JOURNAL OF COMPUTATIONAL PHYSICS **127,** 158–178 (1996)
ARTICLE NO. 0166

A Projection Method for Locally Refined Grids

MICHAEL L. MINION

Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, New York 10012

Received June 20, 1995; revised March 11, 1996

A numerical method for the solution of the two-dimensional Euler equatons for incompressible flow on locally refined grids is presented. The method is a second-order Godunov-projection method adapted from Bell, Colella, and Glaz. Second-order accuracy of the numerical method in time and space is established through numerical experiments. The main contributions of this work concern the formulation and implementation of a projection for refined grids. A discussion of the adjointness relation between gradient and divergence operators for a refined grid MAC projection is presented, and a refined grid approximate projection is developed. An efficient multigrid method which exactly solves the projection is developed, and a method for casting certain approximate projections as MAC projections on refined grids is presented.

CONTENTS

1. INTRODUCTION

In this paper a method is developed to simulate the evolution of an inviscid, incompressible, constant density flow in two dimensions on a computational grid using local grid refinement. The method is based on the second-order projection method of Bell, Colella, and Glaz [6]. Convergence studies will be presented that show that the method is second-order accurate in space and time, even in the presence of local grid refinement.

The evolution of an incompressible, inviscid fluid with constant density in a region of the plane is given by the following form of the Euler equations. Let

$$U = U(x, y, t) = (u(x, y, t), v(x, y, t))$$

denote the horizontal and vertical components of the velocity field at the point (x, y) at time t, and similarly define $p = p(x, y, t)$ as the pressure. Given some domain Ω with a solid wall boundary $\partial\Omega$, the evolution equations for U are

$$\begin{aligned} U_t &= -uU_x - vU_y - \nabla p \\ \nabla \cdot U &= 0. \end{aligned} \tag{1}$$

The boundary condition for the velocity at physical boundaries is the no-flow condition

$$U(x, y, t) \cdot \hat{n} = 0 \quad \text{for } (x, y) \in \partial\Omega, \tag{2}$$

where $\hat{n}$ is the outward unit normal vector on $\partial\Omega$.

The following decomposition theorem, when applied to the Euler equations, puts them into a useful alternative form from which numerical projection methods are derived.

THEOREM 1.1 (Hodge decomposition). *Given a simply connected domain Ω with compact support and smooth boundary $\partial\Omega$ and a vector field $U = (u, v)$ on that domain, U can be uniquely decomposed in the form*

$$U = U^D + \nabla\phi, \tag{3}$$

where

$$\nabla \cdot U^D = 0 \quad \textit{in } \Omega, \qquad U^D \cdot \hat{n} = 0 \quad \textit{on } \partial\Omega.$$

Proof. Taking the divergence and normal component of both sides of (3) yields

$$\Delta\phi = \nabla \cdot U \tag{4}$$

and

$$\frac{\partial\phi}{\partial\hat{n}} = U \cdot \hat{n} \quad \text{on } \partial\Omega. \tag{5}$$

0021-9991/96 $18.00

Equations (4) and (5) define a Neuman problem which is known to have a unique solution, up to an additive constant in ϕ, provided the solvability condition

$$\int_\Omega \nabla \cdot U = \int_{\partial\Omega} U \cdot \hat{n} \tag{6}$$

is satisfied [25]. Equation (6) is simply the divergence theorem; hence, it is trivially satisfied. With ϕ so defined, set $U^D = U - \nabla\phi$. It is then trivial to check that the conditions of the theorem are met. ▌

The preceding proof also supplies the procedure for extracting the divergence-free part U^D from some vector field U, namely solving a Poisson problem with Neumann boundary conditions for ϕ. Since U^D is uniquely determined, we can define the projection $\mathbf{P}$ by

$$\mathbf{P}(U) = U^D.$$

Note also that

$$(I - \mathbf{P})(U) = \nabla\phi.$$

The boundary condition on U^D in Theorem 1.1 is not restricted to $U^D \cdot \hat{n} = 0$. The projection $\mathbf{P}$ can be defined so that $U^D \cdot \hat{n} = g$ for any g such that

$$\int_{\partial\Omega} g = 0.$$

This condition is necessary for $\mathbf{P}$ to exist, as can be seen from the divergence theorem. The effect of prescribing non-zero boundary values for the projection on the above procedure for extracting the divergence-free part of a flow is to simply change the Neumann boundary conditions given in Eq. (5) to

$$\frac{\partial\phi}{\partial\hat{n}} = U \cdot \hat{n} - g \quad \text{on } \partial\Omega. \tag{7}$$

The application of the projection $\mathbf{P}$ to Eq. (1) yields the equivalent projection form of the Euler equations

$$U_t = \mathbf{P}(-u\,U_x - v\,U_y). \tag{8}$$

This has the desirable effect of eliminating the pressure term and the divergence constraint from Eqs. (1).

Projection methods in general attempt to approximate the Euler equations by discretizing Eq. (8) directly. Chorin was the first to propose a numerical method based on this idea [14–17]. In the original method, the velocities are upated by first advancing them without regard to the divergence constraint, and this update is then projected onto the space of discrete divergence-free vector fields. Numerous variations on and improvements of the original method have been advanced. (For a survey of projection methods, see Gresho [20], Peyret and Taylor [33], or Simo [34].) Most have in common a predictor–corrector type process in which a first approximation U^* to the velocity is computed and then a discrete projection is applied to U^* to yield an update of the velocity. An interesting second-order projection method for the incompressible Navier–Stokes equations which uses a higher order Godunov-type procedure to approximate the advective terms was introduced by Bell, Colella, and Glaz (BCG) [6]. Extensions of this method have been successfully used in a number of different regimes, and an inviscid version of it serves as the basis for the projection method on refined grids presented in this work.

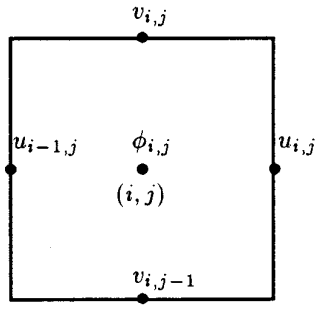

FIG. 1. Cell variable locations for staggered grids.

The emphasis here concerns the formulation and implementation of a discrete projection operator on a refined grid structure. Issues relating to the necessary modifications to handle the viscous equations will be addressed in future work. Section 2 covers some of the details one must consider when implementing discrete projections and Section 3 relates these issues to approximate projections. Section 4 contains a description of the refined grid projection method. A multigrid procedure for solving the projection is presented in Section 5. Convergence results from the method are contained in Section 6, and a summary of the pertinent issues contained in the paper appears in the final section.

2. MAC PROJECTIONS

Before discussing the refined-grid projection method, several issues concerning the motivation for the method will first be discussed in terms of the single grid case. In 1965, Harlow and Welch [22] presented a numerical method for two-dimensional flow which uses a staggered computational grid. In this method, vector quantities such as the velocities u and v are only represented on cell edges while scalar quantities such as the pressure or the divergence are represented at cell centers. (See Fig. 1.) Assum-

ing a square computational cell with width h, arranging the data in this manner allows one to define a compact, second-order divergence operator D,

$$D(U)_{i,j} = \frac{u_{i,j} - u_{i-1,j}}{h} + \frac{v_{i,j} - v_{i,j-1}}{h}, \tag{9}$$

and likewise a gradient operator G

$$G(\phi)_{i,j} = \left(\frac{\phi_{i+1,j} - \phi_{i,j}}{h}, \frac{\phi_{i,j+1} - \phi_{i,j}}{h} \right). \tag{10}$$

Here the first component of G, which will be denoted G_1, is defined at cell edges where the horizontal velocity u is defined. Likewise, the second component G_2 is available at the same cell edges as v. Taking the terminology from Harlow and Welch, the divergence defined in (9) will be referred to as a MAC divergence.

Another advantage of using staggered velocities is that for rectangular domains, the no-flow boundary condition can be set explicitly at walls. The main disadvantage of staggered grids is that the horizontal and vertical velocities are not available at the same points. This often prevents the straightforward evaluation of derivative terms through finite differences. Methods using staggered grids usually average or interpolate velocity components at points at which they are not available. Unfortunately, averaging values tends to smear fine structures in the flow which is undesirable when studying flows with small viscosities.

Given a computational domain Ω, let $\{(i, j)\}_\Omega$ denote the coordinates of the cells that cover Ω. The boundary $\partial\Omega$ consists of a set of cell edges ξ_k, which are indexed by a single integer k. For a staggered velocity U defined on Ω, at each of the cell edges ξ_k only one of the velocity components u or v is defined. One can define cell-edge fluxes $F_k(U)$, for each ξ_k, that are the discrete representation of $U \cdot \hat{n}$ on $\partial\Omega$ by specifying $F_k(U)$ to be the component of the velocity located at ξ_k with a sign given by the sign of the corresponding component of the unit vector normal to ξ_k. For example, if ξ_k is the top edge of cell (i, j) in Ω, then $F_k(U) = v_{i,j}$. Likewise, $F_k(U) = -v_{i,j}$ if ξ_k is the bottom edge of cell (i, j). The values of $F_k(U)$ are specified by boundary conditions on the physical domain. One can recognize the divergence in (9) at a given cell as the sum of the four cell-edge fluxes $F_k(U)$ multiplied by the length of the edge with the sum then divided by the area of the cell, namely h^2. Hence the form of D at a cell adjacent to a physical boundary uses the prescribed values of $F_k(U)$ in place of the velocities at cell edges that correspond to the boundary.

An important fact about the MAC divergence is that it is conservative; i.e., with these definitions, the following discrete version of the divergence theorem is trivially true.

THEOREM 2.1 (Discrete divergence theorem). *Given a computational grid Ω composed of cells $\{(i, j)\}_\Omega$ whose boundary is given by the set of edges ξ_k and any discrete staggered vector field U defined as in figure 1, if D and F_k are defined as above, then*

$$\sum_{\{i,j\}_\Omega} D(U)_{i,j} h^2 = \sum_k F_k(U) h. \tag{11}$$

This theorem will later be used to provide a solvability condition for the multigrid procedure on refined grids.

With the above definitions the following discrete version of the Hodge decomposition (1.1) can be proven. (See [30] for proof.)

THEOREM 2.2. *Let Ω be a computational domain consisting of the cells with indices $\{(i, j)\}_\Omega$ bounded by the set of cell edges ξ_k. Given a discrete vector field U with boundary fluxes given by $F_k(U)$, U can be uniquely decomposed into the form*

$$U_{i,j} = U^D_{i,j} + G(\phi)_{i,j}, \tag{12}$$

where

$$D(U^D)_{i,j} = 0, \quad F_k(U^D) = 0. \tag{13}$$

Since this decomposition exists, a discrete projection operator can be defined by

$$P(U)_{i,j} = U^D_{i,j}.$$

This operator will be referred to as a MAC projection.

