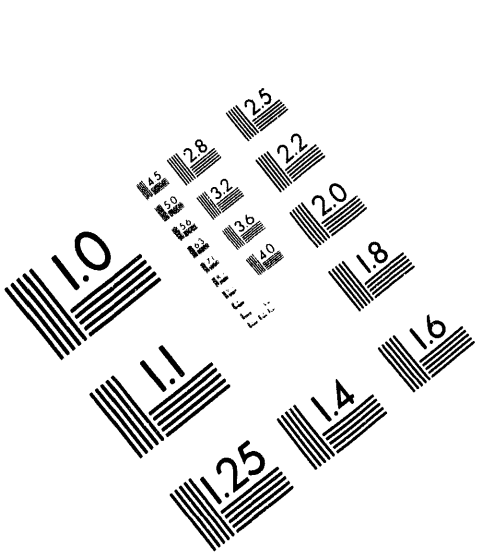
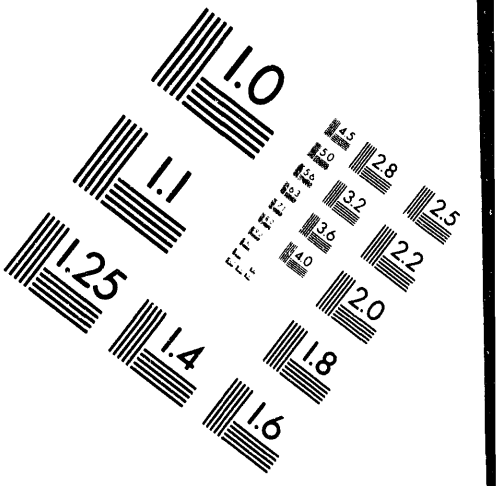
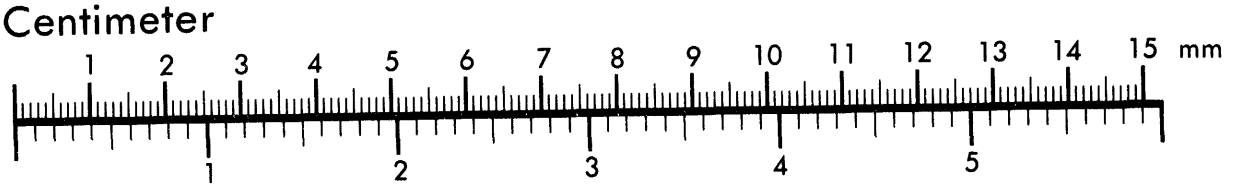


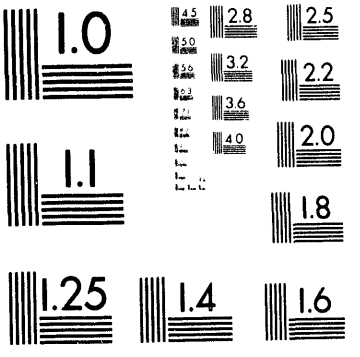
Association for Information and Image Management

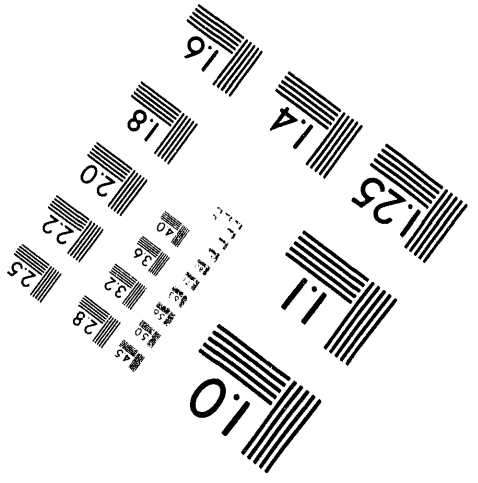
1100 Wayne Avenue, Suite 1100
Silver Spring, Maryland 20910

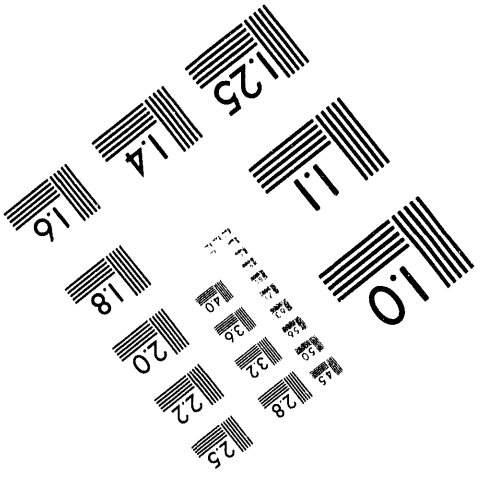
301/587-8202











MANUFACTURED TO AIIM STANDARDS
BY APPLIED IMAGE, INC.

1 of 2

LBL-35719

TWO METHODS FOR THE STUDY OF VORTEX PATCH EVOLUTION ON LOCALLY REFINED GRIDS [1]

Michael Lee Minion

Department of Mathematics
and
Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720

Ph.D. Thesis

May 1994

[1] This work was supported in part by the Applied Mathematical Sciences Subprogram of the Office of Energy Research, U.S. Department of Energy under Contract DE-AC03-76SF00098.

Abstract

Two Methods For The Study Of Vortex Patch Evolution On Locally Refined Grids

by

Michael Lee Minion

Two numerical methods for the solution of the two-dimensional Euler equations for incompressible flow on locally refined grids are presented. The first is a second order projection method adapted from the method of Bell, Colella, and Glaz. The second method is based on the vorticity-stream function form of the Euler equations and is designed to be free-stream preserving and conservative. Second order accuracy of both methods in time and space is established, and they are shown to agree on problems with a localized vorticity distribution.

The filamentation of a perturbed patch of circular vorticity and the merger of two smooth vortex patches are studied. It is speculated that for nearly stable patches of vorticity, an arbitrarily small amount of viscosity is sufficient to effectively eliminate vortex filaments from the evolving patch and that the filamentation process affects the evolution of such patches very little. Solutions of the vortex merger problem show that filamentation is responsible for the creation of large gradients in the vorticity which, in the presence of an arbitrarily small viscosity, will lead to vortex merger. It is speculated that a small viscosity in this problem does not substantially affect the transition of the flow to a statistical equilibrium solution.

The main contributions of this thesis concern the formulation and implementation of a projection for refined grids. A careful analysis of the adjointness relation between gradient and divergence operators for a refined grid MAC projection is presented, and a uniformly accurate, approximately stable 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

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Equations of Motion

The evolution of an incompressible fluid with constant density in a region of the plane is given by the following form of the Navier-Stokes 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

$$U_t = -uU_x - vU_y - \nabla p + \nu \Delta U \tag{1.1}$$

$$\nabla \cdot U = 0 \tag{1.2}$$

where ν is the viscosity of the fluid. The boundary conditions for these equations are

$$U(x, y, t) \cdot \hat{n} = 0 \text{ for } (x, y) \in \partial\Omega \tag{1.3}$$

where $\hat{n}$ is the outward unit normal vector on $\partial\Omega$, and if $\nu \neq 0$

$$U(x, y, t) = 0 \text{ for } (x, y) \in \partial\Omega. \tag{1.4}$$

Equation (1.3) is the constraint that no fluid can pass through the boundary $\partial\Omega$ and will be referred to as the no-flow condition. Equation (1.4) requires that in the presence of viscosity the flow at the wall will have zero velocity. This condition is referred to as the no-slip condition.

The term $-uU_x - vU_y$, which is often referred to as the advective term, gives the time rate of change of U at a point due to the motion of the fluid past that point. The diffusive term $\nu\Delta U$ models the viscous effects in the fluid which are the mechanism by which kinetic energy is removed from the flow. There is no net loss in kinetic energy in flows with $\nu = 0$. Flows of this sort are called inviscid or Euler flows.

When one describes flow that does not occur in a closed container, the boundary conditions (1.3) and (1.4) are augmented or replaced with conditions on the flow such as periodicity constraints or a specification of the flow at infinity. For the vortex patch problems presented in this paper, in-flow and out-flow conditions at the physical boundary are specified.

Proofs of the existence and regularity of solutions of the incompressible Navier-Stokes equations have been developed for two-dimensional flow under certain regularity assumptions. The theory for three dimensions, however, is only partially complete. Solutions are only known to remain smooth for short times. Ladyzhenskaya [38] provides a treatment of these two- and three-dimensional results.

