2(\pi(x - .5))$$

$$v(x, y, t = 0) = \sin(2\pi(x - .5))\sin^2(\pi(y - .5)).$$

	L_1		L_2		L_∞	
Case	32-256	64-256	32-256	64-256	32-256	64-256
Centered: Factor 4	.01403	.003956	.01676	.004832	.03266	.01280
Centered: 2 Factors of 2	.009657	.002334	.01393	.003446	.04161	.01241
Centered: Factor 2	.009411	.002335	.01390	.003457	.04135	.01246
Uniform Grid	.009529	.002312	.01416	.003458	.04208	.01248

Table 3: Errors in x-velocity on centered fine patch only as calculated on a uniform grid vs. on composite grids with different levels of refinement, in constant density inviscid calculations.

This analytically specified velocity field is discretely projected at $t = 0$ to make it approximately divergence-free before beginning the time-stepping. The boundary conditions are no-slip walls on all sides.

The initial density field is

$$\rho(x, y, t = 0) = 1 + 0.25 \tanh(10(r - r_0))$$

where $r = \sqrt{(x - .5)^2 + (y - .5)^2}$ and $r_0 = 0.2$; the viscosity is set to the constant value $\mu = 0.01$.

These calculations are run to $t = 0.5$, also with a fixed time step per calculation. The coarse grid time step for a calculation with coarse grid spacing $1/n$ is $\Delta t = 0.5/n$, effectively equivalent to a CFL number of .5 since again the velocity is less than or equal to one in the domain.

	L_1			L_2			L_∞		
Case	32-64	Rate	64-128	32-64	Rate	64-128	32-64	Rate	64-128
Uniform Grid	9.264	1.87	2.529	11.85	1.96	3.037	48.35	2.53	8.372
Offset Patch	28.86	1.86	7.951	43.84	2.08	10.42	232.5	2.23	49.41

Table 4: Composite grid errors ($\times 10^4$) and convergence rates for x-velocity in variable density Navier-Stokes calculations.

	L_1			L_2			L_∞		
Case	32-64	Rate	64-128	32-64	Rate	64-128	32-64	Rate	64-128
Uniform Grid	3.234	2.45	.5920	5.920	2.39	1.127	29.96	1.80	8.574
Offset Patch	15.40	2.26	3.209	27.86	2.18	6.132	122.5	1.65	39.09

Table 5: Composite grid errors ($\times 10^4$) and convergence rates for density in variable density Navier-Stokes calculations.

In Table 4 we present the L_1, L_2 and L_∞ norms of the error in x-velocity; in Table 5 we show the norms of the error in density. We note here that the errors for the composite calculation do look closer to those on the uniform coarse grid than to those on the uniform fine grid, as in the constant density Euler case. This is again because not only is 75% of the problem domain in fact coarsely gridded, but the data on the fine patch have been advected from the coarse grid. However, the convergence rates for the composite grid calculations are comparable to those on the uniform grid.

The last convergence study shows that one can achieve comparable accuracy by optimal regridding as compared with uniform fine grid for the case of variable density Navier-Stokes. We present data from $r - z$ calculations of a light bubble rising under gravity in a constant density background. The re-gridding criterion is such that all of the bubble is always at the finest resolution.

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.

The data are evaluated at three times: $t = .014, .018, .022$, and contour plots of the density at those times are shown Figure 4; in these the velocity field is superimposed as vectors. As with the data in Table 3, here the calculation is first done on a uniform 256 × 512 grid, and this will be taken as the "exact" solution. Two other calculations are then run; the first on a uniform 64 × 128 grid, the second using a base grid of 16 × 32 and a factor of 4 refinement. Shown in Tables 6–8 are the errors in density, x-velocity, and y-velocity, respectively, from each of the latter two calculations, evaluated only on the region of the domain covered by the level 1 patch of the refined calculation, as calculated using the 256 × 512 solution as exact. Note that the L_1, L_2, and L_∞ norms of the error in velocity and density in the adaptive calculation differ by less than 10% in density and 26% in each component of velocity. It is not surprising that there are differences; in the adaptive calculation only 18.25% of the domain is at the finest resolution at any point in the calculation, the rest of the domain is at one fourth the resolution; in addition, the refined patch is non-static, so the data on the fine patch have been interpolated from the coarser level.