The proof of the above theorem follows the continuous case by defining

$$P(U)_{i,j} = U^D_{i,j} = U_{i,j} - G(\phi)_{i,j},$$

where ϕ is the solution of the discrete Poisson equation

$$DG(\phi)_{i,j} = D(U)_{i,j} \quad \text{with } F_k(G(\phi)) = F_k(U). \tag{14}$$

The proof that a unique solution of this system exists depends on an adjointness or orthogonality condition on the operators D and G

$$\langle G(\phi), U \rangle_V + \langle \phi, D(U) \rangle_S = F(U, \phi), \tag{15}$$

where

$$F(U, \phi) = \sum_k \left[\phi_k + \frac{h}{2} F_k(G\phi) \right] F_k(U) h. \tag{16}$$

Here the vector inner product $\langle W, U\rangle_V$ and the scalar inner product $\langle \phi, \psi\rangle_S$ are simply the sum of the component-wise products of the fields in question, and ϕ_k refers to the value of ϕ at the center of the cell containing the edge ξ_k. Since $\phi_k + (h/2)\, F_k(G\phi)$ is an approximation to $F_k(\phi)$, Eq. (15) is the discrete analog of the identity for continuous functions

$$\int_\Omega \nabla\phi \cdot U + \int_\Omega (\nabla \cdot U)\phi = \int_{\partial\Omega} \phi U \cdot \hat{n}.$$

If one applies Eq. (16) to a field U^D which satisfies $D(U^D)_{i,j} = 0$ and $F_k(U^D) = 0$, it is immediately apparent that

$$\langle G(\phi), U^D\rangle_V = 0; \tag{17}$$

i.e., divergence free fields are orthogonal to gradients of scalars in this inner product. This in turn is used to guarantee that P is well defined. Equation (17) can also be used to show that the projection P has the norm reducing property

$$\|P(U)\|_V^2 \le \|U\|_V^2,$$

where the norm in this inequality is the one induced by the vector inner-product. This inequality is used in [17] to show that the overall method is stable.

Note that the operator DG appearing in Eq. (14) is the familiar five-point Laplacian operator

$$\begin{aligned} D(G(\phi))_{i,j} &= L^5(\phi)_{i,j} \\ &= \frac{-4\phi_{i,j} + \phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1}}{h^2} \end{aligned} \tag{18}$$

in grid interiors. At physical boundaries, the form of L^5 must be modified to include values of $F_k(G(\phi))$ as in the definition of D at physical boundaries.

3. APPROXIMATE PROJECTIONS

Since it was introduced, the BCG method has served as a basis for several more involved methods. Bell and Marcus have presented a method based on BCG for variable density flows [8], and Lai *et al.* for reactive flows [27, 26]. Howell [23], Bell and Howell [24], and Almgren *et al.* [1] have also used variations of the BCG method as the basis for methods on refined grids. Despite these promising applications of the BCG method, the Godunov/projection method combination has been an uneasy marriage. Many of the details of the Godunov method require a cell-centered discretization of physical space in which both velocity components u and v are represented at the center of computational cells. In order to enforce the divergence constraint on these cell-centered velocities, a cell-centered projection operator must be constructed. The natural way to define cell-centered divergence and gradient operators is by the centered difference approximations

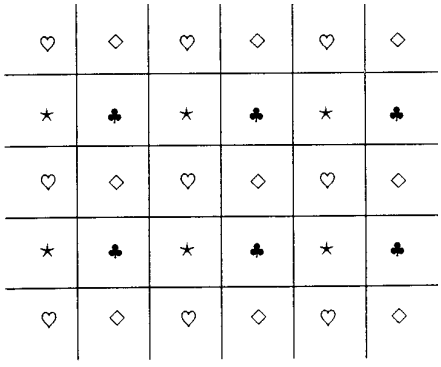

FIG. 2. Decoupled stencil for $\tilde{L}^5$.

$$D^\circ(U)_{i,j} = \frac{u_{i+1,j} - u_{i-1,j}}{2h} + \frac{v_{i,j+1} - v_{i,j-1}}{2h}$$

and

$$G^\circ(\phi) = \left(\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2h}, \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2h}\right).$$

These divergence and gradient operators satisfy the adjointness condition (17) and, hence, it can be shown that a cell-centered projection P° is well defined, where

$$D^\circ(P^\circ(U)) = 0$$

and $P^\circ(U)$ satisfies an analog of the no-flow condition at physical boundaries.

Unfortunately, the composition of $D^\circ G^\circ$ that appears in the Poisson problem associated with P° is an approximation to the Laplacian given by

$$\begin{aligned} D^\circ(G^\circ(\phi))_{i,j} &= \tilde{L}^5(\phi)_{i,j} \\ &= \frac{-4\phi_{i,j} + \phi_{i+2,j} + \phi_{i-2,j} + \phi_{i,j+2} + \phi_{i,j-2}}{4h^2}. \end{aligned}$$

The stencil for $\tilde{L}^5$ is similar to that of the more standard five-point Laplacian L^5, except that the points in the stencil are separated by $2h$. This causes the stencil for $\tilde{L}^5$ to decouple onto four distinct subgrids which are represented in Fig. 2 by hearts, stars, diamonds, and clovers. The value of $\tilde{L}^5$ on any subgrid depends only on values contained on that subgrid (except possibly at boundaries). The null space of $\tilde{L}^5$ therefore contains (for periodic domains) an oscillatory mode consisting of a different constant on each subgrid. This mode is also in the null space of the divergence

operator; hence, if such a mode is introduced into the velocity field by some means, perhaps near sharp gradients in the pressure or velocity, the projection will not remove it. In the method for reacting flows appearing in [26], a "filtering" procedure is employed every time step to remove such oscillations from the velocity fields.

In addition, if a multigrid-based method is to be used to solve the Poisson equation associated with the projection, the decoupling of the stencil of $\tilde{L}^5$ must be considered in defining the multigrid operators. (See [7] for a multigrid method for this projection formulation.) For a refined grid method, it is very desirable to use the multigrid method because it fits naturally into the refined grid structure. Unfortunately, implementing multigrid operators that respect the decoupling of $\tilde{L}^5$ on refined grids is very complicated. (Such a method is described in [23].)

In order to avoid using $\tilde{L}^5$ while still using cell-centered velocities, a cell-centered operator $\tilde{P}$ constructed from the MAC projection operator P is used here. It will be shown that this operator is equivalent to certain so called "approximate projections" that have recently appeared. The basic idea used in constructing $\tilde{P}$ is to apply P to staggered cell-edge velocities that are interpolated from cell-centered velocities. Then, an interpolation of the staggered gradient defined by $I - P$ is subtracted from the original cell-centered velocities.

Specifically, denoting cell-centered values by U^c, using simple averaging for interpolation yields the cell-edge velocities $\tilde{U}_{i,j}$

$$\tilde{U}_{i,j} = \left(\frac{u^c_{i+1,j} + u^c_{i,j}}{2}, \frac{v^c_{i,j+1} + v^c_{i,j}}{2} \right).$$

Applying the MAC projection operator to $\tilde{U}$ yields a staggered gradient $(I - P)(\tilde{U}) = G(\phi)$. Then $\tilde{P}(U^c)$ is defined by

$$\tilde{P}(U^c) = U^{cD},$$

where

$$u^{cD}_{i,j} = u^c_{i,j} - (G_1(\phi)_{i,j} + G_1(\phi)_{i-1,j})/2$$
$$v^{cD}_{i,j} = v^c_{i,j} - (G_2(\phi)_{i,j} + G_2(\phi)_{i,j-1})/2.$$

The velocity field U^{cD} produced by $\tilde{P}$ is not divergence-free in terms of either D or D°, and also $\tilde{P}^2 \neq \tilde{P}$. Therefore, $\tilde{P}$ is not by definition a projection operator but, instead, is expected to approximate the continuous projection operator to second-order accuracy. Such operators have been recently given the name "approximate projections" [26, 2].

An approximate projection operator is presented by Lai for a projection method for reacting flows [26]. If one considers the form of this operator when applied to a flow with constant density, it can be seen that the projection operator is the same as P° which results from the operators D° and G° except that the Laplacian $\tilde{L}^5$ in the Poisson problem associated with the projection is replaced by L^5. Specifically, the Poisson problem solved is

$$DG(\phi) = D^\circ(U^c)_{i,j},$$

with some boundary conditions derived for ϕ. However, it is easy to check that

$$D^\circ(U^c)_{i,j} = D(\tilde{U})_{i,j}.$$

It is also trivial to see that $G^\circ(\phi)_{i,j}$ is the simple average of cell-edge gradient components of the gradient G; hence, the constant density approximate projection in [26] is exactly $\tilde{P}$ defined above. A straightforward generalization of the arguments presented above can be used to present the variable density version of the approximate projection in [26] in terms of a MAC projection. The main advantages of posing approximate projections in terms of MAC projections is that it makes clear the appropriate boundary conditions for the approximate projection Poisson problem and, also, that the well-posedness of the MAC projection guarantees the well-posedness of the approximate projection. It also facilitates the extension of the approximate projection to refined grids. It should be noted also that many so-called "pressure-Poisson" methods which use the divergence constraint to derive a Poisson equation for the pressure are in essence approximate projection methods since the velocity fields thus constructed do not in general satisfy a discrete divergence constraint [19].

The method described in this paper employs an approximate projection based on a MAC projection similar to $\tilde{P}$ appearing above. This approximate projection is applied after each time step to approximately enforce the divergence constraint, and it is described in detail in Section 4.5.

4. THE REFINED GRID PROJECTION METHOD

The BCG method on which the method in this paper is based uses a higher-order Godunov procedure adapted from compressible flow algorithms [18] for approximating the advective terms in the Euler equations. It is designed to robustly treat sharp gradients or discontinuities in the velocity fields without introducing unphysical oscillations into the numerical solutions. On a single grid, the method here most closely resembles the variation of BCG appearing in Bell, Colella, and Howell [7], and is also similar to the constant density version of the method presented by Lai in [26]. The entire method contains several steps and is somewhat involved. In particular, the evaluation of the advective terms is not particularly straightforward, and

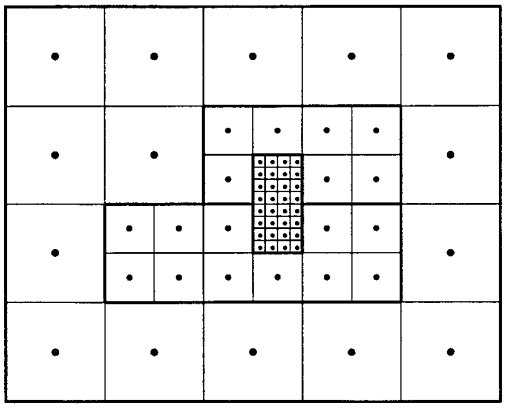

FIG. 3. Cell-centered grid refinement with refinement factors 2 and 4. The coarsest grid has two child grids, and the finest grid has two parent grids.

only a brief summary of the steps involved are included here with an emphasis on the details relating to the refined grid extensions. Therefore the reader is encouraged to refer to the original method of Bell, Colella, and Glaz or subsequent papers [6–8, 26, or 30], for motivation for and details of the steps involved.

4.1. *Local Grid Refinement*

The grid refinement strategy employed in this paper allows rectangular blocks of grid cells to be subdivided by some refinement ratio with fine grid cell edges lining up with coarse grid cell edges. (See Fig. 3.) Grids that cover these regions of refinement will be referred to as child grids while a grid containing a child grid will be called a parent. To properly fit into the multigrid framework used in the projection step, the refinement factor r, which is equal to the ratio of parent to child grid spacing, must be a power of two—specifically two, four, or eight.