1.2 Equation for the Vorticity

The vorticity ω of a flow is defined as the curl of the velocity field

$$\omega = \nabla \times U$$

which, for two-dimensional flow, is the scalar

$$\omega = v_x - u_y.$$

The evolution equation for vorticity for planar incompressible flow is derived by taking the curl of both sides of equation (1.1) and is given by

$$\omega_t = -u\omega_x - v\omega_y + \nu\Delta\omega \tag{1.5}$$

This equation can be rewritten in terms of the material derivative $\frac{D}{Dt}$ which gives the time rate of change of a quantity at a point that is moving along with the fluid. To define the time rate of change of some scalar ψ, denoted $\frac{D\psi}{Dt}$, suppose the point $(x(t), y(t))$ moves with the fluid velocity (u, v). In other words

$$\frac{dx}{dt} = u(x(t), y(t)) \tag{1.6}$$

$$\frac{dy}{dt} = v(x(t), y(t)). \tag{1.7}$$

The chain rule yields

$$\frac{D\psi}{Dt} = \frac{\partial}{\partial t}\psi(x(t), y(t)) = \psi_t + (\psi_x)\frac{dx}{dt} + (\psi_y)\frac{dy}{dt}$$

or

$$\frac{D\psi}{Dt} = \psi_t + u\psi_x + v\psi_y \tag{1.8}$$

By using equation (1.8), equation (1.5) can be written in terms of the material derivative

$$\frac{D\omega}{Dt} = \nu\Delta\omega,$$

or in the absence of viscosity

$$\frac{D\omega}{Dt} = 0. \tag{1.9}$$

Equation (1.9) indicates that in two-dimensional Euler flow, the value of the vorticity at a point being transported by the fluid does not change in time.

In general, the total amount of vorticity in Euler flow is conserved in the following sense. Suppose Ω is the domain of interest with boundary $\partial\Omega$. Using the fact that the velocity is divergence free, one can rewrite equation (1.5) in the following form:

$$\omega_t = -(u\omega)_x - (v\omega)_y + \nu\Delta\omega. \tag{1.10}$$

For inviscid flow, (1.10) can be rewritten as

$$\omega_t = -\nabla \cdot (\omega U). \tag{1.11}$$

and by integrating both sides of (1.11) and using the divergence theorem, one can show that

$$\frac{d}{dt}\int_\Omega \omega = -\int_{\partial\Omega} (\omega U) \cdot \hat{n}. \tag{1.12}$$

For domains for which the boundary integral on the right vanishes (and the divergence theorem holds), the total vorticity is constant in time. This is surely the case for flow in a closed container, flow in a doubly-periodic domain, or flow in an unbounded domain in which the vorticity has compact support.

1.3 Vortex Patch Dynamics

A consequence of equation (1.9) and the assumption that the velocity field is continuous at all times is that lines of constant vorticity in Euler flow will remain so in time without breaking. Consider then the evolution of a vorticity field in Euler flow which initially consists of a single closed region of the plane inside which the vorticity is some constant value α and outside of which the vorticity is a different constant value β. Since the value of the vorticity at a point moving in the fluid does not change, at any later time the value of the vorticity at any point in the plane must be either α or β. Also, because lines of equal vorticity will not break, the region in the plane in which the vorticity has value α will always be bounded by a closed contour. Only the shape of the contour will change. Such a closed region of constant vorticity will be referred to in general as a vortex patch.

More can be said about the evolution of vortex patches. Let Θ denote a closed curve in the plane and define the circulation Γ_Θ around Θ as

$$\Gamma_\Theta = \int_\Theta U \cdot ds. \tag{1.13}$$

It can be shown that for Euler flow, the circulation around a closed curve that is moving in the flow does not change in time (for a proof see Chorin and Marsden [24]). In other words, if one denotes by $\Theta(t)$ the curve moving in the fluid given by velocity U, then

$$\frac{d}{dt}\Gamma_{\Theta(t)} = \frac{d}{dt}\int_{\Theta(t)} U \cdot ds = 0. \tag{1.14}$$

Let the interior of the region bounded by $\Theta(t)$ be denoted by $\Sigma(t)$. When applied to equation (1.13), Stokes' theorem yields

$$\Gamma_\Theta = \int_\Sigma \omega dA,$$

and therefore by (1.14)

$$\frac{d}{dt}\int_{\Sigma(t)} \omega dA = 0.$$

If $\Sigma(t)$ is now taken as a region of constant vorticity, then it is apparent that the area of that patch is constant in time. In short, patches of constant vorticity in two-dimensional Euler flow may change shape as they move with the flow, but the value of the vorticity in the patch and the area of the patch will remain constant. More generally, given any piecewise-constant initial distribution of vorticity in which the plane is divided into n regions in which

the vorticity has value α_n, then at any later time the plane will still be divided into n disjoint regions within which the vorticity is α_n. Furthermore, the (possibly infinite) area of each of these regions will remain constant in time.

Consider now the special case of an elliptical patch of constant vorticity, centered at the origin, outside of which the vorticity is zero. It was first shown by Kirchoff in 1876 that such a vortex patch will maintain its elliptical shape for all time with the ellipse rotating at a uniform rate in the plane (see [17] for a history of elliptical patch results). Since a circle is a radially symmetric ellipse, a circular patch of vorticity is a stationary solution of the Euler equations in the plane. If the boundary of a circular vortex patch is slightly perturbed, the perturbation will rotate with the patch as the patch evolves.

In general, the boundary of a non-stationary vortex patch in Euler flow will eventually undergo a process referred to as filamentation wherein thin arm-like sections of the boundary will protrude away from the patch and quickly become elongated by the rotational flow outside of the patch. As these vortex filaments lengthen, they become thinner which is necessary to preserve the area of the vortex patch. In the absence of viscosity, there is no lower limit to the scale of vortical structures, and vortex filaments will continue to lengthen and become thinner as they repeatedly wrap around the core of the vortex patch.

In two dimensions, statistical theory has been successfully applied to the equations of vortex dynamics to form a theory of statistical equilibrium for the Euler equations. It is believed (and has been shown through numerical experiments) that most initial conditions will produce flows that in time will converge to certain equilibria that represent maximal entropy states. The vorticity and stream function of these equilibria satisfy the Joyce-Montgomery equation (see [23] page 84). It is believed that filamentation is the mechanism by which a vorticity field, given initially by an unsteady patch of constant vorticity, will eventually reach an equilibrium state. One can picture a sort of "vortex hairball" consisting of a small core of vorticity surrounded by increasingly complicated vortex filaments whose density in the plane approximates the maximal entropy distribution. The solutions of the vortex capture problem appearing in Section 6.5 support the theory that filamentation is the dominant mechanism in the approach to equilibrium states. Calculations presented by Montgomery et al. [41] exhibit an approach to equilibrium for a periodic vorticity distribution by way of repeated vortex capturing.

The rapid emergence of very small scales associated with filamentation makes the numerical modeling of vortex patch evolution difficult. For a patch of constant vorticity,

integral equations for the motion of the boundary of the patch can be developed. The evolution of the boundary of a constant vortex patch can thereby be computed by tracking the path of particles on the boundary as they move according to the these integral equations. A method based on this idea, called Contour Dynamics, was presented by Zabusky, Hughes, and Robert [52]. Since the boundary of a vortex patch becomes increasingly more complicated in time, Contour Dynamics methods must greatly increase the number of computational elements in use as a patch evolves. A "Contour Surgery" method to remove the smallest filaments of vortex patches and hence retard the rapid growth in the number of elements needed has been presented by Dritschel [27]. It is argued that removal of very small vortex filaments does not qualitatively change the nature of the flow because the filaments themselves contain very little energy. Applications of the Contour Surgery method to vortex patch filamentation problems appear in [28] and [29].

Other methods for modeling vortex patches use specialized versions of vortex methods. In [51], Wang presents an extension of a method by Buttke [18] for modeling patches of constant vorticity in which the interior of the patch is represented by a triangulation of a polygon which traces the boundary. The small lens-shaped regions between the actual patch boundary and the polygon used to describe it are approximated with vortex sheets. Equations are developed for the evolution of the triangular patches and the sheets which are then evolved by approximating these equations.

A major disadvantage of these two methods is that they are restricted to problems which have a piecewise-constant distribution of vorticity and zero viscosity (although a triangulated method for non-constant patches of vorticity has also been recently developed by Russo and Strain [46]). Finite difference methods can provide a way to compute the evolution of a smoothed vortex patch in viscous flows. In order to resolve small features such as filaments, the grid resolution of a finite difference calculation must be smaller than the size of the smallest scale wished to be resolved. For vortex patch filaments, which in time become increasingly thin and complicated, resolving these filaments can require an extremely fine mesh. In order to reduce the computational cost of uniformly fine grids, local grid refinement can be used to concentrate grid points near interesting features in the flow such as vortex patch filaments. The design and application of such refined grid methods is the main goal of this thesis.