	$t = .14$			$t = .18$			$t = .22$		
Time	L_1	L_2	L_∞	L_1	L_2	L_∞	L_1	L_2	L_∞
Uniform	2.55	9.41	144	3.78	14.6	234	4.89	18.1	197
Refined	2.83	9.96	151	4.06	15.1	244	5.19	18.7	209

Table 6: Errors in density on the region around the bubble as calculated using a uniform 256 × 512 grid vs. as calculated using a factor of four refinement with a base grid of 16 × 64.

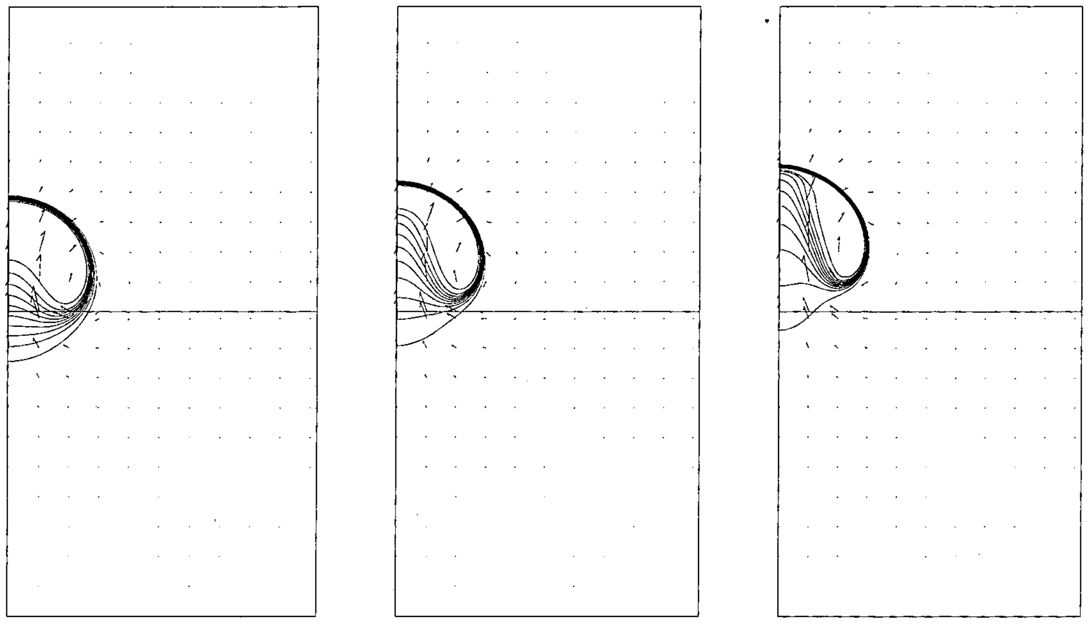
	$t = .14$			$t = .18$			$t = .22$		
Time	L_1	L_2	L_∞	L_1	L_2	L_∞	L_1	L_2	L_∞
Uniform	.000369	.000934	.00794	.000473	.00128	.0111	.000494	.00150	.0183
Refined	.000450	.00105	.00855	.000539	.00135	.0115	.000545	.00151	.0181

Table 7: Errors in x-velocity on the region around the bubble as calculated using a uniform 256 × 512 grid vs. as calculated using a factor of four refinement with a base grid of 16 × 64.

	$t = .14$			$t = .18$			$t = .22$		
Time	L_1	L_2	L_∞	L_1	L_2	L_∞	L_1	L_2	L_∞
Uniform	.000454	.00129	.0102	.000533	.00138	.00962	.000599	.00146	.0136
Refined	.000566	.00151	.0112	.000636	.00155	.0121	.000672	.00152	.0143

Table 8: Errors in y-velocity on the region around the bubble as calculated using a uniform 256 × 512 grid vs. as calculated using a factor of four refinement with a base grid of 16 × 64.