A collection of grids with the same-size computational cells will be called a grid level. Refinement factors between different grid levels can differ, but in the current implementation, the refinement factor for all child grids in a particular level must be the same. The number of grid levels is limited only by the memory limitations of the computer being used. The coarsest level grid, or base grid, will always have the number of grid cells in each dimension equal to a power of two to facilitate the solution of the Poisson equation associated with the projection.

The following illustrates the notation that will be used for specifying refined grid variables. $(U^n))^{l,k}$ refers to the velocity at the nth time step represented on the kth grid of the lth level. Likewise $D(U^{n+1/2})^{l,k}_{i,j}$ refers to the value of the divergence of U at the i, jth cell on the kth grid of the lth level at time $n + \frac{1}{2}$. Often the k or l index will be implied. The coarsest level will be denoted with $l = 0$ with subsequently finer levels corresponding to increasing l.

Where refined grid data exist on a parent grid, the parent grid data are defined as an average of child grid data. In the case of data represented on cell edges, this average is taken over the r child cell-edge values corresponding to the parent cell edge. For cell-centered data, parent values are set to the average of the r^2 child values contained in that cell. Whenever grids of the same level are contiguous, cell-edge values on the shared edge will be equal.

For computational reasons, it is convenient to place additional rows of cells around child grids. It is then possible to assign values in cells around child grids so that finite difference operators near coarse–fine grid interfaces can be implemented with a convenient stencil. Whenever a child grid operator is defined across a coarse–fine boundary, it is required to depend exclusively on values from the next coarsest level of grid refinement and the values in the child grid itself. This is equivalent to requiring that cells from which data are used to set child grid boundary cells belong only to the next coarsest level of grid refinement and the child grid. A consequence of this is that there must be at least one layer of cells in the next coarsest level of refinement surrounding each child grid. This does not imply that the edges of a parent and child grid cannot coincide. At physical boundaries, the required extra layer of cells surrounding a child grid can be the boundary cells of the parent grid. Also, in the case where a parent grid is contiguous to another grid with the same cell size, a child grid can end or even lie across the parent grid interface.

In the implementation, additional cells are also located around the outside of the boundary of the physical domain. Physical variables in these cells can be given values so that the stencil of finite difference operators near domain boundaries does not have to be altered from the form used in the interior of grids. The values assigned to these cells will always be derived from the form of the difference operators near boundaries.

4.2. *The Refined Grid MAC Projection*

A second-order discrete MAC-divergence operator D can be defined on a refined grid structure with staggered velocities in the same way as for a single grid. For each cell, the divergence is defined at the cell center as the sum of centered differences of cell edge velocities (i.e., as in Eq. 9). This definition of the divergence has two important consequences. First, the divergence of a coarse grid cell that has been refined is automatically the average of the divergences of the fine grid cells it contains. This is consistent with the above rules for forming coarse grid values from fine grid values. Second, the above fact provides a trivial proof of a refined grid version of the discrete divergence theorem 2.1. This means that even in the presence of refinement, the sum of cell edge fluxes around a region of the refined grid structure is always equal to the sum of

the divergences in that region multiplied by cell areas. These two facts will be used to specify multigrid operators for the Poisson problem on refined grids and to show that solvability conditions for residual problems within the multigrid procedure are met.

It is important to note that, although the divergence operator defined above appears to be second-order accurate for coarse and fine grids, for coarse grid cells that border fine grid regions the true truncation error of the divergence contains a first-order term. This results from the fact that the coarse grid velocity used in the divergence is an average of fine grid values and, hence, contains an $O(h^2)$ term which is subsequently divided by h in the formula for the divergence. This implies that the operator DG that will be constructed from this divergence is also first order at these cells. Using higher order interpolation would eliminate this problem but at the expense of sacrificing the conservative property of the MAC-divergence discussed above. When solving a Poisson equation, however, it is well known that reducing the order of accuracy of the discrete operator at a set of points with lower dimension does not affect the global accuracy of the solution; hence the conservative form is used.

The derivation of the single grid MAC projection operator relied on the adjointness condition (15). To follow this derivation in the refined grid case, scalar and vector inner products must first be defined. The refined grid analogue for the scalar inner-product $\langle \phi, \psi \rangle_S$ is defined as the sum over all cells of the product of $\psi_{i,j}$ and $\phi_{i,j}$ multiplied by the area of the cell. In this definition, coarse grid cells in which finer grids exist are not included in the sum. This inner product is an exact approximation of $\int_\Omega \psi(\bar{x})\phi(\bar{x})\, d\bar{x}$ for functions ψ and ϕ that are constant on each cell.

To illustrate, consider $\langle \phi, D(U) \rangle_S$ for the small section of refined grid shown in Fig. 4. Let h be the grid spacing on the coarse grid. For this example

$$\langle \phi, D(U) \rangle_S = \phi^0_{1,1} \left(\frac{(u^1_{0,1} + u^1_{0,2})/2 - u^0_{0,1}}{h} + \frac{v^0_{1,1} - v^0_{1,0}}{h} \right) h^2 + \sum_{i,j=1}^{2} \phi^1_{i,j} \left(\frac{u^1_{i,j} - u^1_{i-1,j}}{h/2} + \frac{v^1_{i,j} - v^1_{i,j-1}}{h/2} \right) \frac{h^2}{4}. \tag{19}$$

The above definitions of the divergence and the scalar inner product are both natural and convenient in the sense that, on any individual grid, the definitions appear exactly as in the single grid case. One would hope that through the adjointness condition with an appropriate vector inner-product, a gradient can be defined that is identical to the staggered gradient in the single grid case away from coarse–fine interfaces. Indeed this is the case. The question of interest is: What form do the vector inner product and, hence, the refined grid gradient take at coarse-fine grid interfaces?

Unfortunately, a careful argument is presented in [30] showing that the refined grid gradient operator that is adjoint to the above divergence in the correct inner product is not second-order accurate at coarse–fine interfaces. In particular, returning to the grid in Fig. 4, the definition of the gradients at the interface using the adjointness condition is

$$G_1(\phi)^1_{0,j} = \frac{\phi^1_{1,j} - \phi^0_{1,1}}{3h/4}$$

which are only exact for functions linear in x and constant in y. The coarse grid value of the gradient at interfaces is defined as the average of the fine grid gradients as is the

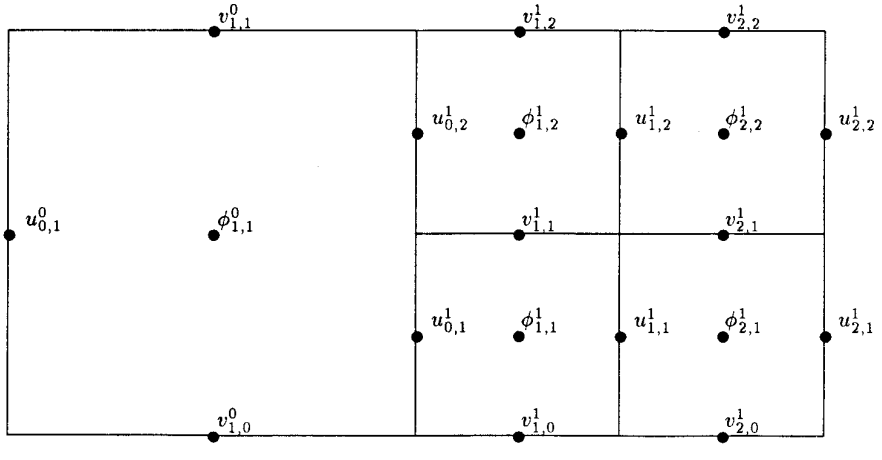

FIG. 4. Partial grid example with $r = 2$.

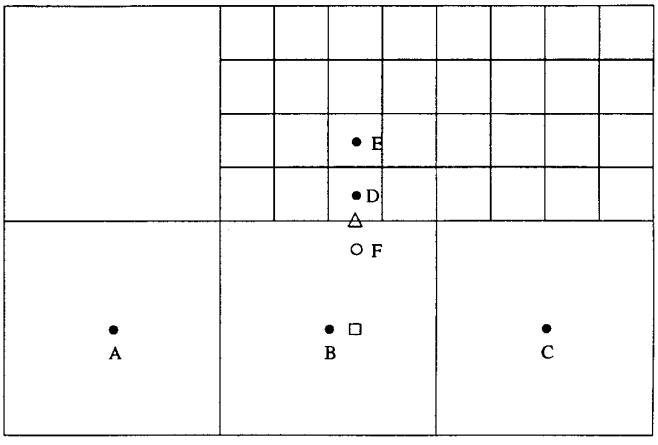

FIG. 5. Normal derivative or interpolation stencil with $r = 4$.

case for velocities. For the example shown in Fig. 4, averaging gives

$$G_1(\phi)^0_{1,1} = \frac{(\phi^1_{1,2} - \phi^0_{1,1})/3h/4 + (\phi^1_{1,1} - \phi^0_{1,1})/3h/4}{2} = \frac{(\phi^1_{1,2} + \phi^1_{1,1})/2 - \phi^0_{1,1}}{3h/4}. \tag{20}$$

When viewed as a coarse grid operator, the gradient at the interface is first-order accurate, but this is really no consolation.

The loss of accuracy for the gradient at grid interfaces is clearly unacceptable. Two remedies to the problem exist. One option is to alter the divergence at grid interfaces such that the adjoint of the divergence is a more accurate gradient. A divergence operator that depends on additional velocity values will define a gradient through the adjointness condition that is based on more than two points at each cell edge. In theory, this gradient could be second order. If such a stencil for the divergence exists, it should also be conservative so that the discrete divergence theorem holds. Unfortunately, the author's efforts to produce a conservative second-order divergence, the adjoint of which in some inner product is a second-order gradient, have thus far failed.

A second option, the one which is used here, is to simply ignore the adjointness condition at the coarse–fine grid interfaces. A second-order gradient is defined by using more coarse and fine grid values in the stencil, specifically three coarse and two fine. To avoid potential problems with wide stencils, the derivative operator used at grid interfaces depends on coarse grid cells that lie directly outside of child grids. These are in fact the coarse grid cells that contain boundary cells for the child grid.

For example, consider the section of a refined grid shown in Fig. 5. Let the value of some function ψ at each of the marked locations be denoted by the corresponding subscripts ψ_A, ψ_B, etc. Suppose that an approximation to the derivative of ψ in the vertical direction is desired at the child cell-edge location marked by the triangle in the figure. The formula for the derivative of ψ normal to the boundary at the point marked by the triangle is given by

$$\frac{\frac{1}{5}\psi_E + \frac{1}{3}\psi_D + \frac{1}{20}\psi_A - \frac{1}{2}\psi_B - \frac{1}{12}\psi_C,}{h} \tag{21}$$

where h is the cell size of the fine region. This is equivalent to defining the gradient in terms of the boundary value ψ_F,

$$\frac{\psi_D - \psi_F}{h}, \tag{22}$$

where the interpolation stencil for ψ_F is

$$\psi_F = -\tfrac{1}{5}\psi_E + \tfrac{2}{3}\psi_D - \tfrac{1}{20}\psi_A + \tfrac{1}{2}\psi_B + \tfrac{1}{12}\psi_C. \tag{23}$$

The form of the derivative of ψ is also equivalent to a second-order approximation which depends on the values of ψ_E and ψ_D and a third "virtual value" of ψ the location of which is denoted by a square in Fig. 5 and the value of which is interpolated from the coarse grid values ψ_A, ψ_B, and ψ_C.