It is generally accepted that most unsteady patches of constant vorticity in Euler flow will eventually filament. However, for smoothed vortex patches (those for which the

vorticity field is continuous), a clear understanding of when and if filamentation occurs is not available. The exact effect of viscosity on vortex filamentation is also unclear. Certainly the presence of moderate amounts of viscosity will limit the appearance of small scales in the fluid, and hence prohibit filamentation, but the effect of very small amounts of viscosity on the filamentation mechanism is not completely understood. Calculations appearing in Chapter 6 suggest that the filamentation exists in slightly viscous flows, and also that the filamentation process drives the creation of large gradients and small structures in the vorticity field which are eventually diffused by the viscosity.

In this thesis, two finite difference methods for locally refined grids are developed. One is based on a projection method and the other on a vorticity-stream function method. The projection method is described in Chapter 2 and the extension of this method to refined grids is outlined in Chapter 4. The vorticity-stream function method is presented in Chapter 3 while its extension to refined grids is described in Chapter 5. Numerical convergence studies and the presentation and discussion of numerical solutions of vortex patch and vortex capture problems are contained in Chapter 6.

Chapter 2

A Projection Method for a Single Grid

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

Theorem 2.0.1 *(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*

$$U = U^D + \nabla\phi \tag{2.1}$$

where

$$\nabla \cdot U^D = 0, \text{ in } \Omega \text{ and } U^D \cdot \hat{n} = 0 \text{ on } \partial\Omega.$$

Proof: Taking the divergence and normal component of both sides of (2.1) yields

$$\Delta\phi = \nabla \cdot U \tag{2.2}$$

and

$$\frac{\partial\phi}{\partial\hat{n}} = U \cdot \hat{n} \text{ on } \partial\Omega. \tag{2.3}$$

Equations (2.2) and (2.3) define a Neumann problem which is known to have a unique solution, up to an additive constant in ϕ, provided the solvability condition

$$\int_\Omega \nabla \cdot U = \int_{\partial\Omega} U \cdot \hat{n} \tag{2.4}$$

is satisfied [36]. Equation (2.4) is simply the divergence theorem, hence 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 an operator $\mathbf{P}$ by

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

Note also that

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

The operator $\mathbf{P}$ satisfies the definition of a projection,

$$\begin{aligned} \mathbf{P}^2 &= \mathbf{P}; \\ \mathbf{P}(\alpha U + V) &= \alpha\mathbf{P}(U) + \mathbf{P}(V). \end{aligned}$$

Note that the boundary condition on U^D in Theorem 2.0.1 is not restricted to $U^D \cdot \hat{n} = 0$. The projection operator $\mathbf{P}$ 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 application of the projection operator $\mathbf{P}$ to equations (1.1) yields the equivalent projection form of the Navier-Stokes equations

$$U_t = \mathbf{P}(-uU_x - vU_y + \nu\Delta U). \tag{2.5}$$

This has the desirable effect of eliminating the pressure term and the divergence constraint from equations (1.1). Although ΔU is divergence free, it may fail to satisfy the same boundary conditions as U^D, hence $\mathbf{P}$ generally commutes with the Laplacian operator Δ only in the absence of boundaries.

Projection methods in general attempt to approximate the Navier-Stokes equations by discretizing equation (2.5) directly. Chorin [19] [21] [20] [22] was the first to propose a numerical method based on this idea. In the original method, the velocities are updated 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. 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 which uses a higher order Godunov-type procedure to approximate the advective terms of the Navier-Stokes equations was introduced by Bell, Colella, and Glaz [8]. Extensions of this method have been successfully used in a number of different regimes, and a version of it serves as the basis for the projection method on refined grids presented in this work. Specifics of this single grid method are contained in Sections 2.4 and 2.5. For a survey of projection methods, see Gresho [31], Peyret and Taylor [42], or Simo [49].

What follows is a description of a projection method algorithm for a standard computational grid. There were several important criteria which this method was chosen to satisfy:

1. The method should be more than first order accurate in time and space. The method presented here is second order accurate.

2. The method should perform well when the time step is small compared to the grid size. When the method is implemented on refined grids, a uniform time step is used for coarse and fine grids. Therefore, the ratio of time step to grid size on the coarse grids becomes very small.

3. The method should be stable for any value of the viscosity. Although the value of under-resolved calculations can be debated, the method should not become unstable when it under-resolves the flow. If possible, such regions of under-resolution should be dealt with by grid refinement.

4. The implementation of the method on refined grids should be as straightforward as possible. The form of the projection in particular is motivated by issues pertaining to refined grids.

5. There should be a natural extension of the method to three-dimensional flow.

These items will be discussed in more detail in the following sections.

2.1 The Computational Grid

The starting point for any finite difference method is the discretization of the physical domain by a computational grid. For simplicity, the domain is restricted to be a rectangle divided into a regular $N_x \times N_y$ array of equally-sized square computational cells. The lower left corner of the computational grid is associated with the physical coordinate $(x_{\ell\ell}, y_{\ell\ell})$. The individual cells are denoted by the coordinates i, j with cell (i, j) having its center corresponding to the physical location $(x_i, y_j) = (x_{\ell\ell} + (i - 1/2)h, y_{\ell\ell} + (j - 1/2)h)$ where h is the width (and height) of each cell. This "cell-centered" form of discretization will be convenient when grid refinement is being used. Note that the cell centers do not lie on the boundary of the domain.

In the numerical implementation of the method, it is convenient to have additional cells located around the outside of the boundary of the domain. Physical variables in these cells can be given values so that the stencil of finite difference operators near grid 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 and not from assumptions about the form of solutions of the continuous equations at grid boundaries. These cells will be referred to as boundary cells or, slightly ambiguously, as boundary values.

2.2 Discrete Projections

Central to any numerical method based on the projection form of the Navier-Stokes equations (2.5) is the notion of a discrete projection operator. All projection methods must restrict the evolution of the solution to maintain an approximation to the divergence constraint. In the method presented here, as in the original method of Chorin, the solution is first updated without enforcing the divergence constraint. This approximate solution is then corrected using a discrete projection so that it is approximately divergence free. Until recently, most projection methods required that the solution be such that a discrete divergence of the velocity be exactly (or to machine precision) zero. Several recent methods weaken this constraint so that velocities must only approximate a divergence-free field. Methods that do not strictly constrain the solution to a space of discretely divergence-free fields will be referred to as approximate projection methods. This section contains a

discussion of issues relating to discrete projections in general. Details which relate these general issues to a specific exact projection and a certain approximate projection appear in subsequent sections.

Consider a computational domain Ω on which exists a velocity $U_{i,j} = (u_{i,j}, v_{i,j})$. Suppose also that there exists a discrete divergence operator D such that

$$D(U)_{i,j} \approx \nabla \cdot U(x_i, y_j).$$

One would like to have a version of the Hodge decomposition (2.0.1) on Ω so that U can be uniquely decomposed into the sum of a field whose discrete divergence is zero and some discrete gradient G of a scalar field ϕ. If this decomposition exists, a projection operator could be defined by

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

where

$$\begin{aligned} U_{i,j} &= U^D_{i,j} + G(\phi)_{i,j} \\ D(U^D)_{i,j} &= 0, \end{aligned}$$

with U^D satisfying some appropriate condition on the grid boundary.

One could follow the proof of theorem 2.0.1 and define

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

where ϕ is the solution of the linear system

$$DG(\phi)_{i,j} = D(U)_{i,j}, \tag{2.6}$$

and the components of $G(\phi)$ that correspond to the edges of the computational domain have the same value as the corresponding values of U at the boundary. In order for P to be well defined, a unique solution of this system must exist. One way to ensure this is to enforce an adjointness or orthogonality condition on the operators D and G. The basic idea is to make the discretely divergence-free fields orthogonal to gradients of scalars. Since both the set of discretely divergence-free fields and the set of all gradients are linear subspaces, then if they are also orthogonal subspaces, it would not be surprising that velocity fields could be represented by a direct sum of the two subspaces.

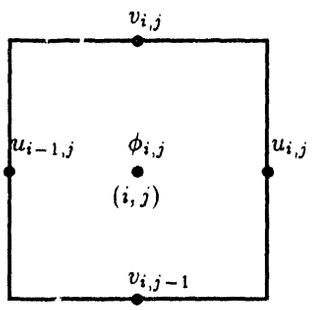


Figure 2.1: Cell variable locations for the projection method.