This calculation has identified an algorithmic difficulty which will require future modification. When a refined calculation was attempted that was identical to the one above but



(a) t = 0.014 (b) t = 0.018 (c) t = 0.022

Figure 4: Contour plots of density and superimposed velocity vectors for the 256 x 512 r-z bubble calculation at times t= 0.014, 0.018, 0.022 at which the errors were measured. The line across the center shows the two level 0 grids used in this calculation.

with half the width across the tanh profile, the calculation failed before reaching $t = 0.22$ because the density became negative. Running the same calculation without the MAC sync solve was successful.

Recall that the single-level MAC sync solve requires that the correction for scalars such as density be interpolated from the level at which the solve is done (in this case level 0) to the next finest level and above. The correction is done using the correction velocity field and face-state values of the scalar on the coarse grid, rather than re-computing advection terms at the finer resolution. In a region of steep gradients for fields with a large dynamic range this can result in unacceptable under- or over-shoot in the interpolated value, as we saw in this calculation. Future work will involve altering the algorithm so that this behavior does not occur. One possible algorithmic change is to modify the interpolation scheme so that it "redistributes" the interpolated quantity among the fine cells as necessary to avoid under- or over-shoot. An alternate approach would be to make the MAC sync solve a multilevel solve, so that interpolation is only used for levels $\ell+2$ and greater. However, this approach is not as simple as it might appear. The corrections being made to the level ℓ and the level $\ell+1$ data correspond to a mismatch in advection velocity which has accumulated over a full level ℓ time step. While one would expect the correction velocity (U_{corr}) to be small, it is possible for U_{corr} to be large enough that performing the correction at level $\ell+1$ could violate the level $\ell+1$ CFL constraint. This would necessitate a modified correction step with subcycling in time.

4.2 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 [17] 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 [21] 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 $512 \times 128 \times 384$. The base grid was $32 \times 8 \times 24$, 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 $256 \times 64 \times 96$, with $\Delta x_{finest} = 2$. 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

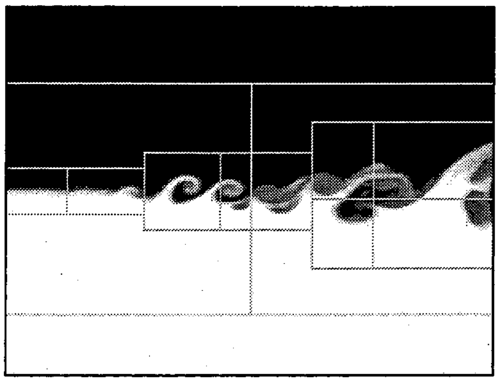
$$U(x = 0, y, z, t) = (1 + \Sigma_{i=1}^{10} m_i \sin(f_i t)) u_0(z - z_{pert}),$$

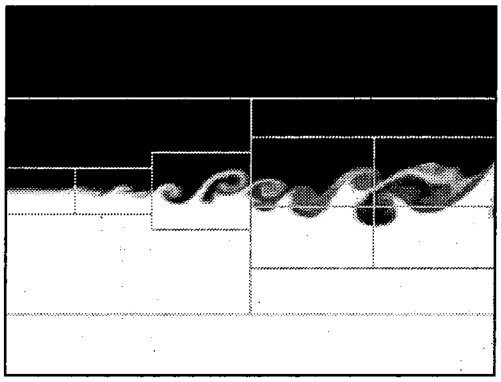
where $z_{pert} = 0.1 \sin(f_1 t) \sin(.22089323y)$. The frequencies were $f_1 = .219$, and $f_i = f_{i-1}/i$ for $2 \leq i \leq 10$, and the magnitudes were $m_1 = .01$, $m_2 = .75m_1$, $m_3 = .55m_1$, $m_4 = .44m_1$, and $m_i = 1.7m_1/i$ for $5 \leq i \leq 10$. The density of the fluid flowing in through $x = 0$ 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.