Recall that the adjointness condition for the divergence and gradient operators enables one to prove that the projection is well-posed and norm reducing. Therefore, abandoning the adjointness condition at coarse–fine interfaces forces one to prove these facts in a different way if, in fact, they are true. It is possible to show directly that the projection defined above is well-posed for some simple refined grids, but a proof for arbitrary geometries has not been developed. In practice, however, solution of the projection Poisson problem has yet to fail.

One more note should be made. Consider the form of the coarse grid gradient $G_1(\phi)^0_{1,1}$ in Fig. 4. The value here cannot be written as

$$G_1(\phi)^0_{1,1} = \frac{A(\phi^1_{i,j}) - \phi^0_{1,1}}{h},$$

where A is some average depending only on fine grid values $\phi^1_{i,j}$. This implies that, at coarse grid cells neighboring refined regions, the solution of the projection Poisson problem will not satisfy the normal coarse grid Laplacian with coarse grid values derived from fine grid values where needed. The form of the Laplacian is always $L = DG$, where coarse grid G are consistently defined as the average of fine grid G at coarse–fine interfaces.

The Poisson problem associated with the MAC projection P,

$$DG(\phi)_{i,j} = D(U^*)_{i,j} \quad \text{with } F_k(G(\phi)) = F_k(U^*), \tag{24}$$

is solved using a multigrid routine described in Section 5. Since the right-hand side of Eq. (24) is in the form of a divergence, the discrete version of the solvability condition for the Neumann problem (see Eq. (6)) is guaranteed by the discrete form of the divergence theorem 2.1. This is a necessary, but not sufficient condition for Eq. (24) to have a solution.

4.3. *Temporal Discretization*

In recent methods for gas dynamics which employ grid refinement, solutions are updated using time refinement as well as spatial grid refinement (e.g., [10, 12]). The same ratio of grid spacing to time step is used for each level of grid refinement so that the time step on a parent grid is larger than the time step of its child by a factor of r. In practice, coarse grids are updated first, then finer grids are updated using coarse grid values interpolated in time and space as boundary values for finer grids. There are several reasons why temporal refinement is used in the compressible case. An obvious one is that there is a computational savings involved since coarse levels are updated less frequently than fine grids. Second, the advection schemes often used in these methods typically perform more poorly at low CFL numbers than when the CFL is equal to one, especially in the presence of shocks. Also, since the propagation speed in the compressible case is limited by the local speed of sound, one can ensure that interesting features will remain inside fine grid regions during a certain time interval.

The projection method on refined grids presented here employs no temporal refinement; velocities at all grid levels are updated every time step. For problems for which a majority of the grid points lie in the finest grid levels, computational savings associated with temporal refinement would be modest. More importantly, since the flow is incompressible, there is not a finite limit to the speed at which disturbances can propagate in the flow. There have been refined grid methods for incompressible flow presented recently that include temporal refinement [1, 35], but the question of how pressure effects can be adequately modeled locally has not been settled and will be left here for future work.

For this method, a computational grid is used in which the velocities and the pressure are represented at cell centers. At any given time step n, it is assumed that the values of the velocities at time n, denoted by U^n, and the time-centered pressure from the previous time step $p^{n-1/2}$ are known at the cell centers. In a single time step, updated velocities U^{n+1} and an updated pressure $p^{n+1/2}$ are computed by using an approximation of the second-order accurate temporal discretization of the projection form of the Euler equations:

$$\begin{aligned} \frac{U^{n+1} - U^n}{\Delta t} &= \mathbf{P}(-[(U \cdot \nabla)U]^{n+1/2} - \nabla p^{n+1/2}) \\ &= \mathbf{P}(-[(U \cdot \nabla)U]^{n+1/2}). \end{aligned} \tag{25}$$

Equation (25) implies

$$\nabla p^{n+1/2} = (I - \mathbf{P})([U \cdot \nabla)U]^{n+1/2}). \tag{26}$$

In the current method, Eq. (25) is approximated by first explicitly computing an approximation $A^{n+1/2}$ to the advective term $[(U \cdot \nabla)U]^{n+1/2}$. Details of this procedure are described in Section 4.4.

Next, a provisional value U^* is computed which is given by

$$\frac{U^* - U^n}{\Delta t} = -A^{n+1/2}. \tag{27}$$

The new velocity U^{n+1} is then computed by applying an approximate projection operator $\tilde{P}$ to U^*. The approximate projection consists of applying the MAC projection operator P to cell-edge values of U^* that are computed from the cell-centered values using fourth-order interpolation. A cell-centered gradient is then interpolated from the resulting cell-edge gradient and subtracted from the original values of U^*. Specifically, let $U^* = (u^*, v^*)$ and define the vector $\tilde{U} = (\tilde{u}, \tilde{v})$; then in grid interiors

$$\tilde{u}_{i,j} = \frac{-u^*_{i-1,j} + 9(u^*_{i,j} + u^*_{i+1,j}) - u^*_{i+2,j}}{16}$$

$$\tilde{v}_{i,j} = \frac{-v^*_{i,j-1} + 9(v^*_{i,j} + v^*_{i,j+1}) - v^*_{i,j+2}}{16}.$$

The values of $\tilde{U}$ at the edge of child grids are set by fourth-order interpolation using three values from the interior of child grids and a boundary value interpolated in the same manner as for the gradient operator. This higher-order interpolation makes the approximate projection operator significantly different from those in the references cited above. The author noted a modest reduction in the error of computed solutions to the problem studied in the first example of Section 6 when using the fourth-order interpolation, as opposed to simple averaging.

Applying the MAC projection operator to the edge values $\tilde{U}$ produces the Hodge decomposition

$$\tilde{U} = \tilde{U}^D + G(\phi),$$

where ϕ is the solution of

$$DG(\phi)_{i,j} = D(\tilde{U})_{i,j} \quad \text{with } F_k(G(\phi)) = F_k(\tilde{U}).$$

The boundary fluxes $F_k(\tilde{U})$ are set by third-order extrapolation of $\tilde{U}$ to the physical boundary.

The new velocity U^{n+1} is then computed by interpolating the cell-edge values of $G(\phi)$ back to the cell-center locations and subtracting these values from the original correction;

$$U^{n+1} = U^* - \tilde{G}(\phi),$$

where

$$\tilde{G}_1(\phi)_{i,j} = \frac{G_1(\phi)_{i,j} + G_1(\phi)_{i-1,j}}{2}$$

$$\tilde{G}_2(\phi)_{i,j} = \frac{G_2(\phi)_{i,j} + G_2(\phi)_{i,j-1}}{2}.$$

Using Eq. (26) the pressure is updated by

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

The time-centered advective term $A^{n+1/2}$ appearing in (27) is computed using an explicit second-order Godunov method. In [31], it is shown that the single grid version of the method used in this work is stable for the linearized Euler equations under the CFL restriction

$$\Delta t \le h/\max_{i,j}(|u_{i,j}|, |v_{i,j}|).$$

In practice Δt is recomputed each time step by

$$\Delta t = Ch/\max_{i,j}(|u_{i,j}|, |v_{i,j}|),$$

where C is the CFL number whose value is less than one.

4.4. *Evaluation of the Advective Term*

The term $A^{n+1/2}$ required in Eq. (27) is computed by an explicit method based on a second-order Godunov procedure. The basic idea of the procedure is to use a Taylor series expansion to calculate time-centered cell edge values of the velocities that can then be differenced to yield the advective term. As an example, the procedure for computing the value $U^{n+1/2,L}_{i+1/2,j}$ on the left side of the cell face centered at $(i + \frac{1}{2}, j, n + \frac{1}{2})$ is explained; the values on each side of the other faces are computed analogously.

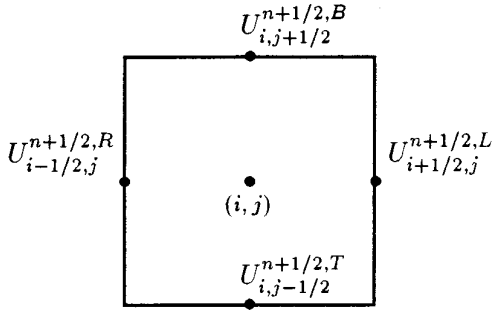

FIG. 6. Location of cell variables. The cell center is located at (i, j). Cell edge values are denoted using half-integer indices. For example, a cell edge value extrapolated from the left to the right cell face at time level $n + \frac{1}{2}$ is denoted by $U^{n+1/2,L}_{i+1/2,j}$.

The leading terms of the Taylor series are used to extrapolate the velocities to cell edges (see Fig. 6 for the location of cell-edge variables):

$$U^{n+1/2,L}_{i+1/2,j} = U^n_{i,j} + \frac{h}{2}(U^n_x)_{i,j} + \frac{\Delta t}{2}(U^n_t)_{i,j}. \tag{28}$$

By using the Euler equations, the temporal derivatives are replaced with spatial derivatives yielding

$$U^{n+1/2,L}_{i+1/2,j} = U^n_{i,j} + \left[\frac{h}{2} - \frac{\Delta t}{2}u^n_{i,j}\right](U^n_x)_{i,j} - \frac{\Delta t}{2}v^n_{i,j}(U^n_y)_{i,j} - \frac{\Delta t}{2}\nabla p^n_{i,j}. \tag{29}$$

This is approximated in three steps. First, the predicted edge values $\hat{U}^L$ of the velocity are computed which contain only the derivative terms from (29) normal to the cell edge,

$$\hat{U}^L_{i,j} = U^n_{i,j} + \left[\frac{h}{2} - \frac{\Delta t}{2}u^n_{i,j}\right](U^n_x)_{i,j}. \tag{30}$$

The term $(U^n_x)_{i,j}$ is approximated using fourth-order slopes which can be limited in cases where the velocity field is not smooth in order to avoid oscillatory behavior. When limiters are employed the monotonicity-preserving slopes presented in [7] are used. At the edge of child grids bordering a coarser grid, only one boundary cell is available, so second-order slopes are used.

A procedure similar to the one above extrapolates from the right in the cell centered at $i + 1, j$ to obtain $\hat{U}^R_{i+1/2,j}$

A method based on the Riemann problem for Burgers' equation is used to form one value from these two. Specifically,

$$\hat{U}_{i+1/2,j} = \begin{cases} \hat{U}^{\mathrm{L}}_{i+1/2,j} & \text{if } u^n_{i+1,j} > 0,\ u^n_{i,j} > 0, \\ \hat{U}^{\mathrm{R}}_{i+1/2,j} & \text{if } u^n_{i+1,j} < 0,\ u^n_{i,j} < 0, \\ (\hat{U}^{\mathrm{L}}_{i+1/2,j} + \hat{U}^{\mathrm{R}}_{i+1/2,j})/2 & \text{otherwise,} \end{cases} \quad (31)$$

is computed.

The extrapolation to grid edges of child grids is only done from cell centers inside the child grid. Therefore, at the edges of child grids, only one value of $\hat{U}^{\mathrm{L}}$ or $\hat{U}^{\mathrm{R}}$ is computed by the above procedure. The second value, which corresponds to extrapolation to cell edges at the grid boundary from cells just outside the grid, is interpolated from the appropriate parent grid values. Fourth-order polynomial interpolation is used for these edge values. Interpolation is unnecessary along child grid edges that are contiguous to another grid at the same level. In this case the second value needed is taken from the contiguous grid, as would be expected.