Chorin showed in [22], that for a particular choice of D and G, the relationship

$$\langle G(\phi), U\rangle_V = -\langle \phi, D(U)\rangle_S \tag{2.7}$$

held for a scalar inner product $\langle \phi, \psi\rangle_S$ and a vector inner product $\langle U, V\rangle_V$. If one applies equation (2.7) to a field U^D which satisfies $D(U^D)_{i,j} = 0$, it is immediately apparent that

$$\langle G(\phi), U^D\rangle_V = 0, \tag{2.8}$$

i.e., divergence free fields are orthogonal to gradients of scalars in this inner product. This in turn can be used to guarantee the existence and uniqueness (up to a constant in ϕ) of a solution of (2.6).

Since D and G are just linear operators acting on vectors, they have a natural matrix form. Note that if $D = -G^T$ as matrices and the inner products are defined as the vector dot product, equation (2.7) is satisfied trivially.

In the next section, an orthogonality condition will be used to prove a discrete form of the Hodge decomposition theorem for a specific D and G, and therefore provide a discrete projection operator P.

2.3 Staggered Grids and MAC Projections

In 1965, Harlow and Welch [32] presented a numerical method for two dimensional flow which uses a staggered computational grid. In this method, the computational domain is divided into a regular array of grid cells, and 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 figure 2.1.) As seen below, arranging the

data in this manner allows one to define convenient second-order divergence and gradient operators. Another advantage of this approach 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. For example, consider the term uv_x which appears in the Navier-Stokes equations. If one attempts to approximate this term at the top of cells where values of v are available, the derivative of v can be computed using standard finite differences, but there are no values of u immediately available to be used in the product. If the evaluation were instead done at a horizontal cell edge, values of v would no longer be horizontally centered around the evaluation point.

Given a computational domain Ω represented by a staggered computational grid, there is a set of cells with coordinates (i,j) which cover Ω. Let $\{(i,j)\}_\Omega$ denote the set of all the coordinates of cells covering Ω. The boundary $\partial\Omega$ consists of a set of cell edges ξ_k, which are indexed by a single integer k. For a velocity U defined on Ω, at each of the cell edges ξ_k only one of the velocity components u or v is defined. The cell-edge fluxes $F_k(U)$, defined for each ξ_k, are the discrete representation of $U \cdot \hat{n}$ on $\partial\Omega$. These fluxes are defined 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) and cell (i,j) is inside Ω, then $F_k(U) = v_{i,j}$. Likewise, $F_k(U) = -v_{i,j}$ if cell (i,j) lies outside Ω. It should be noted that values of $F_k(U)$ are not unknowns to be computed; they are in all cases specified by boundary conditions on the physical domain.

With the staggered data arrangement, a second-order approximation to the divergence at each cell has the simple, compact form

$$D(U)_{i,j} = \frac{u_{i,j} - u_{i-1,j}}{h} + \frac{v_{i,j} - v_{i,j-1}}{h}. \tag{2.9}$$

This form of the divergence is exactly the sum of the cell-edge fluxes $F_k(U)$ divided by h for a single computational cell. The form of D near a physical boundary utilizes the prescribed values of $F_k(U)$ at the physical boundary in place of the velocities at cell edges that correspond to the boundary.

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

Theorem 2.3.1 *(Discrete Divergence Theorem) Given a simply connected 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 2.1, if D and F_k are defined as above, then

$$\sum_{\{i,j\}_\Omega} D(U)_{i,j}h^2 = \sum_k F_k(U)h. \tag{2.10}$$

This theorem will be used in Chapter 5 to provide a solvability condition for the multigrid procedure on refined grids as well as to prove a discrete version of the Hodge decomposition for staggered grids.

A second-order approximation to the gradient of a cell-centered scalar field $\phi_{i,j}$ is given by

$$G(\phi)_{i,j} = (\frac{\phi_{i+1,j} - \phi_{i,j}}{h}, \frac{\phi_{i,j+1} - \phi_{i,j}}{h}). \tag{2.11}$$

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.

In a numerical implementation, artificial cells around the boundary of a grid can be given values so that if $G(\phi)$ is computed in the usual way at a boundary, the resulting value will be consistent with the specified value $F_k(G\phi)$; however, these artificial boundary values have no physical meaning. In particular, a value of $F_k(G\phi)$ is not computable by a difference equation across the grid boundary; $F_k(G\phi)$ must be specified at physical boundaries.

Consider again the domain Ω composed of the cells $\{(i,j)\}_\Omega$ and bounded by a set of edges ξ_k. Each ξ_k is the edge of exactly one cell $(i,j)_k \in \{(i,j)\}_\Omega$ with the possibility that some ξ_k may correspond to the same value of $(i,j)_k$ at corners of Ω. Let the set of indices in $\{(i,j)\}_\Omega$, which represent cells whose right cell edges lie inside of Ω, be denoted $\{(i,j)\}_u$, and likewise $\{(i,j)\}_v$ be the set of indices of cells whose top cell edges lie inside of Ω. (The edges ξ_k are not considered to lie inside of Ω.) For discrete velocities $W = (w_1, w_2)$ and $V = (v_1, v_2)$ with cell edge fluxes $F_k(W)$ and $F_k(V)$ at the boundary of Ω, a vector inner product on Ω is given by

$$\langle W, V\rangle_V = \sum_{\{i,j\}_u} w_{1i,j}v_{1i,j}h^2 + \sum_{\{i,j\}_v} w_{2i,j}v_{2i,j}h^2 + \sum_k F_k(W)F_k(V)\frac{h^2}{2}.$$

Likewise, for scalars ϕ and ψ one can define

$$\langle \phi, \psi\rangle_S = \sum_{\{i,j\}_\Omega} \phi_{i,j}\psi_{i,j}h^2.$$

These inner products are exactly the rectangle rule approximation for the integrals

$$\int_\Omega w_1(\bar{x})v_1(\bar{x}) + w_2(\bar{x})v_2(\bar{x})d\bar{x}$$

and

$$\int_\Omega \phi(\bar{x})\psi(\bar{x})d\bar{x}$$

where W, V, ϕ, and ψ are continuous functions.

Vector and scalar norms $\|\cdot\|_V$ and $\|\cdot\|_S$ are defined from these inner product in the usual manner, e.g.

$$\|W\|_V^2 = \langle W, W\rangle_V.$$

Given the above definitions of inner products, the following equation can be shown to be true simply by writing out the summations and canceling terms:

$$\langle G(\phi), U\rangle_V + \langle \phi, D(U)\rangle_S = F(U, \phi) \tag{2.12}$$

where

$$F(U,\phi) = \sum_k [\phi_{(i,j)_k} + \frac{h}{2}F_k(G\phi)]F_k(U)h \tag{2.13}$$

Since $F_k(G\phi)$ is an approximation to a the normal derivative of ϕ at the boundary cell edge ξ_k, the terms $\phi_{(i,j)_k} + \frac{h}{2}F_k(G\phi)$ are first-order approximations to the value of ϕ at the edge ξ_k. Therefore, equation 2.12 is the discrete analog of the identity for continuous functions

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

A consequence of equation (2.12) is that for a field U which has zero flux at the boundary, (i.e. $F_k(U) = 0$ for all k), equation 2.7 holds:

$$\langle G(\phi), U\rangle_V = -\langle \phi, D(U)\rangle_S \tag{2.14}$$

which in turn implies that for a field U^D such that $D(U^D)_{i,j} = 0$ for all i and j and $F_k(U^D) = 0$ for all k,

$$\langle G(\phi), U^D\rangle_V = 0. \tag{2.15}$$

A proof of a staggered grid version of the Hodge theorem is now presented. The proof follows the one given by Chorin in [22].