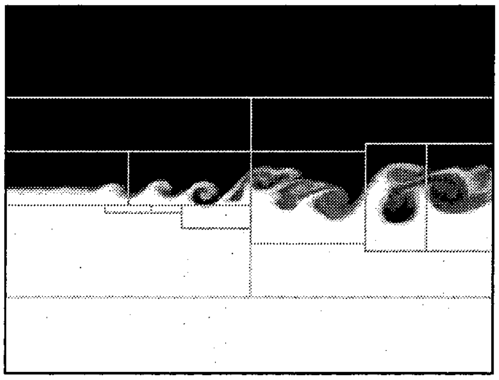
The flow requires about 100 coarse grid time steps for the initial perturbations to pass through the domain and the pattern of vortex formation to become established. For these initial cycles we adjusted the error criteria so that no level 2 grids were formed and so that level 1 grids followed the structures. 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.

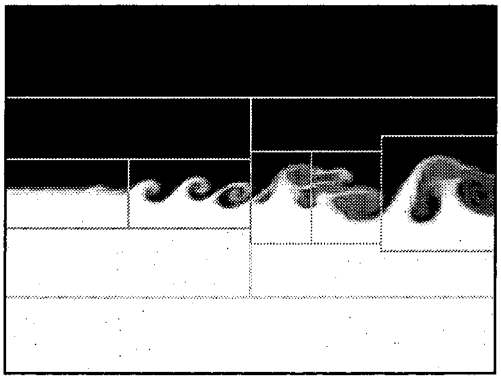
In Figures 5a-d we show a time sequence of density in the x-z cross-section centered spanwise in the domain. These "snapshots" are taken at intervals of 10 level 0 time steps; the times are: (a) $n = 205, t = 1585$, (b) $n = 215, t = 1630$, (c) $n = 225, t = 1673$, (d) $n = 235, t = 1715$. Recall that eight level 2 time steps are taken for each level 0 time step. Figures 5e-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 6 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.

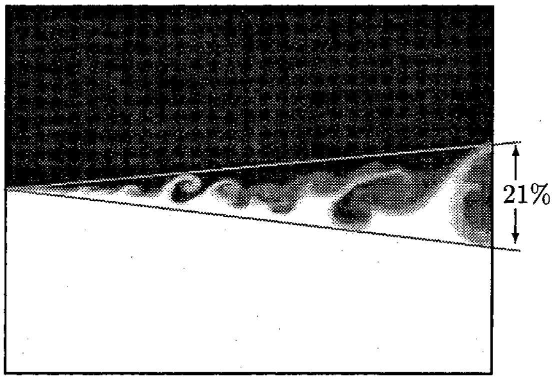
Finally, in order to have a more quantitative comparison with the experimental data we accumulated flow statistics from $n = 200$ to $n = 325$. In particular, we computed time- and spanwise-averaged values of the mean x-velocity and density and the density perturbation. (The spanwise averaging was across 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.


(a) t = 1585

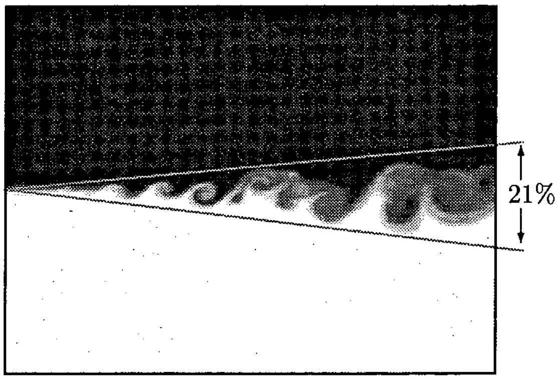

(b) t = 1630


(c) t = 1673


(d) t = 1715



(e) t = 1585



(f) t = 1673

Figure 5: (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.