The second step in computing time-centered edge values is to approximate the transverse derivative term in (29) to yield edge values $\tilde{U}$. A difference of the computed value $\hat{U}$ is used for these terms (in this case $(U^n_y)_{i,j}$). Thus,

$$\tilde{U}^{\mathrm{L}}_{i+1/2,j} = \hat{U}^{\mathrm{L}}_{i+1/2,j} - \frac{\Delta t}{2} v^n_{i,j} \frac{(\hat{U}_{i,j+1/2} - \hat{U}_{i,j-1/2})}{h}.$$

Edge values $\tilde{U}_{i+1/2,j}$ are then determined from the left and right states using the formula analogous to (31).

To complete the approximation of time-centered cell-edge values, the pressure gradient term from Eq. (29) must be included. Since the gradient of the pressure is simply the term that enforces incompressibility, it is approximated by performing a MAC-projection on the computed values $\tilde{u}_{i+1/2,j}$ and $\tilde{v}_{i,j+1/2}$ [7]. This results in the divergence-free values $u^{n+1/2}_{i+1/2,j}$ and $v^{n+1/2}_{i,j+1/2}$.

An approximation to the gradient resulting from the MAC projection is also subtracted from the values of $\tilde{u}_{i,j+1/2}$ and $\tilde{v}_{i+1/2,j}$ to yield $u^{n+1/2}_{i,j+1/2}$ and $v^{n+1/2}_{i+1/2,j}$. Specifically, the four nearest values of the relevant gradient component are averaged and subtracted from the above values. Finally, the values $U^{n+1/2}$ are differenced to get the advection term

$$A^{n+1/2} = \frac{(u^{n+1/2}_{i+1/2,j} + u^{n+1/2}_{i-1/2,j})}{2} \frac{(U^{n+1/2}_{i+1/2,j} - U^{n+1/2}_{i-1/2,j})}{h} + \frac{(v^{n+1/2}_{i,j+1/2} + v^{n+1/2}_{i,j-1/2})}{2} \frac{(U^{n+1/2}_{i,j+1/2} - U^{n+1/2}_{i,j-1/2})}{h}.$$

4.5. *Comments on the Approximate Projection*

In the above, an approximate projection is described as a MAC projection on cell-edge velocities which are averaged from cell-centered values. Once cell-centered approximate projections are cast in the form of MAC projections, it is natural to ask what the best strategy is for interpolating cell-edge velocities from cell-centered velocities. One method that may at first seem appealing, because the approximate projection that results has the norm-reducing property of exact projections, contains the following three steps: First, interpolate cell-centered velocities to cell edges using centered interpolation operators. Second, apply the MAC projection to these cell-edge values to yield MAC divergence-free cell-edge velocities. And, lastly, interpolate the divergence-free edge velocities back to the cell centers with the same centered interpolation stencils as in the first step. Unfortunately, when the interpolants are simple averages, this method introduces a diffusive term into the discretized equation which resembles a one-dimensional Laplacian with a magnitude which scales like the grid size h. Even if higher-order interpolation is used, a similar diffusive term will result, making this form of an approximate projection undesirable for methods designed to model nearly inviscid flows. It is for this reason that the method used here interpolates the gradient part of the decomposition from edges to cell centers rather than the velocities themselves.

The treatment of the pressure term in the projection step of the current method is somewhat different than that of the original BCG method. In the original BCG method, the pressure gradient is defined at cell centers and a "pressure increment" form of the projection is used in which U^* is defined as

$$\frac{U^* - U^n}{\Delta t} = -A^{n+1/2} - \nabla p^{n-1/2}. \quad (32)$$

The new pressure gradient is then defined by

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

In the current method, the pressure itself is defined at cell centers and the pressure gradient at cell edges. Because the approximate projection $\tilde{P}$ used here is based on a MAC projection, it would be natural to incorporate a pressure increment form by adding the lagged pressure gradient term in Eq. (32) to the cell edge values $\tilde{U}$, i.e., by solving

$$U^{n+1} = \tilde{P}[\tilde{U} - \Delta t G(p^{n-1/2})].$$

It can easily be shown, however, that this is equivalent to not including the lagged pressure term and using $\Delta t p^{n-1/2}$

as the initial guess for ϕ in the multigrid routine used to solve Eq. (24).

Note that the form of U^* presented above is not required to have the same values of U^{n+1} at the computational boundary. If this were the case, the Poisson equation for ϕ associated with the projection would have homogeneous Neumann boundary conditions. This would imply that the normal derivative of the pressure at the boundary would be constant in time which is not true for all flows. When the boundary fluxes for the velocity are prescribed to be other than no-flow conditions (as in the example in Section 6), the projection operator is redefined so that it produces a velocity field with the correct boundary conditions. The only change to the above procedure this represents is to change the boundary conditions $F_k(G(\phi))$ in Eq. (24) above to

$$F_k(G(\phi)) = F_k(U^*) - F_k(U),$$

which is the discrete analog of Eq. (7).

5. A MULTIGRID METHOD FOR THE REFINED GRID MAC PROJECTION

In this section, an efficient procedure for solving the Poisson problem (24) associated with the MAC projection is presented. It is a logical extension to refined grids of the standard multigrid procedure for a single grid. The algorithm is similar in spirit to multilevel techniques proposed by Bai and Brandt [5] or to the *fast adaptive composite* methods of McCormick [28, 29].

In [3], Almgren presents a multigrid method for solving the Poisson problem associated with the method of local corrections on refined grids. This multigrid method is limited to the case where the refinement ratio between all grid levels is two. Similar methods developed for solving the Poisson problem associated with a discrete projection have been advanced by Almgren *et al.* [1] and Howell [23]. Two important features of the method presented here make it a significant improvement over the methods in [1, 23]. First, when applied to the Poisson problem associated with the MAC projection, the method computes the exact gradient that appears in the definition of the MAC projection (up to the error tolerance of the multigrid method). Consequently, the staggered velocity that results from the MAC projection has zero discrete divergence at all cells (again, up to the error tolerance). Second, the multigrid sweeps that make up the basis of the method progress directly from fine to coarse grids, using intermediate grids when the refinement factor is greater than two. Restriction and interpolation operators never need to be applied between grids with a refinement factor greater than two. Details of how intermediate grids are used and how this procedure differs from earlier multigrid algorithms are discussed in Section 5.5.

The basic idea in extending the multigrid procedure to refined grids with refinement ratios of two is to first perform relaxation on individual grids starting with the finest level. Then, instead of forming a residual problem for each fine grid, the fine grid residuals are averaged into the residual problem of their parent grids. Once these composite residual problems are solved on parent grids, the correction is then interpolated back to child grids. The specifics of how residuals and boundary values are averaged and interpolated are derived from the form of the divergence and gradient operators discussed in Section 4.2. The most significant fact about the form of the refined grid multigrid operators is that on any specific grid, they take nearly the same form as the standard operators for a single grid method. For completeness and to establish notation, the single grid operators will now be defined. (See [21] for an introduction to multigrid methods.) First, the form of the relaxation operator is presented. The problem to be solved is

$$DG\phi_{i,j} = f_{i,j}.$$

Given an initial guess to the solution ϕ^0, it can be shown that the following iteration procedure will converge to a solution of the above problem,

$$\phi^{k+1}_{i,j} = \phi^k_{i,j} - \lambda(DG(\phi^k_{i,j}) - f_{i,j}).$$

The usual value of $\lambda = -h^2/4$ is used which in the interior of a grid yields

$$\phi^{k+1}_{i,j} = \frac{\phi^k_{i+1,j} + \phi^k_{i-1,j} + \phi^k_{i,j+1} + \phi^k_{i,j-1} - h^2 f_{i,j}}{4}. \quad (33)$$

The form of relaxation at grid boundaries is modified to take into account the form of DG at the boundary. The values of ϕ^{k+1} are updated using a Gauss–Seidel relaxation sweep with red–black ordering. This procedure will be denoted by the operator notation

$$\phi^{k+1} = G^M(\phi^k, f, h).$$

Once the above relaxation is performed, a residual problem is formed and solved on a coarser grid. The computation of a residual will be described by the operator R^M defined by

$$\rho_{i,j} = R^M(\phi, f, h)_{i,j} = f_{i,j} - DG(\phi)_{i,j}.$$

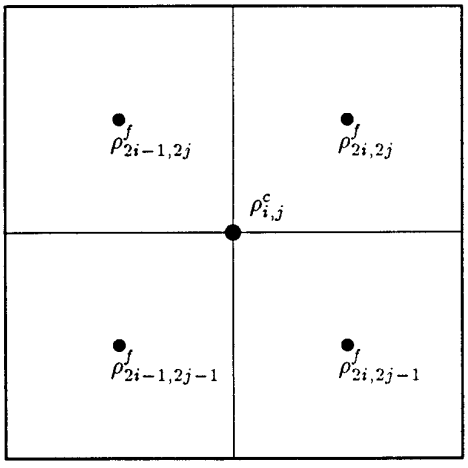

FIG. 7. Section of a grid showing multigrid refinement.

Again, the form of the residual at grid boundaries will depend on the form of the gradient there.

In order to represent residual problems on coarse grids and interpolate solutions of residual problems back to fine grids, averaging and interpolation operators must be defined. In the version of the method presented here, these operators will always be applied to grids for which the refinement ratio between them is two. Suppose $\rho^f_{i,j}$ is represented on the fine grid with grid size h and $\rho^c_{i,j}$ on a coarse grid with cell size $2h$. The generic arrangement of the data is shown in Fig. 7. Values of $\rho^c_{i,j}$ are computed from $\rho^f_{i,j}$ by the averaging operator A^M by

$$\rho^c_{i,j} = A^M(\rho^f)_{i,j} = \frac{\rho^f_{2i,2_j} + \rho^f_{2i-1,2j} + \rho^f_{2i,2j-1} + \rho^f_{2i-1,2j-1}}{4} \quad (34)$$

A bilinear interpolation operator is used for interpolating values of ξ^f from ξ^c, where ξ^f and ξ^c are residual problem solutions on fine and coarse grids, respectively. Each value of ξ^f depends on the four nearest values of ξ^c. For example,

$$\xi^f_{2i,2j} = \frac{9\xi^c_{i,j} + 3\xi^c_{i+1,j} + 3\xi^c_{i,j+1} + \xi^c_{i+1,j+1}}{16} \quad (35)$$

while

$$\xi^f_{2i-1,2j} = \frac{9\xi^c_{i,j} + 3\xi^c_{i-1,j} + 3\xi^c_{i,j+1} + \xi^c_{i-1,j+1}}{16}.$$

The notation

$$\xi^f = I^M(\xi^c)$$

will be used to denote interpolation in this manner.

Using this operator notation, the multigrid method can be described in a convenient recursive fashion. A multigrid V-cycle consists of recursively forming and solving residual problems on coarser grids until some coarseness criterion is reached. A description of a V-cycle in pseudo-code takes the form:

```
MGV(f, φ^k, h)
if (h < htol)                 /* Not coarsest level */
   φ^k = G^M(φ^k, f, h)       /* Relax initial guess */
   ρ^f = R^M(φ^k, f, h)       /* Calculate residual */
   ρ^c = A^M (ρ^f)            /* Average residual to next coarsest grid */
   ξ^c = MGV(ρ^c, 0, 2h)      /* Call MGV with residual problem */
   ξ^f = I^M(ξ^c)             /* Interpolate correction */
   φ^k = φ^k + ξ^f            /* Add correction to solution */
   φ^k = G^M(φ^k, f, h)       /* Relax corrected solution */
else                          /* Coarsest level */
   φ^k = G^M(φ^k, f, h)       /* Do one relaxation */
END MGV
```

The multigrid method for refined grids is easiest to describe when each grid level consists of a single grid, and the refinement factor between each level is two. The necessary adjustments for handling levels with multiple grids and for refinement factors greater than two are discussed below.