Theorem 2.3.2 *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 $F_k(U) = g_k$, and discrete divergence and gradient operators D and G as defined by equations (2.9) and (2.11), U can be uniquely decomposed into the form*

$$U_{i,j} = U^D_{i,j} + G(\phi)_{i,j} \tag{2.16}$$

where

$$D(U^D)_{i,j} = 0 \text{ and } F_k(U^D) = 0. \tag{2.17}$$

To prove the theorem, we view equations (2.16) and (2.17) as a finite dimensional linear system and show by the Fredholm alternative that this system in non-singular. Given a discrete velocity $U = (u, v)$ with boundary fluxes given by $F_k(U)$, and a discrete scalar field ϕ, define a single vector X whose components are made up of all the components of u, v and ϕ,

$$X = \begin{pmatrix} u \\ v \\ \phi \end{pmatrix}.$$

We can actually set one of the $\phi_{i,j}$ variables to zero and omit it from X since ϕ need only be determined up to a constant. Define a linear operator A by

$$AX = \begin{pmatrix} u + G_1(\phi) \\ v + G_2(\phi) \\ D(U) \end{pmatrix}. \tag{2.18}$$

As was the case with ϕ, the last entry of $D(U)$ can be omitted from the vector on the right-hand side of equation (2.18). Omitting this term is possible because one can derive its value from $F_k(U)$ and the remaining values of $D(U)$ by the discrete divergence theorem (2.3.1). If the vector B is defined as

$$B = \begin{pmatrix} u \\ v \\ \mathbf{0} \end{pmatrix}.$$

where the $\mathbf{0}$ term is a vector of zeroes of the appropriate length, then a solution of equations (2.16) and (2.17) can be written as the solution X^D of the equation

$$AX^D = B \text{ with } X^D = \begin{pmatrix} u^D \\ v^D \\ \phi \end{pmatrix}. \tag{2.19}$$

Implicit in this equation is the fact that $F_k(G\phi) = F_k(U)$.

By the Fredholm alternative, equation (2.19) has a unique solution if and only if $AX = \mathbf{0}$ implies $X = \mathbf{0}$. But if $U = U^D + G(\phi) = \mathbf{0}$, then

$$0 = \langle U^D + G(\phi), U^D + G(\phi)\rangle_V = \|U^D\|_V^2 + 2\langle U^D, G(\phi)\rangle_V + \|G(\phi)\|_V^2.$$

Since $D(U^D) = 0$ and $F_k(U^D) = 0$, by equation (2.15)

$$\langle U^D, G(\phi)\rangle_V = 0,$$

which implies

$$\|U^D\|_V^2 = \|G(\phi)\|_V^2 = 0,$$

which proves the theorem. •

As in the continuous case, since U^D is well-defined, we can define the projection operator P

$$\begin{aligned} P(U)_{i,j} &= U^D_{i,j}, \\ (I - P)(U)_{i,j} &= G(\phi)_{i,j}. \end{aligned}$$

One additional consequence of equation (2.15) is that the projection P is norm reducing. This can be seen from the identity

$$\|U\|_V^2 = \|U^D\|_V^2 + \|G(\phi)\|_V^2,$$

from which it follows that

$$\|P(U)\|_V^2 \leq \|U\|_V^2.$$

The method for implementing P also mirrors the continuous case; ϕ is given by the solution of the equation

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

with the boundary conditions

$$F_k(G\phi) = F_k(U),$$

where the Laplacian operator DG takes the standard five-point, second-order accurate form

$$DG(\phi)_{i,j} = \frac{-4\phi_{i,j} + \phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1}}{h^2}$$

away from grid boundaries. A projection defined with these operators will be referred to as a MAC projection, and the divergence D defined on staggered grids as in equation (2.9) will be referred to as the MAC divergence. In the following discussion, the five-point form that DG assumes away from grid boundaries will be referred to as L^5, i.e.

$$L^5(\phi)_{i,j} = \frac{-4\phi_{i,j} + \phi_{i+1,j} + \phi_{i-1,j} + \phi_{i,j+1} + \phi_{i,j-1}}{h^2}$$

2.4 Approximate Projections and MAC Projections

In 1989, Bell, Colella and Glaz (BCG) [8] introduced a projection method that utilizes higher order Godunov methods for approximating the advective terms in the Navier-Stokes equations. These Godunov methods, developed by Colella for gas dynamics [25], robustly treat regions of large gradients or discontinuities without introducing unphysical oscillations into the numerical solutions. When these methods are coupled with the projection method, the result is a second-order method that has no inherent cell-Reynolds number restriction, i.e. the method produces stable results even when the grid refinement is much larger than the smallest scale of the actual physical flow being modeled (for a study of under-resolved results, see Brown and Minion [16] and E and Shu [30]).

Since it was introduced, the BCG method has been refined and extended by Bell and Marcus to variable density flows [10], and by Lai et al. to reactive flows [40] [39]. Howell [35] 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 used in these projection methods require a cell-centered discretization of physical space in which both velocity components u and v are represented at the center of computational cells. When velocities are located at cell centers, it is natural to define a discrete divergence by the centered difference approximation

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

To define a projection from this divergence, a cell-centered gradient operator must also be defined, and it is again natural to use centered differences which yield

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

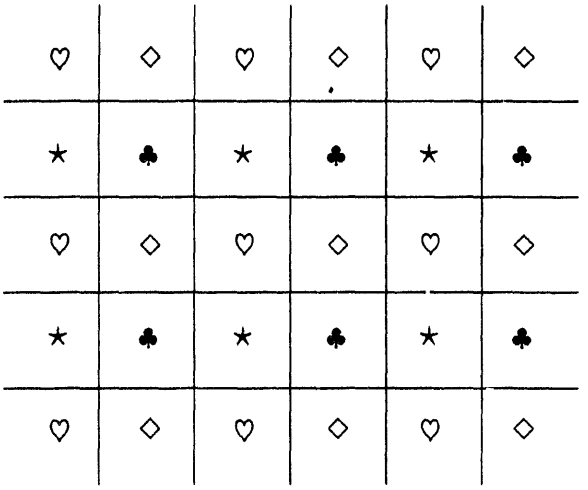


Figure 2.2: Decoupled stencil for $\tilde{L}^5$.

The composition of these divergence and gradient operators produce an approximation to the Laplacian given by

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

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 an extra grid cell. This causes the stencil for $\tilde{L}^5$ to decouple onto four distinct subgrids which are represented in figure 2.2 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. Therefore, when designing a projection method using the centered divergence operator and a projection based on $\tilde{L}^5$, several computational issues must be faced.

First, 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 respected in the multigrid operators. (See [9] 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 [35].) Recall that the Laplacian operator for the MAC projection presented in the previous section has a compact five-point stencil. It is exactly the desire to use the compact Laplacian that motivates the formulation of a projection based on a MAC projection in this thesis.

A second complication stems from the form of the centered difference divergence D^o. A cell-centered velocity field in which both the u and v components of the velocity consist of the oscillating mode mentioned above is divergence free with respect to the divergence operator D^o. A consequence of this fact is that a projection operator base on D^o will not remove such a mode from the flow if it is introduced. In the method for reacting flows appearing in [39], a "filtering" procedure is employed every time step to remove oscillations from the velocity fields. The rationale for the filtering step is that the oscillating modes, although they are divergence free, are not physical and should be removed. No filters of any sort are used for the projection method presented in this thesis.

In order to avoid using $\tilde{L}^5$ while retaining the cell-centered Godunov methodology, approximate projection methods have recently been developed which replace $\tilde{L}^5$ with a more convenient form in the projection step. The velocity field produced by an approximate projection does not have a zero discrete divergence, but instead it is expected to approximate a divergence-free field to some order of accuracy. A consequence of using approximate projections is that one can no longer rely on the adjointness of the divergence and gradient to prove that the projection is well posed or norm reducing. This of course does not imply that approximate projections are not well posed or norm reducing, but forces one to prove in a different way that the approximate projection has these properties, if in fact it does.

Almgren et al. [2] present an approximate projection method which is based on a Laplacian derived from finite element methodology and can result in a Laplacian with a compact nine-point or the standard five-point form. This single grid method serves as the basis for a method on locally refined grids that appears in [1].

Another approximate projection is presented by Lai [39], for a projection method for reacting flows. This approximate projection is a generalized projection for use with flows that have a non-constant density. If one considers the form of the approximate projection in [39] when applied to a flow with constant density, it can be seen that the projection operator is identical to the one which results from the operators D^o and G^o except that the Laplacian $\tilde{L}^5$ is replaced by L^5 in the discrete Poisson problem. Denoting cell-centered

values by U^c, the approximate projection $\tilde{P}$ for constant density flows in [39] becomes

$$\tilde{P}(U^c)_{i,j} = U^c_{i,j} - G^o(\phi)_{i,j} \tag{2.20}$$

where ϕ satisfies

$$L^5(\phi)_{i,j} = D^o(U^c)_{i,j}. \tag{2.21}$$

It is quite simple to formulate this approximate projection in terms of a MAC projection. If one interpolates cell edge values of the velocities $\tilde{U}_{i,j}$ from the cell-centered values $U^c_{i,j}$ by simple averaging

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

then it is easy to check that

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

Hence the Poisson problem associated with the approximate projection in [39] is exactly the one that results from performing a MAC projection on the $\tilde{U}_{i,j}$ values. It is also trivial to see that $G^o(\phi)_{i,j}$ is the simple average of cell-edge gradient components of the gradient G which results from the MAC projection on the $\tilde{U}_{i,j}$ values. Therefore, the approximate projection (2.20) is equivalent to averaging cell-centered velocities to cell edges, performing a MAC projection on those edge values, and subtracting the average of the resulting gradients from the initial velocities. Specifically,

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

where

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

and where ϕ is the solution of

$$\begin{aligned} DG(\phi)_{i,j} &= D(\tilde{U})_{i,j} \\ \tilde{u}_{i,j} &= (u^c_{i+1,j} + u^c_{i,j})/2 \\ \tilde{v}_{i,j} &= (v^c_{i,j+1} + v^c_{i,j})/2. \end{aligned}$$

A straightforward generalization of the arguments presented above can be used to present the variable density version of the approximate projection in [39] in terms of a MAC

projection. However, the problems presented in this paper will be for constant density flows only.