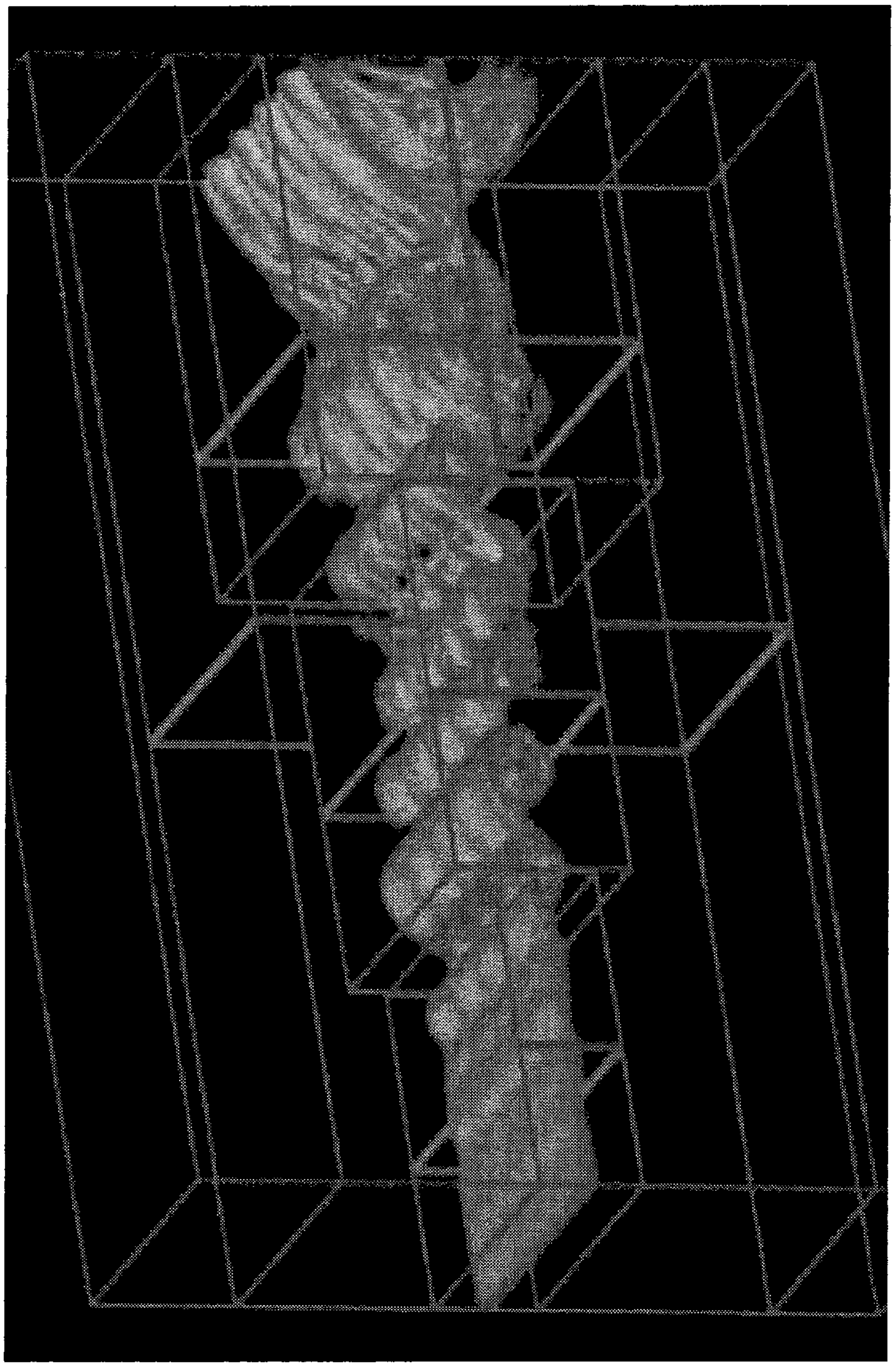


Figure 6: Three-dimensional rendering of vorticity at $t = 1585$ with the level 1 and level 2 grids superimposed. The domain is cropped slightly in the z-direction.

In Figure 7 the profiles are plotted on the same axes as the experimental data from [12] (for the mean values) and [17] (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.