Given a refined grid consisting of levels *lmin* through *lmax* indexed by the integer l, suppose that an initial guess to the solution ϕ^l is available on each level and that the residuals ρ^l have also been computed on each level. The

following uses the operator notation from above to describe a single V-cycle on the entire refined grid. Details of the operators used in the V-cycle are included in the following subsections.

```
RGMGV(l)
if(l > lmin)                    /* Not coarsest level */
    ξ^l = G^M(ξ^l, ρ^l, h^l)      /* Relax residual problem */
    ρ̄^l = R^M(ξ^l, ρ^l, h^l)      /* Calculate residual of residual problem */
    ρ^{l-1} = A^{RM}(ρ̄^l, ρ^{l-1})  /* Average residual to parent grid */
    RGMGV (l − 1)               /* Call RGMGV for next level */
    ξ̃^l = I^{RM} (ξ^{l-1})        /* Interpolate coarse correction */
    ξ^l = ξ^l + ξ̃^l               /* Update correction */
    ξ^l = G^M (ξ^l, ρ^l, h^l)     /* Relax correction */
    φ^l = φ^l + ξ^l               /* Add correction to solution */
else                            /* Coarsest level */
    MGV (ρ^l, ξ^l, h^l)           /* Do one V-cycle on residual problem */
END RGMGV
```

In practice, the procedure outlined above is performed on the entire grid structure until some error criterion is met. For the problems presented here, the error criterion is that the absolute maximum of the residual on each grid is less than 10^{-9}. A single application of the above procedure on the entire grid structure typically reduces the norm of the residual by a factor of 5 to 10 which is approximately the same size reduction as for a V-cycle on a single grid problem. Some convergence statistics for the method are included in Section 6.

5.1. *Computing the Residuals*

The first step in the multigrid procedure described above is to compute the residual problem on each grid level. The residual operator R^M has the form

$$R^M(\phi) = f - L(\phi).$$

Recall, however, that the right-hand side for the projection Poisson problem is $D(U)$, and since $L = DG$ we have

$$R^M(\phi) = D(U) - D(G(\phi)) = D(U - G(\phi)).$$

The residual for a given ϕ is simply the divergence of $U - G(\phi)$. This is why the MAC projection produces a field with MAC-divergence that is the size of the multigrid tolerance.

Note that the form of the residual at grid boundaries or coarse–fine grid interfaces must be modified to take into account the form of the gradient at these points. In particular, at the boundaries of child grids, the form of the residual depends on the form of the gradient given in Section 4.2 (unless the child grid is contiguous to another grid with the same grid spacing). In the implementation, a row of boundary cells around child grids are given values in the same manner as discussed in Section 4.2. The value of parent grid cells just outside of child grids must also use the special form of the gradient across the coarse–fine interface when computing the residual.

5.2. *Averaging the Residuals*

During the refined grid V-cycles, residuals of residual problems are computed on child grids and then averaged into the residuals of parent grids. In the previous section it was shown that a residual is actually the divergence of an intermediate velocity. The same is true for residuals of residual problems. Since the divergence of a parent grid cell is equal to the average of the divergences of the refined cells it contains, the residual of a coarse cell is just the average of the residuals of the refined cells it contains.

A residual problem on a grid is simply

$$DG(\xi) = \rho = D(U) - DG(\phi).$$

The residual of a residual problem is then

$$\tilde{\rho} = \rho - DG(\xi) = D(U - G(\phi + \xi)).$$

In other words, the residual of a residual problem is simply the value that the residual of the original problem would take if ϕ was updated with the current solution of the residual problem ξ. This in turn is the same as the divergence of $U - G(\phi + \xi)$.

Once child grid residuals are constructed, they are averaged into the residual problem of their parent grid using the simple average discussed above. In addition, the parent grid residual in each of the cells surrounding the child grid must be adjusted to account for the fact that the gradient

across the coarse–fine interface now depends on $\phi + \xi$ on the child grid.

There is an important consequence of the fact that coarse grid residuals are always the average of fine grid residuals and that the right-hand side of residual problems are always divergences. The composite residual problem that is formed on the coarsest grid in the refined grid multigrid procedure is a Poisson problem with homogeneous Neumann boundary conditions. Hence, there is a solvability condition on the right-hand side of this problem, namely

$$\sum_{i,j} f_{i,j} = 0.$$

But since the right-hand side is in the form of a divergence, this solvability condition is automatically satisfied by Theorem 2.1. A different method of averaging residuals–even a more accurate method, could lead to a violation of the solvability condition for composite residual problems. Divergences or residual problems never need to be adjusted to satisfy the solvability constraints associated with the Neumann problem as is done in [23].

5.3. *Relaxation on Refined Grids*

On child grids, Gauss–Seidel relaxation operators must be altered near the boundaries to match the form of $L = DG$. In the implementation, a single row of boundary values is supplied for the relaxation operator, allowing it to take the same form as in Eq. (33). As in the computation of the residual, boundary values are determined using the interpolation stencil for the gradient operator. When a grid is contiguous to another grid with the same cell spacing, values of ϕ from the contiguous grid are used as would be expected. Since child grid boundary values depend in general on interior cells, the red and black subgrids are coupled at the boundary and, technically, a change in either subgrid will change the correct value in boundary cells. In the refined grid method, however, boundary values are only recomputed after a complete red–black sweep. This rule also applies in cases where grids at the same level are contiguous. Although boundary values of one grid in this case are interior cells of the other, boundary values are not recomputed until the relaxation on all grids on that level has been completed. This appears to cause no loss of efficiency for the method.

During the relaxation at the beginning of the V-cycle, parent grid values of the correction for any particular child grid are zero (they have yet to be improved by relaxation). This does not mean, however, that the child grid boundary values are homogeneous Dirichlet values. As said before, the boundary values are computed using the interpolation stencil for the gradient.

5.4. *Interpolating Corrections*

In the single grid method, the solution of residual problems on coarser grids is interpolated and added to a fine grid with a bilinear interpolation operator. The same operator is used for the refined grid method.

In the numerical implementation, boundary values for child grids also need to be set after a correction is interpolated and added to a child grid. This gives the illusion that boundary conditions are being interpolated from the residual problem on coarse grids, when in actuality the boundary values are being set so that the subsequent relaxation on the child grid can be implemented with the same stencil at the boundary as in the interior. As above, interpolation operators for boundary cells are derived from the form of the gradient operator at grid boundaries. Interior cells are interpolated first, then the boundary values are calculated by interpolation of the new interior cells and the parent grid values of the residual problem solution.

5.5. *Refinement Ratios Greater than Two*

When the refinement factor between grid levels is greater than two, solutions to residual problems may not be smooth enough to be accurately represented on the next coarsest grid level. Residual problems must be relaxed on grids of intermediate coarseness to ensure that the error in the residual problem is smooth enough to be averaged onto the next coarsest grid level. Multigrid methods similar to the one presented here appear in [1, 23] which, for refinement factors greater than two, replace the Gauss–Seidel relaxation procedure on individual grids with short multigrid V-cycle sweeps to ensure that residual problems have the necessary smoothness. Residual problems are still averaged between grid levels with refinement factors greater than two, and likewise, residual problem solutions are interpolated between grids with refinement factors greater than two.

In the method presented here, intermediate multigrid levels are created so that the refinement factor between all levels is two. These intermediate levels are used only in the multigrid method for relaxing residual problems and have no corresponding physical variables associated with them. For clarity, multigrid levels that do correspond to the grids containing physical variables will be referred to as physical levels. By consistently defining values of the divergence and gradient operators across coarse–fine grid interfaces on all grid levels, residual problems can be relaxed on intermediate grid levels that do not correspond to physical levels. The averaging and interpolating operators can also be defined to average residuals and interpolate corrections directly between intermediate and physical grids. The manner in which this is done also ensures that a completely solvable composite residual problem is computed for each grid level.

The grids in intermediate multigrid levels are constructed in the same manner as coarsened grids in the single grid multigrid method. For example, suppose a grid level corresponding to physical variables exists and that the refinement ratio between that level and the next coarsest physical level is eight. In this case, two intermediate multigrid levels would be needed. For each grid in the fine level, two grids would be created with the same physical dimensions but with grid spacing twice and four times larger than that of the fine grid.

In the multigrid algorithm when the refinement factor is two, the residual problem for a fine grid is averaged onto its parent grid, and the residual of the parent grid is altered at cells adjoining the fine region to reflect the update in the fine grid region. When the refinement factor is greater than two, the residual problem is averaged, instead, onto the next intermediate grid to form a coarse grid problem. This problem is then relaxed, and a new residual problem is computed which is subsequently averaged to the next level. This procedure continues until the next coarsest physical level is reached.

Whenever a residual problem is relaxed on a fine grid, the value of the residual at coarse grid cells surrounding the fine grid regions must be changed to correct for the fact that the value of the residual inside the fine grid region now depends on the gradient of $\phi + \xi$. Precisely the same procedure must be completed when a residual problem is relaxed on an intermediate grid: The value of the residual on the next coarsest physical level near the fine grid boundary must be updated to account for the fact that the residual problem on the intermediate grid is being calculated with an updated value of ϕ. This ensures that the right-hand side of a composite residual problem is always a divergence and, hence, the solvability condition for the discrete Poisson problem with Neumann boundary conditions is satisfied. In particular, if the coarsest grid level consists of a single grid, the composite residual problem that is formed on that grid is completely solvable.

6. CONVERGENCE RESULTS

To verify the second-order accuracy of the refined grid projection method, two convergence tests are presented. In the first test, a set of computations are run on a refined grid for a problem for which the exact solution is known. Second-order convergence to the exact solution is demonstrated for a refined region of the grid as the cell size of the entire grid is reduced. The initial conditions for this example are given on the periodic unit square by

$$\begin{aligned} u(x, y) &= 1 - 2\cos(2\pi x)\sin(2\pi y) \\ v(x, y) &= 1 + 2\sin(2\pi x)\cos(2\pi y). \end{aligned} \tag{36}$$

TABLE I

Convergence Rates for the Projection Method in a Refined Patch with Refinement Factors 2 and 4

n	Error, $r = 2$	Order in L_2	Error, $r = 4$	Order in L_2
16	3.969e-03		6.766e-03	
32	1.350e-03	1.56	2.182e-03	1.63
64	3.837e-04	1.82	5.912e-04	1.88
128	9.890e-05	1.96	1.492e-04	1.99

The exact solution for these initial conditions is

$$\begin{aligned} u(x, y, t) &= 1 - 2\cos(2\pi(x - t))\sin(2\pi(y - t)) \\ v(x, y, t) &= 1 + 2\sin(2\pi(x - t))\cos(2\pi(y - t)) \\ p(x, y, t) &= -\cos(4\pi(x - t)) - \cos(4\pi(y - t)). \end{aligned}$$

Four runs are performed on a grid with a single refined region consisting of the square defined by the points (0.25, 0.25) and (0.5, 0.5) and with grid spacing on the coarsest grid being $\frac{1}{16}$, $\frac{1}{32}$, $\frac{1}{64}$, and $\frac{1}{128}$, respectively. The length of the run for each case is 0.5, the CFL number is 0.75. The slope limiters in the computation of the advective derivatives are not utilized for these runs. The refinement ratio for the refined section of the grid remains constant over the four runs, so that as the grid spacing on the base grid is halved, the grid spacing in the refined region is also halved. For each run, the L_2 norm of the error in the u-velocity is computed for the refined region only. Table I contains the errors and convergence rates for this example.