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 same norm reducing property of exact projection 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 divergence-free cell-edge velocities. And lastly, interpolate the divergence-free edge velocities back to 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.

Projection methods for the Navier-Stokes equations

$$U_t = -uU_x - vU_y - \nabla p + \nu \Delta U \tag{2.22}$$

$$\nabla \cdot U = 0 \tag{2.23}$$

approximate the effect of the divergence constraint by restricting the solution to the space of discretely divergence-free vector fields. In many methods this is accomplished by applying a discrete projection operator to the velocity field after each time step. For the approximate projection method discussed in the next section, the velocities are not constrained to a space of discretely divergence-free fields. Instead, an approximate projection based on a MAC projection similar to the one just described is applied after each time step to approximately enforce the divergence constraint.

2.5 A Second Order Godunov Projection Method

The projection method employed in this paper is a variation of the one appearing in Bell, Colella, and Howell [9]. It is very similar to the constant density version of the

method presented by Lai in [39]. The details of the single grid version of the method used in this thesis are presented here. For more complete details of the method as well as further motivation behind the details see [8] or [9].

2.5.1 Temporal Discretization

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 gradient from the previous time step $\nabla p^{n-1/2}$ are known at cell centers. In a single time step, updated velocities U^{n+1} and an updated pressure gradient $\nabla p^{n+1/2}$ are computed by using an approximation of the second-order accurate temporal discretization of the Navier-Stokes equations:

$$\frac{U^{n+1}-U^n}{\Delta t}+\nabla p^{n+1/2}=-[(U\cdot\nabla)U]^{n+1/2}+\frac{\nu}{2}\Delta(U^n+U^{n+1}) \tag{2.24}$$

or equivalently

$$\frac{U^{n+1}-U^n}{\Delta t}=\mathbf{P}[-[(U\cdot\nabla)U]^{n+1/2}+\frac{\nu}{2}\Delta(U^n+U^{n+1})]. \tag{2.25}$$

Equation (2.25) is an implicit equation that couples the projection with an elliptic equation and requires that values U^{n+1} and $\nabla p^{n+1/2}$ be computed simultaneously. The linear algebra required to solve this implicit equation is computationally expensive, hence equation (2.24) is approximated by first solving for a provisional velocity update U^* which is the solution of

$$\frac{U^*-U^n}{\Delta t}=-\nabla p^{n-1/2}-[(U\cdot\nabla)U]^{n+1/2}+\frac{\nu}{2}\Delta(U^n+U^*). \tag{2.26}$$

The time-centered advective term $[(U\cdot\nabla)U]^{n+1/2}$ appearing in (2.26) is computed using an explicit second-order Godunov method described in the next section. Note that for Euler flow ($\nu=0$), equation (2.26) is not implicit. The new velocity U^{n+1} and pressure gradient $\nabla p^{n+1/2}$ are then computed by applying an approximate projection operator $\tilde{P}$ to the right-hand side of equation (2.26). U^{n+1} is given by

$$\frac{U^{n+1}-U^n}{\Delta t}=\tilde{P}(\frac{U^*-U^n}{\Delta t}) \tag{2.27}$$

and the updated value $\nabla p^{n+1/2}$ is given by

$$\nabla p^{n+1/2}=\nabla p^{n-1/2}+(I-\tilde{P})(\frac{U^*-U^n}{\Delta t}). \tag{2.28}$$

Equations (2.27) and (2.28) are the discrete analog of the requirement that the Hodge decomposition of

$$\frac{U^* - U^n}{\Delta t} + \nabla p^{n-1/2}$$

be equal to the sum

$$\frac{U^{n+1} - U^n}{\Delta t} + \nabla p^{n+1/2}$$

which is already in the form of a Hodge decomposition.

For the initial time step, a value of $\nabla p^{1/2}$ must be supplied. In [8] equations 2.27 and 2.28 are presented as one step of an iterative procedure to calculate a value for $\nabla p^{n+1/2}$ (as well as U^{n+1}). It is shown therein that this iterative procedure converges to a second-order accurate value of $\nabla p^{n+1/2}$ and U^{n+1} after a single iteration. To get the initial value of $\nabla p^{1/2}$, this iterative procedure, which is essentially the same as one time step of the full method, is repeated three times with the velocities reset to the initial conditions after each iteration, and the pressure gradient equal to zero for the first iteration. Complete details of the iterations are found in [8].

The projection operator used in the method is an approximate projection based on the MAC-projection as was discussed in Section 2.4. A further discussion of its form appears in Section 2.5.3.

The method presented here employs an explicit procedure for the evaluation of the advective terms and hence is required to have a time-step restriction for stability. If one examines the stability requirements for the advective terms applied to a linear constant coefficient version of the Euler equations, it is found that this method is stable for

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

where the maximum is taken as the maximum over all cells. In practice Δt is recomputed each time step by

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

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

2.5.2 Evaluation of the Advective Term

The term $[(U \cdot \nabla)U]^{n+1/2}$ required in equation (2.26) is computed by an explicit method based on a second-order Godunov procedure. The basic idea of the procedure is

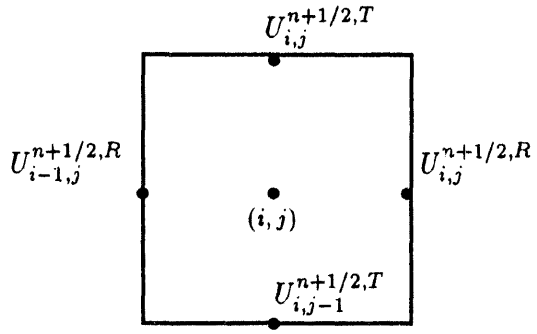


Figure 2.3: Location of time-centered edge values.

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. The velocities at time $n+1/2$ at the top edge of cell (i,j) are denoted by $U_{i,j}^{n+1/2,T}$, and $U_{i,j}^{n+1/2,R}$ denotes the velocities at time $n+1/2$ at the right edge of cell (i,j). (See figure 2.3 for location of cell-edge variables.)

Once the values $U^{n+1/2,T}$ and $U^{n+1/2,R}$ are computed, they are differenced to get an approximation to the advective derivative term $[(U\cdot\nabla)U]^{n+1/2}$. Suppressing the $n+1/2$ index, the advective term is given by:

$$\begin{aligned}(U\cdot\nabla)U_{i,j} = (uU_x + vU_y)_{i,j} &= \frac{(u_{i,j}^R + u_{i-1,j}^R)}{2}\frac{(U_{i,j}^R - U_{i-1,j}^R)}{h} \\ &+ \frac{(v_{i,j}^T + v_{i,j-1}^T)}{2}\frac{(U_{i,j}^T - U_{i,j-1}^T)}{h}.\end{aligned} \tag{2.29}$$

The necessary steps to compute the time-centered cell-edge velocities are explained below. (See [9] for more details.)

Given cell-centered values of the velocities U^n, the second-order Taylor series expansion for the time-centered values of the velocities at the top and side edges of the cell are given by

$$U_{i,j}^{n+1/2,T} = U_{i,j}^n + \tfrac{h}{2}(U_y^n)_{i,j} + \tfrac{\Delta t}{2}(U_t^n)_{i,j} \tag{2.30}$$

and

$$U_{i,j}^{n+1/2,R} = U_{i,j}^n + \tfrac{h}{2}(U_x^n)_{i,j} + \tfrac{\Delta t}{2}(U_t^n)_{i,j}. \tag{2.31}$$

The right-hand side of the Navier-Stokes equations (2.22) is used to replace the temporal derivatives in equations (2.30) and (2.31) with spatial derivatives which, after rearranging terms, yields

$$U_{i,j}^{n+1/2,T} = U_{i,j}^n + [\tfrac{h}{2} - \tfrac{\Delta t}{2}v_{i,j}^n](U_y^n)_{i,j} + \tfrac{\Delta t}{2}[-u_{i,j}^n(U_x^n)_{i,j} + \nu(\Delta U^n)_{i,j} - \nabla p_{i,j}^n] \tag{2.32}$$

$$U_{i,j}^{n+1/2,R} = U_{i,j}^n + [\tfrac{h}{2} - \tfrac{\Delta t}{2} u_{i,j}^n](U_x^n)_{i,j} + \tfrac{\Delta t}{2}[-v_{i,j}^n(U_y^n)_{i,j} + \nu(\Delta U^n)_{i,j} - \nabla p_{i,j}^n]. \quad (2.33)$$

In order to approximate these equations, each of the derivative terms must be computed. In all, four terms for each edge value must be approximated; in the case of equation (2.32) these are U_y^n, U_x^n, ΔU^n, and ∇p^n. The approximation to each of these terms is explained below; an analogous method is used for the terms in equation (2.33).