4.3 Timings

Here we present some timings of the code as run on the variable density vortex-in-a-box problem presented in the previous section. Four cases are run, each with an effective fine grid resolution of 64×64 at the finest level. These cases are: uniform 64×64 grid, 32×32 base grid with one level of factor two refinement, 16×16 base grid with one level of factor four refinement, and 16×16 base grid with two levels of factor two refinement. The calculations here, as well as all those presented above, were run on a single-processor DEC Alpha. The initial conditions are those presented earlier; here the code is run in each case to $t = 1.$ with Δt determined as described above. Presented in Table 9 are the number of CPU seconds required to complete the calculation, as well as the CPU time per cell advanced as measured in μs / cell. This number is the total CPU time divided by the sum over all levels of number of cells at that level times the number of time steps that level is advanced, and is interesting for evaluating how the cost of the calculation scales with the size of the problem and refinement strategy. For a calculation with a base grid having N cells and a patch refined by a factor of r having M cells, run for p coarse time steps taking a total of T CPU μs, this number would be $T/((Np) + (Mrp))$ μs/cell, since rp time steps were taken at the finer level.

Also shown in Table 9 are the percentages of time spent at each level in the calculation doing level operations and the time spent doing synchronization operations (the sync projection and the MAC sync solve). In parentheses after each of the level numbers is the percentage of the total number of cells ($N + M$ above) at each level; recall, though, that a factor of r more time steps are taken a finer level than at a coarser level, and that the synchronization operations occur once per coarse grid time step.

For this same problem we investigated the time saved by doing a single-level rather than multilevel nodal sync projection. For the case with a 32×32 base grid and one factor of two refinement, the calculation with the single-level sync projection took about 13% less time than the calculation with the multilevel sync projection (72.3s vs. 83.8s). For the two-level r-z constant density Navier-Stokes problem, the difference in CPU time was approximately 5%. These percentages vary, of course, with the relative time spent in advancing all quantities at a single level vs. doing the sync projection.