From the data in the table, it is apparent that the error in the refined patch is converging to zero in the L_2 norm at nearly a second-order rate. However, a closer look at the tables reveals a seemingly paradoxical result. If one compares the magnitude of the errors for two runs with the same size base grid and refinement factors 2 and 4, it is evident that increasing the refinement factor and, hence, the number of points in the refined region, does not necessarily reduce the size of the errors in the refined region. For this example, in fact, the error for the $r = 4$ case is larger than for $r = 2$.

The explanation of this depends on two facts. First, since the exact solution is simply a translation in time of the initial conditions, the regions in the flow that are least accurately computed, namely where the velocities, and pressure gradient are least smooth, repeatedly cross through the refined grid region. The errors in the fine regions are therefore caused for the most part by errors in the surrounding coarse cells. As the coarse grid error converges to second order, so does the fine grid error.

The second reason stems from the form of the refined grid MAC divergence operator. For this problem, the dis-

TABLE II

Grid Cell Sizes for the Three Runs in the Vortex Capture Convergence Study

Level	Grid location	Run 1	Run 2	Run 3
0	(−64, −64) (64, 64)	2	2	2
1	(−16, −16) (16, 16)	1/2	1/2	1/2
2	(−6, −6) (6, 6)	1/4	1/8	1/16
3	(−2, −2) (2, 2)	1/16	1/32	1/64

crete divergence of the exact solution is zero at all cells except coarse grid cells that border fine grid regions, where averaging of fine grid velocities to coarse grid values breaks the symmetry of the coarse grid divergence. This introduces an error into the solution in the first and successive time steps. Furthermore, this error is larger for the $r = 4$ case than $r = 2$ because the truncation error of the averaging stencil for $r = 4$ is slightly larger than $r = 2$ (both are still second order). A higher order interpolant would remedy this problem, but as discussed in Section 4.2, using simple averaging of the velocities is necessary for other reasons and does not affect the global accuracy of the projection.

A much more convincing convergence example would show the solution in a refined region converging as the refinement ratio is increased while the grid spacing of the coarse grid region remains constant. The next example in this section is such a case.

The above example illustrates an important point for the would-be user of refined-grid methods for incompressible flow. Since regions of grid refinement use coarser grid values near boundaries for the computation of derivatives, one must be careful to arrange refined grids so that errors in coarse grid values surrounding them are as small as the intended errors in the refined regions. Also, since errors in the solutions can propagate with the fluid, one must be careful that coarse grid error does not flow into fine grid regions and affect the accuracy therein. Effective strategies for applying adaptive mesh refinement to incompressible flow would have to address both of these concerns.

In order to give a more convincing convergence example, convergence of refined grid solutions is illustrated while coarse grids are held to a fixed size. The problem used for this example is the merging of two regions of vorticity set in an unbounded domain. For this convergence test, three computations are done on a refined grid structure containing four grid levels, each consisting of a single grid. The coarsest two grids remain the same for the three runs, while the finest two levels are refined by increasing the refinement factor between levels one and two from 2 to 4 to 8. Table II gives the grid locations and cell sizes for each level for each of the three runs. The column labeled "Grid location" gives the coordinates of the lower left and upper right corner of the particular grid, while the columns labeled "Run 1" et cetera give the cell size for the grid.

The initial conditions for the problem are given by the superposition of two Gaussian patches of vorticity located in the plane at the points (0, 1) and (0, −1). Specifically, the initial vorticity distribution is given by

$$\omega(x, y) = 2\pi(e^{-2(x^2+(y-1)^2)} + e^{-2(x^2+(y+1)^2)}).$$

A vorticity contour plot of the initial conditions on a square grid extending from −2 to 2 in each dimension is shown in Fig. 9. Given these initial conditions, the two patches of vorticity will wind around each other to form a larger vortex patch centered at the origin. This phenomenon is sometimes referred to as vortex capture and plays a critical role in the transition of two-dimensional flows to quasi-equilibrium states. (See for example [32].)

Since the computational domain for the above example is, of course, finite, some approximation to the free-space boundary conditions must be made at the edges of the computational domain. The vorticity distribution for this problem has compact support (the Gaussian quickly decay to below machine precision), and the boundary conditions are set from the vorticity by considering the flow induced by a point vortex. Specifically, given a point vortex with strength Γ centered at the origin, the flow it induces is given by

$$\begin{aligned} u(x, y) &= -\Gamma \frac{y}{2\pi R^2} \\ v(x, y) &= \Gamma \frac{x}{2\pi R^2}. \end{aligned} \tag{37}$$

where R is the radial distance from the origin. If Γ is set equal to the integral of the initial vorticity (in this case $2\pi^2$), then as the distance from the origin is increased, the flow induced by the point vortex will converge to the flow induced by the Gaussian vortex patches. This is analogous to the gravitational effect of a cluster of stars such as a galaxy; at a great distance, it is indistinguishable from that of a single body with a mass equal to that of the galaxy. For the vortex capture problem the velocities given in Eq. (37) are used as the boundary conditions for the computational grid. By using a sequence of coarser grids centered at the origin, the computational domain is made to represent a large area around the patch so that applying the boundary conditions given by (37), instead of using an approximation to free-space boundary conditions affects the solution very little. The determining factor for how big the domain must be depends on the locality of the vorticity distribution and, hence, is problem dependent. For this convergence example, the box size is of little importance

and was found experimentally by increasing the domain size until further increases did not noticeably effect the solution.

For free-space problems, where the vorticity distribution is not localized about the origin, free-space boundary conditions can be explicitly computed using potential theory. A numerical method for approximating free-space boundary conditions in vortex systems similar to a method proposed by Anderson [4] is presented by Almgren in [3] and could be readily implemented on refined grids.

The initial conditions for the above problem are given in terms of the vorticity. For the projection method, however, initial conditions are required for the velocities and the pressure gradient and are computed by first solving

$$\Delta\psi_{i,j} = -\omega_{i,j}, \tag{38}$$

where ω is the initial vorticity and ψ is the initial stream function. Centered finite difference approximations to the derivative of ψ are then used to compute the velocities. The Poisson equation (38) is solved using the same multigrid procedure used for the projection. The imposed velocity boundary conditions discussed above serve as the Neumann boundary data for the vorticity-stream function Poisson problem. In order for the solvability condition to be met, it must be true that the sum of the imposed boundary velocity values equal the total integral of the vorticity in the domain. To enforce this, the values of the imposed velocities at the boundaries are initially adjusted after being set so that the solvability condition is met. The adjustment is typically on the order of 10^{-5} or less and is performed only during the computation of the initial conditions.

For each of the three grid structures described in Table II, solutions were computed to time $t = 4.0$ with CFL = 0.9. Convergence rates of the velocity were computed every 0.5 time units in the L_2 norm using the standard Richardson's procedure. The graph in Fig. 8 plots the computed convergence rate versus time. The convergence rate is initially almost exactly 2 but begins to drop off after time $t = 3$. This is presumably due to the fact that by $t = 4$ the solution contains a sharp gradient in the vorticity at the center of the two patches and hence is not smooth enough to show optimal convergence. Figure 9 shows the vorticity contours for the solution of the finest grid for the three runs at time 4.0. The coarsest run is shown in the top right figure, and the solution from the second finest and finest runs appear in the bottom left and bottom right pictures, respectively.

Multigrid convergence statistics were also collected for the three runs used in the above convergence study. For each run, the average number of multigrid iterations used in each multigrid solution during the run, as well as the average order of magnitude of the reduction in residual,

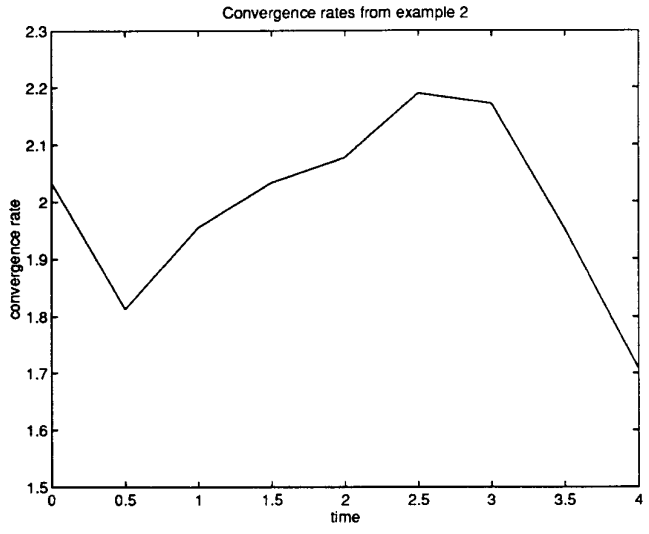

FIG. 8. Convergence rate for vortex capture problem.

was computed. The second of these numbers was found by taking the $\log_{10}$ of the ratio of the initial and final residuals. Table III contains the information for the case when only one Gauss–Seidel relaxation step was used in both the upward and downward sweeps of the multigrid V-cycle. It has been found experimentally that using two relaxations instead of one results in a more efficient solution in terms of overall computation time, and the corresponding numbers for this case are given in Table IV. The second table shows a typical reduction in residual of a factor of 10 for each iteration. For this problem, at least, the rate of convergence does not depend on the refinement ratios between grids, although for other problems a tendency for the method to perform slightly better for smaller refinement factors has been noted. The multigrid algorithm was used successfully in [30] on problems using six levels of refinement and 60 separate grids with similar convergence results.

In terms of CPU time, the various multigrid routines which are used for the two separate projections in the algorithm account for upwards of 60% of the total time for a typical run when computed on a single processor of a Cray YMP 8/64. As a specific example, the second run in the above convergence test, which used 33,792 grid points, took 114 s of CPU time, including initialization. There were 293 time steps in the run giving an average of about 11.5 μs per cell per time step. More than 65% of the total CPU time was spent in the basic multigrid routines.