The normal and tangential derivatives to each cell edge are computed using a Godunov procedure developed by Colella in [25]. For the normal derivative term (which in this case is U_y^n), a fourth-order slope profile in each cell is computed and a slope-limiting procedure is employed to prevent the introduction of new maxima or minima in the velocity field. Specifically

$$[\tfrac{h}{2} - \tfrac{\Delta t}{2} v_{i,j}^n](U_y^n)_{i,j} = \frac{1}{2}[1 - s_L \tfrac{\Delta t}{h} v_{i,j}^n]\delta_y^4(U_{i,j}^n) \quad (2.34)$$

$$s_L = \begin{cases} 1 & \text{if } v_{i,j} > 0 \\ 0 & \text{otherwise} \end{cases}$$

where the term $\delta_y^4(\phi_{i,j})$ is computed by first defining the difference operators

$$\begin{aligned} D_y^c(\phi)_{i,j} &= (\phi_{i,j+1} - \phi_{i,j-1})/2 \\ D_y^+(\phi)_{i,j} &= (\phi_{i,j+1} - \phi_{i,j}) \\ D_y^-(\phi)_{i,j} &= (\phi_{i,j} - \phi_{i,j-1}). \end{aligned}$$

These difference operators are then used to define

$$\hat{\delta}(\phi)_{i,j} = \begin{cases} \min(2|D_y^-(\phi)_{i,j}|, 2|D_y^+(\phi)_{i,j}|) & \text{if } (D_y^-(\phi)_{i,j})(D_y^+(\phi)_{i,j}) > 0 \\ 0 & \text{otherwise} \end{cases}$$

and

$$\delta^f(\phi)_{i,j} = \min(|D_y^c(\phi)_{i,j}|, \hat{\delta}(\phi)_{i,j}) \times sign(D_y^c(\phi)_{i,j}). \quad (2.35)$$

The value of δ^f is the second-order limited slope in each cell. The fourth-order slope is then given by

$$\delta_y^4(\phi)_{i,j} = \min(|\frac{4(D_y^c(\phi)_{i,j})}{3} - \frac{(\delta^f(\phi)_{i+1,j} + \delta^f(\phi)_{i-1,j})}{6}|, \hat{\delta}(\phi)_{i,j}) \times sign(D_y^c(\phi)_{i,j}). \quad (2.36)$$

The transverse derivative terms in equation (2.33) and (2.32) are calculated by using difference operators the stencil of which depends on the local fluid direction, a procedure known as upwind differencing. For example, the transverse derivative term in (2.32)

is uU_x, and its value is computed by using the upwind difference $D^u(U)$

$$(uU_x)_{i,j} \approx (uD_x^u(U))_{i,j} = \begin{cases} u_{i,j}(U_{i+1,j} - U_{i,j})/h & \text{if } u_{i,j} < 0 \\ u_{i,j}(U_{i,j} - U_{i-1,j})/h & \text{otherwise.} \end{cases}$$

The Laplacian terms in (2.32) and (2.33) are computed using the standard five-point approximation to the Laplacian L^5:

$$L^5(U)_{i,j} = \frac{-4U_{i,j} + U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1}}{h^2}$$

The difference approximations to the normal and tangential derivative terms and the Laplacian term in equations (2.32) and (2.33) are now used to compute provisional time-centered edge values. The effect of the pressure gradient term in (2.32) and (2.33) is not included in these values. Replacing each term in (2.32), except for the pressure, yields a provisional approximation to $U^{n+1/2,T}$ which is denoted $\tilde{U}^T$:

$$\tilde{U}^T_{i,j} = U^n_{i,j} + \frac{1}{2}[1 - s_L \tfrac{\Delta t}{h} v^n_{i,j}]\delta_y^4(U^n_{i,j}) + \tfrac{\Delta t}{2}[-(uD_x^u(U))_{i,j} + \nu L^5(U^n)_{i,j}] \tag{2.37}$$

The value of $\tilde{U}^T_{i,j}$ represents the Taylor series extrapolation to the top edge of cell (i,j). A similar extrapolation to the bottom edge of the cell is also computed and denoted by $\tilde{U}^B_{i,j}$:

$$\tilde{U}^B_{i,j} = U^n_{i,j} - \frac{1}{2}[1 + s_L \tfrac{\Delta t}{h} v^n_{i,j}]\delta_y^4(U^n_{i,j}) + \tfrac{\Delta t}{2}[-(uD_x^u(U))_{i,j} + \nu L^5(U^n)_{i,j}]. \tag{2.38}$$

Values of $\tilde{U}^R_{i,j}$ and $\tilde{U}^L_{i,j}$ at cell edges are computed in a similar manner.

The Taylor series procedure just described produces two values of each provisional velocity at each cell edge. For example, at the top edge of cell (i,j), the value of $\tilde{U}^T_{i,j}$ and $\tilde{U}^B_{i,j+1}$ are computed. These values must still be corrected for the omission of the pressure gradient term in equations (2.32) and (2.33), but first a procedure which is based on the solution of the Riemann problem for Burgers' equation is used to produce a single value of each provisional time-centered velocity at each cell edge. For edges at the top of cells this results in the single value $\tilde{U}^T_{i,j}$, and similarly $\tilde{U}^R_{i,j}$ at the vertical cell edges. For example, given the pair $\tilde{U}^T_{i,j}$ and $\tilde{U}^B_{i,j+1}$,

$$\tilde{U}^T_{i,j} = \begin{cases} \tilde{U}^T_{i,j} & \text{if } v_{i,j} > 0, v_{i,j+1} > 0 \\ \tilde{U}^B_{i,j+1} & \text{if } v_{i,j} < 0, v_{i,j+1} < 0 \\ (\tilde{U}^T_{i,j} + \tilde{U}^B_{i,j+1})/2 & \text{otherwise.} \end{cases}$$

To complete the computation of the time-centered edge values $U^{n+1/2,T}$ and $U^{n+1/2,R}$, the pressure gradient term from equations (2.32) and (2.33) must be approximated. The

effect of the pressure gradient is approximated by applying a MAC projection operator to the computed values $\tilde{v}^T$ and $\tilde{u}^R$. The values of $u^{n+1/2,R}$ and $v^{n+1/2,T}$ are then given by the resulting divergence-free field. In other words, $u^{n+1/2,R}$ and $v^{n+1/2,T}$ satisfy

$$\frac{u_{i,j}^{n+1/2,R} - u_{i-1,j}^{n+1/2,R}}{h} + \frac{v_{i,j}^{n+1/2,T} - v_{i,j-1}^{n+1/2,T}}{h} = 0,$$

and are given by

$$\begin{aligned} u_{i,j}^{n+1/2,R} &= \tilde{u}_{i,j}^{R} - G_1(\phi)_{i,j} \\ v_{i,j}^{n+1/2,T} &= \tilde{v}_{i,j}^{T} - G_2(\phi)_{i,j} \end{aligned}$$

where ϕ is the solution of

$$DG(\phi)_{i,j} = \frac{\tilde{u}_{i,j}^{R} - \tilde{u}_{i-1,j}^{R}}{h} + \frac{\tilde{v}_{i,j}^{T} - \tilde{v}_{i,j-1}^{T}}{h}.$$

An approximation to the gradient $G(\phi)$ is also subtracted from the values of $\tilde{u}^T$ and $\tilde{v}^R$ to yield $v^{n+1/2,R}$ and $u^{n+1/2,T}$. Specifically, the four nearest values of the relevant gradient component are averaged and subtracted from $\tilde{u}^T$ and $\tilde{v}^R$

$$\begin{aligned} u_{i,j}^{n+1/2,T} &= \hat{u}_{i,j}^{T} - \frac{G_1(\phi)_{i,j} + G_1(\phi)_{i-1,j} + G_1(\phi)_{i,j+1} + G_1(\phi)_{i-1,j+1}}{4} \\ v_{i,j}^{n+1/2,R} &= \tilde{v}_{i,j}^{R} - \frac{G_2(\phi)_{i,j} + G_2(\phi)_{i,j-1} + G_2(\phi)_{i+1,j} + G_2(\phi)_{i+1,j-1}}{4}. \end{aligned}$$