The three-dimensional shear layer calculation presented above took approximately

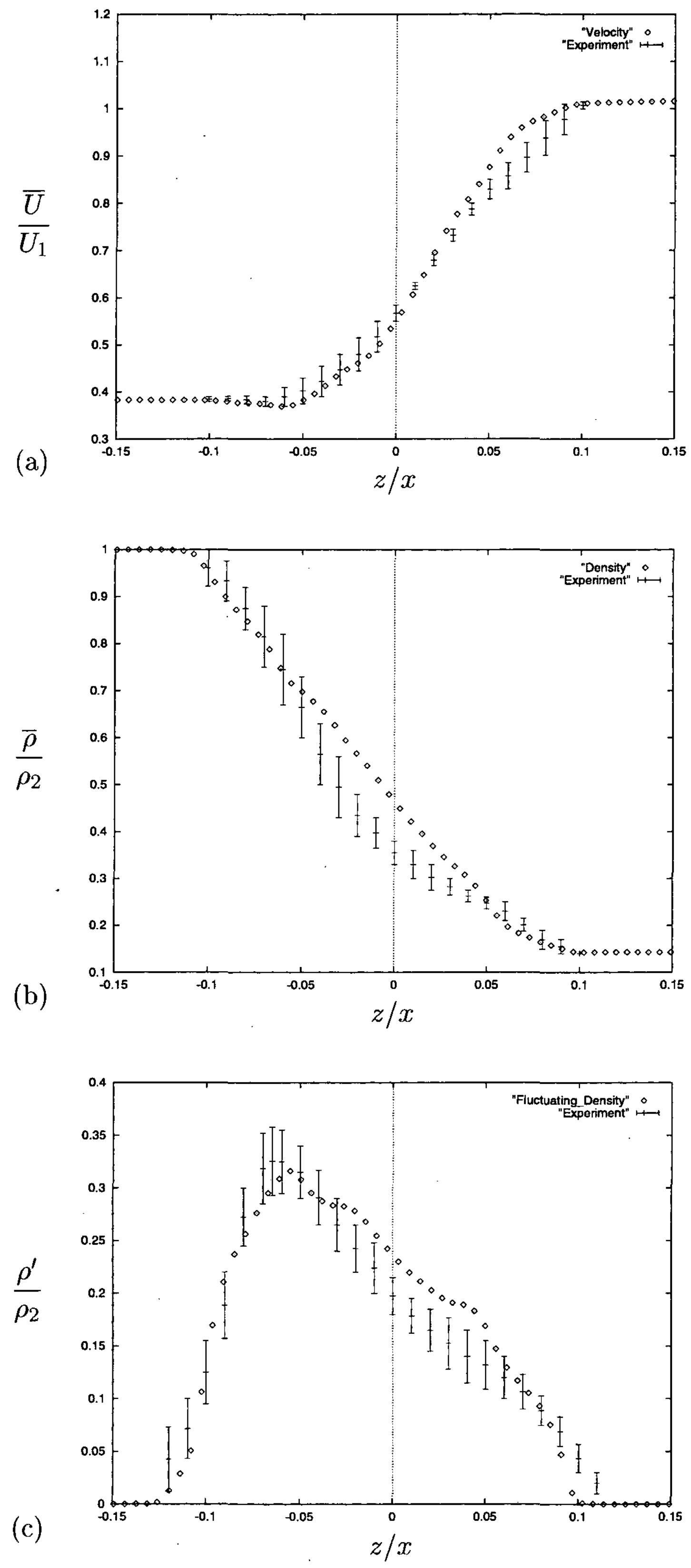


Figure 7: Mean velocity and density profiles and fluctuating density profile for the three-dimensional shear layer. The error bars on the mean profiles denote the experimental data band from Brown and Roshko [12]; the error bars on the fluctuating density profile denote the experimental band from Konrad [17].

	CPU Time		% of Time at Level				
Time	Total(s)	μs/cell	0	1	2	0-1	1-2
64^2 Uniform	133.5	255	100 (100)	–	–	–	–
32^2 Base + Factor of 2	83.8	426	23 (50)	49 (50)	–	28	–
16^2 Base + Factor of 4	52.7	367	7 (20)	74 (80)	–	19	–
16^2 Base + 2 Factors of 2	73.9	434	5 (15)	13 (27)	54 (58)	7	22

Table 9: Timings for uniform grid and refined grid calculations on a single-processor DEC Alpha

500–510μs / cell advanced on a single processor of a four-processor DEC Alpha. For this calculation, unlike the two-dimensional example above, there was significant output; both restart files and graphics files were output every five coarse grid time steps.

Preliminary timings were also computed on the CRAY C-90, and in some cases the code actually takes more CPU time to complete on the C-90 than on the DEC Alpha. In these cases, the nodal projections take about half as long on the C-90 as on the Alpha, but the cell-centered solves take about twice as long because they have not yet been optimized for vector machines. It is clear that 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 projection. These were the ratio of the final residual to original residual in the L_∞ norm for the equation to be considered solved. In practice, the tolerances must vary somewhat between levels in order for the synchronization equations on the coarser levels to be solvable; the numbers given above are for the level 0 solves.

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/or 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: modification of the algorithm as discussed in the previous section to avoid the problems associated with the current interpolation of scalar data following a single-level MAC sync solve; further 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. We are also 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. 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.

Acknowledgements

We would like to thank 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 9-point stencils have a bilinear form on each cell, the 2-d 5-point stencils are linear on each of two triangles in each cell, and the 3-d 7-point stencils do not have an obvious geometric interpretation—they are essentially an extension of the 2-d 5-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

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

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

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

With the contributions from the other three cells we then have

$$[D] = [D_{--}] + [D_{-+}] + [D_{+-}] + [D_{++}] \tag{A.3}$$

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,

$$[D]_o = \frac{1}{w_o} \left[\sum_{c \in S_c(o)} [D_c]_o + \frac{1}{r^d} \left(\sum_{f \in S_f(o)} [D_f]_o + \sum_{p \in F(o)} \chi(p) \sum_{f \in S_f(p)} [D_f]_p \right) \right]. \tag{A.4}$$