7. CONCLUDING REMARKS

In this work, a projection method for solving the Euler equations on locally refined grids is presented. Numerical evidence is presented that shows the method is converging

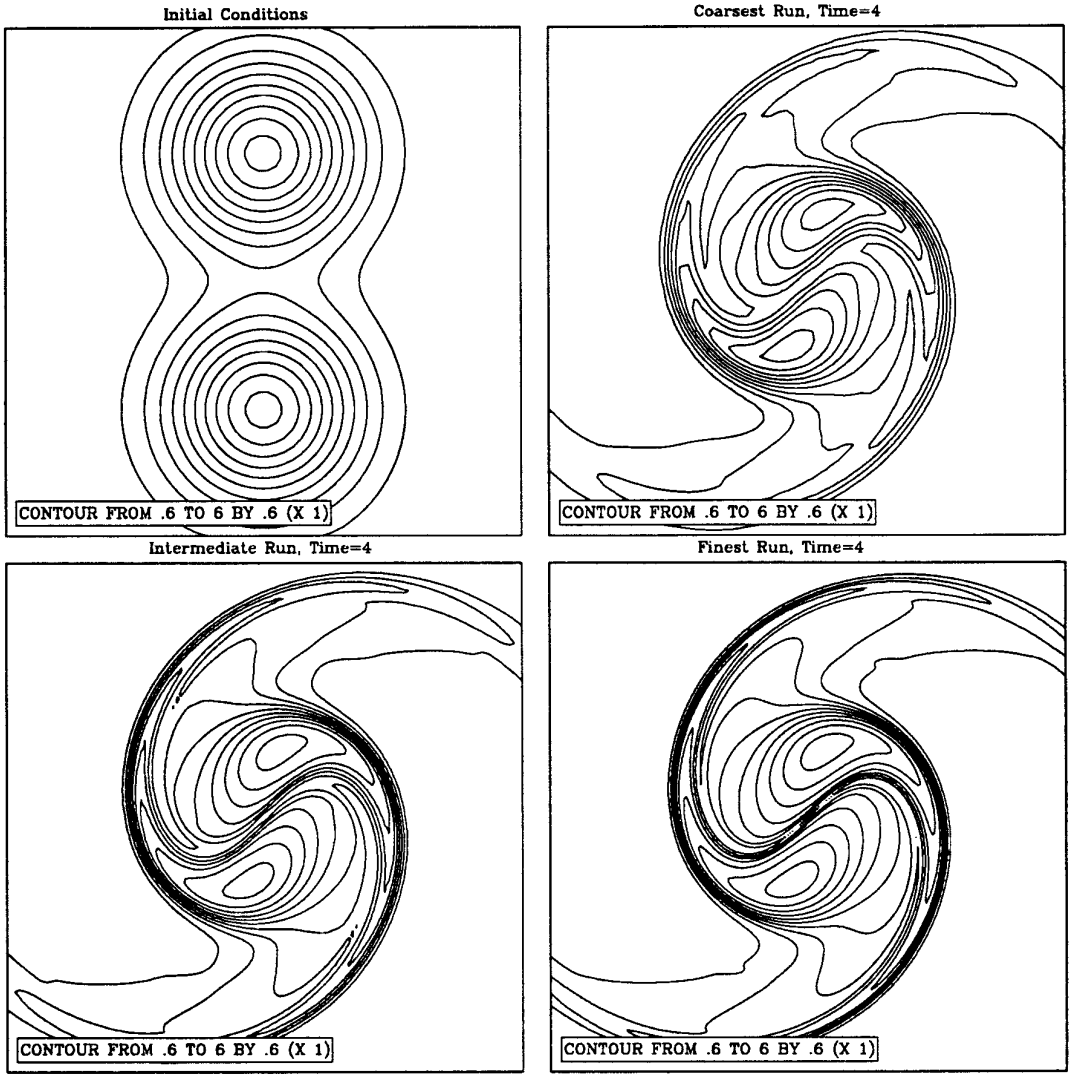

FIG. 9. Vorticity contours for the vortex capture convergence runs. The initial conditions appear in the top left corner. The physical domain shown represents the square given by $(-2, -2)$ and $(2, 2)$.

at a second-order rate relative to the cell size, even when cell size is held constant in other regions of the flow. Also, an efficient procedure for solving the projection on refined grids is presented.

It is relatively straightforward to apply the machinery developed for solving the Euler equations on refined grids to more complicated physical systems. For example, the multigrid method presented in this work is easily adapted to solve the implicit equations for viscous terms in the BCG method for the Navier–Stokes equations. Likewise, the advection of density or mass fraction of variable density flows can be easily implemented using the Godunov proce-

TABLE III

Multigrid Convergence Results when Using One Gauss–Seidel Relaxation

	Run 1	Run 2	Run 3
Mag. of reduction	8.04	7.73	7.34
Num. of iterations	12.35	11.26	11.52

TABLE IV

Multigrid Convergence Results when Using Two Gauss–Seidel Relaxations

	Run 1	Run 2	Run 3
Mag. of reduction	8.21	7.91	7.54
Num. of iterations	8.06	7.44	7.11

dure employed in the present method. In fact, this has already been done by the author and others. However, much research remains to determine the most effective strategies for such extensions. It is hoped that the analysis presented in this work will be an important contribution to the current research towards these goals.

In order to perform complicated engineering applications, it will be necessary to develop techniques for implementing refined grid methods on more complicated computational domains. Current areas of research to this end include methods using Cartesian grids, body-fitted grids, and overlapping composite grids. Each technique presents its own set of difficulties, and research into these methods is hampered by the significant time involved in their implementation.

A third area of research that needs to be more fully developed concerns the question of when and where to place refined grids during a calculation. For the examples used in this work, the position of refined regions of the grid were selected at the beginning of the run and did not change during the run. In many situations, a general knowledge of the evolution of the solution is known beforehand, and therefore, such manual positioning of refined regions is possible. For general problems with solutions that are not well understood, it is desirable to have a method for automatically determining where regions of grid refinement should be positioned. Adaptive mesh refinement (AMR) methods modeled after work by Berger [9] have been applied to systems of partial differential equations by Berger and Oliger [12], to the steady Euler equations by Berger and Jameson [11] and to the time-dependent equations by Berger and Colella [10]. Recent works have also included AMR in methods for the two- and three-dimensional incompressible fluid systems [1, 35].

The goal of developing a fully automatic algorithm for AMR in incompressible flow simulations will likely prove to be extremely difficult. In some instances, numerical viscosity caused by underresolution of a region of the flow may prohibit the appearance of some physically relevant phenomenon if refined grids are not employed before the phenomenon develops. On the other hand, calculations appearing in [13] illustrate that underresolution of a flow can cause seemingly physical, yet completely incorrect, large-scale structures to appear in flow calculations. It is not at this point clear how to detect the emergence of these phenomena before they occur. Fine tuning AMR methods using information from preliminary numerical experiments, as well as physical insight, may very well be necessary to effectively handle such situations.

ACKNOWLEDGMENTS

A majority of the work in this paper was completed as part of a thesis at the University of California under the supervision of Alexandre Chorin. I would like to thank Dr. Chorin for his abundance of helpful advice and encouragement. I would also like to thank Phil Colella for many useful conversations, David Brown, Jeff Saltzman, and Marsha Berger for insight and career support, Vinh Ton and DeAnne Musolf for help with proofreading, and Chip Sasso for his various contributions. This work was made possible by grants from the U.S. Department of Energy under Contracts DE-FG02-92ER25139 and DE-AC03-76SF00098. Calculations were performed at Los Alamos National Lab under Contract W-7405-ENG-36.

REFERENCES

1. A. S. Almgren, J. B. Bell, P. Colella, and L. H. Howell, ''An Adaptive Projection Method for the Incompressible Euler Equations,'' in *Proceedings Eleventh AIAA Computational Fluid Dynamics Conference*, p. 530 (AIAA, Washington, DC, June 1993).
2. A. S. Almgren, J. B. Bell, and W. G. Szymczak, *SIAM J. Sci. Comput.*, to appear.
3. A. S. Almgren, Ph.D. thesis, University of California, Berkeley, May 1991.
4. C. R. Anderson, ''Domain Decomposition Techniques and the Solution of Poisson's Equation in Infinite Domains,'' in *Proceedings, Second International Symposium on Domain Decomposition Methods, 1988*, p. 129.
5. D. Bai and A. Brandt, Technical report, The Weizmann Institute of Science, Rehovot 76100, Israel, 1984 (unpublished).
6. J. B. Bell, P. Colella, and H. M. Glaz, *J. Comput. Phys.* **85**(2) 257 (1989).
7. J. B. Bell, P. Colella, and L. H. Howell, ''An Efficient Second-Order Projection Method for Viscous Incompressible Flow,'' in *Proceedings, Tenth AIAA Computational Fluid Dynamics Conference, June 1991,* p. 360.
8. J. B. Bell and D. L. Marcus, *J. Comput. Phys.* **101**(1), (1992).
9. M. J. Berger, Ph.D. thesis, Stanford University, 1982.
10. M. J. Berger and P. Colella, *J. Comput. Phys.* **82,** 64 (1989).
11. M. J. Berger and A. Jameson, *AIAA J.* **23,** 561 (1985).
12. M. J. Berger and J. Oliger, *J. Comput. Phys.* **53,** 484 (1984).
13. D. L. Brown and M. Minion, *J. Comput. Phys.* **122,** 165 (1995).
14. A. J. Chorin, *Bull. Am. Math. Soc.* **73,** 928 (1967).
15. A. J. Chorin, *Stud. Numer. Anal.* **2,** 64 (1968).
16. A. J. Chorin, *Math. Comput.* **22,** 742 (1968).
17. A. J. Chorin, *Math. Comput.* **23,** 341 (1969).
18. P. Colella, *J. Comput. Phys.* **87,** 171 (1990).
19. C. A. J. Fletcher, *Computational Techniques for Fluid Dynamics, Vol. II* (Springer-Verlag, New York/Berlin, 1988).
20. P. M. Gresho, *Int. J. Numer. Methods Fluids* **11,** 587 (1990).
21. W. Hackbusch and U. Trottenberg (Eds.), *Multigrid Methods* (Springer-Verlag, New York, 1981).
22. F. H. Harlow and J. E. Welch, *Phys. Fluids* **8**(12), (1965).
23. L. H. Howell, ''A Multilevel Adaptive Projection Method for Unsteady Incompressible Flow,'' in *6th Copper Mountain Conference on Multigrid Methods, Apr. 1993.*
24. L. H. Howell and J. B. Bell, LLNL unclassified report UCRL-JC-105181, Lawrence Livermore National Laboratory, 1994 (unpublished).
25. F. John, *Partial Differential Equations* (Springer-Verlag, New York, 1982).
26. M. F. Lai, Ph.D. thesis, University of California, Berkeley, Sep. 1993 (unpublished).

27. M. F. Lai, J. Bell, and P. Colella, ''A Projection Method for Combustion in the Zero Mach Number Limit,'' in *Proceedings, Eleventh AIAA Computational Fluid Dynamics Conference, June 1993,* p. 776.
28. S. McCormick ''Fast Adaptive Composite Grid (fac) Methods: Theory for the Variational Case, in *Computing Supplementrum 5,* edited by K. Bohmer and H. J. Stetter (Springer-Verlag, New York/Berlin, 1984).
29. S. McCormick and J. Thomas, *Math. Comput.* **46,** 439 (1986).
30. M. Minion, Ph.D. thesis, University of California, Berkeley, May 1994.
31. M. Minion, *J. Comput. Phys.* **123,** 435 (1996).
32. D. Montgomery, W. Matthaeus, W. Stribling, D. Martinez, and S. Oughton, *Phys. Fluids A* **4,** 3 (1992).
33. R. Peyret and T. D. Taylor, *Computational Methods for Fluid Flow* (Springer-Verlag, New York, 1983).
34. J. C. Simo and F. Armero, *Comput. Methods Appl. Mech. Eng.* **111,** 111 (1994).
35. D. E. Stevens, Ph.D. thesis, University of Washington, 1994 (unpublished).