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

$$\chi(p) = \frac{(r - |p_x - o_x|)(r - |p_y - o_y|)}{r^2} \tag{A.5}$$

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:

$$\chi(p) = \frac{(r - |p_x - o_x|)(r - |p_y - o_y|)(r - |p_z - o_z|)}{r^3}. \tag{A.6}$$

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:

$$w_o = \sum_{c \in S_c(o)} \frac{1}{4} + \sum_{f \in S_f(o)} \frac{1}{4r^2} + \sum_{p \in F(o)} \chi(p) \sum_{f \in S_f(p)} \frac{1}{4r^2}. \tag{A.7}$$

In 3-d, analogously,

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

For the linear operator L_ρ we have several different stencils. Bilinear basis functions in 2-d give a 9-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:

$$[(D\sigma G)_{+-}] = \frac{\sigma^{(x)}_{+-}}{6\Delta x^2} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & 2 \\ 0 & -1 & 1 \end{bmatrix} + \frac{\sigma^{(y)}_{+-}}{6\Delta y^2} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & -1 \\ 0 & 2 & 1 \end{bmatrix}. \tag{A.9}$$

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

$$[(D\sigma G)] = [(D\sigma G)_{--}] + [(D\sigma G)_{-+}] + [(D\sigma G)_{+-}] + [(D\sigma G)_{++}]. \tag{A.10}$$

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

$$[(D\sigma G)]_o = \frac{1}{w_o} \left[\sum_{c \in S_c(o)} [(D\sigma G)_c]_o + \frac{1}{r^d} \left(\sum_{f \in S_f(o)} [(D\sigma G)_f]_o + \sum_{p \in F(o)} \chi(p) \sum_{f \in S_f(p)} [(D\sigma G)_f]_p \right) \right]. \tag{A.11}$$

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 5-point formula obtained from linear basis functions over triangles differs from the 9-point formula only in having a different unit cell:

$$[(D\sigma G)_{+-}] = \frac{\sigma^{(x)}_{+-}}{2\Delta x^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \frac{\sigma^{(y)}_{+-}}{2\Delta y^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{A.12}$$

The extension of this formula to 3-d to obtain the 7-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

$$(I^{\ell-1}_{\ell} s)_o = w_o s_o + \frac{1}{r^d} \sum_{p\in G(o)} \chi(p) s_p. \tag{A.13}$$

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

$$[rD] = \left(\frac{1}{2\Delta r}\begin{bmatrix} -r_- & r_+ \\ -r_- & r_+ \end{bmatrix}, \frac{1}{2\Delta z}\begin{bmatrix} r_- & r_+ \\ -r_- & -r_+ \end{bmatrix} + \frac{\Delta r}{12\Delta z}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\right), \tag{A.14}$$

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.

$$[\overline{G}] = \left(\frac{1}{2\Delta r}\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}, \frac{1}{2\Delta z}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} + \frac{\Delta r}{12r\Delta z}\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}\right)^{\mathrm{T}}, \tag{A.15}$$

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 formulas 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

$$[(rD(1/\rho)G)_{+-}] = \frac{\sigma^{(r)}_{+-}}{6\Delta r^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & 2 \\ 0 & -1 & 1 \end{bmatrix} + \frac{\sigma^{(z)}_{+-}}{6\Delta z^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & -1 \\ 0 & 2 & 1 \end{bmatrix} + \frac{\sigma^{(z)}_{+-}\Delta r}{12 r_+ \Delta z^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix}. \tag{A.16}$$

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

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

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

[17] 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.

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

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

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

[21] 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.

[22] 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. 1996. submitted for publication.

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

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

[25] 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.

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

[27] David E. Stevens and Christopher S. Bretherton. A new forward-in-time advection scheme and adaptive multilevel flow solver for nearly incompressible atmospheric flow. 1995. submitted for publication.

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

Prepared for the U.S. Department of Energy under Contract No. DE-AC03-76SF00098