2.5.3 The Approximate Projection

Once the advective terms have been computed, a provisional update to the velocity U^* is given by the solution of

$$\frac{U^* - U^n}{\Delta t} = -[(U \cdot \nabla)U]^{n+1/2} - \nabla p^{n-1/2} + \frac{\nu}{2}\Delta(U^n + U^*). \tag{2.39}$$

The approximate projection operator is then applied to provide updates to the solution U^{n+1} and the pressure gradient $\nabla p^{n+1/2}$ by

$$\begin{aligned} \frac{U^{n+1} - U^n}{\Delta t} &= \tilde{P}\left(\frac{U^* - U^n}{\Delta t}\right) \\ \nabla p^{n+1/2} &= \nabla p^{n-1/2} + (I - \tilde{P})\left(\frac{U^* - U^n}{\Delta t}\right). \end{aligned}$$

The approximate projection is applied to the provisional velocity correction rather than U^*. This is consistent with the earlier definition of the projection in which $P(U) = U^D$

was defined to have the no-flow condition at the boundary. For the vortex patch problems studied in this thesis, U^* does not have the no-flow boundary condition. One could instead redefine the projection operator P so that the space of divergence-free fields onto which velocities are projected has the desired boundary conditions of U. This would be necessary if the boundary conditions for U were time dependent. Since an approximate projection $\tilde{P}$ is being used, applying $\tilde{P}$ to the correction is not equivalent to applying $\tilde{P}$ to U^* with correct boundary conditions.

Note that the form of U^* presented above is not required to have the same values of U^n at the computational boundary. If this were the case, the Poisson equation for ϕ associated with the projection of the correction would have homogeneous Neumann boundary conditions. Since the new pressure $\nabla p^{n+1/2}$ is simply the sum of the old pressure $\nabla p^{n-1/2}$ and $\nabla\phi$, having homogeneous Neumann boundary conditions on ϕ would imply that the normal derivative of the pressure at the boundary would be constant in time. In the numerical implementation, boundary fluxes for the provisional correction are set by third-order polynomial interpolation from interior values. These fluxes then serve as the Neumann boundary conditions for the projection Poisson problem.

The approximate projection $\tilde{P}$ consists of performing a MAC projection on cell-edge values of the velocity correction that are interpolated from the cell-centered values. A cell-centered gradient is then interpolated from the gradient which results from the MAC projection and then subtracted from the original correction. Specifically, let the provisional correction $C_{i,j} = (C_{1i,j}, C_{2i,j})$ be given by

$$C_{i,j} = \frac{U^* - U^n}{\Delta t}. \tag{2.40}$$

Then cell-edge values of $\tilde{C}_{i,j} = (\tilde{C}_{1i,j}, \tilde{C}_{2i,j})$ are given by

$$\begin{aligned} \tilde{C}_{1i,j} &= \frac{-C_{1i-1,j} + 9(C_{2i,j} + C_{2i+1,j}) - C_{1i+2,j}}{16} \\ \tilde{C}_{2i,j} &= \frac{-C_{2i,j-1} + 9(C_{2i,j} + C_{2i,j+1}) - C_{2i,j+2}}{16}. \end{aligned}$$

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

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

of the cell-edge values where ϕ is the solution of

$$DG(\phi)_{i,j} = D(\tilde{C})_{i,j}. \tag{2.41}$$

The velocity correction is then computed by interpolating the cell-edge values of $G(\phi)$ back to the cell-center location and subtracting this value from the original correction.

$$\frac{u^{n+1} - u^n}{\Delta t} = C_{1i,j} - \frac{G_1(\phi)_{i,j} + G_1(\phi)_{i-1,j}}{2} \tag{2.42}$$

$$\frac{v^{n+1} - v^n}{\Delta t} = C_{2i,j} - \frac{G_2(\phi)_{i,j} + G_2(\phi)_{i,j-1}}{2}. \tag{2.43}$$

Numerical convergence results and a low CFL number study of the scheme presented in the Section appear in Chapter 6.

2.6 A Multigrid Method for the Single Grid Projection

The major task, in terms of computational cost, in the implementation of the projection method presented in this chapter is the solution of the Poisson problem (2.41). This problem could, in theory, be solved using any standard direct or iterative solver, but a multigrid method is described here because it forms the basis of the refined grid method presented in Section 4.3.

The multigrid method is a technique that greatly increases the convergence rate of iterative solvers such as Jacobi or Gauss-Seidel. It was observed that iterative solvers decrease high frequency errors in the solution at a much greater rate than low frequency errors. The multigrid method in effect increases the reduction of low frequency errors by solving residual problems on coarser grids that represent the residuals as high frequency components. Solutions of residual problems are then interpolated back onto the original fine grids. These techniques fit naturally into the locally refined-grid context. For an introduction to multigrid methods see [15].

Basic to the multigrid method is the idea of the residual problem. If one is attempting to solve the linear equation

$$L(\psi) = f \tag{2.44}$$

and has some approximate solution ψ^0, then a solution of (2.44) can be found by solving the residual equation

$$L(\xi) = \rho \tag{2.45}$$

where

$$\rho = f - L(\psi^0). \tag{2.46}$$

The solution of the original problem is then given by $\psi = \psi^0 + \xi$. When L is a Laplacian, as it is in the solution of the Poisson problem associated with the projection, boundary conditions must be supplied. Since the solution of the residual problem is eventually added to the initial guess, which is assumed to satisfy the correct boundary conditions, the residual problem is specified to have homogeneous boundary conditions.

If it is known that the residual is smooth, then ρ can be adequately represented on a coarser grid and the residual problem can be solved on that grid at a reduced computational cost. Likewise, if an approximate solution of the residual problem on the coarse grid is known to have a smooth residual, another residual problem can be formed on a yet coarser grid. This process can be recursively repeated until some predetermined level of coarseness is reached.

For the Poisson problem associated with the MAC projection, the linear operator L of interest is DG. For any given grid, a solution to (2.6) can be computed by an iterative process called relaxation. In general, if ψ is a solution of (2.44), then ψ satisfies

$$\psi_{i,j} = \psi_{i,j} - \lambda(L(\psi)_{i,j} - f_{i,j}).$$

When $L = DG$, it can be shown that for appropriate values of λ, given an initial guess to the solution ψ^0, the following iteration method will converge to the solution ψ

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

Each of these iterations is called a relaxation. Different relaxation methods depend on the parameter λ and the order in which the $\psi^{k+1}_{i,j}$ are updated. The method used here is Gauss-Seidel with red-black ordering. The value of λ is given by $-h^2/4$ which for the operator DG in the interior of a grid gives

$$\psi^{k+1}_{i,j} = \frac{\psi_{i+1,j} + \psi_{i-1,j} + \psi_{i,j+1} + \psi_{i,j-1} - h^2 f_{i,j}}{4}. \tag{2.47}$$

Each value of ψ^{k+1} depends only on values of ψ at its four nearest neighbor cells. The form of relaxation at the boundary is modified to take into account the form of DG at the boundary. The value of λ which makes the $\psi^{k+1}_{i,j}$ at a specific cell not depend on values of $\psi^k_{i,j}$ at that cell is $h^2/4$. It is also the value of λ that produces the largest reduction in the error of the solution for each relaxation. In red-black ordering, the ψ^{k+1} are updated in two sweeps, the first of which updates values of $\psi^{k+1}_{i,j}$ at cell locations corresponding to the red squares on a checkerboard using the $\psi^k_{i,j}$ at black-square locations. Next, the black

$\psi_{i,j}^{k+1}$ are computed using the new red-square $\psi_{i,j}^{k+1}$. This method is desirable because each of the two red-black sweeps can be fully vectorized on a vector computer and new values can overwrite old values with no additional storage.

Operator notation will be used for each of the multigrid components, and in particular, a Gauss-Seidel relaxation sweep will be denoted by

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

Gauss-Seidel iteration is known to damp high frequency modes in the error at a much higher rate than low frequency modes. This allows one to solve the residual problem after a few Gauss-Seidel sweeps on a coarser grid with good accuracy. The computation of a residual will be described by the operator R^M defined by

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

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, coarse grids have precisely the same physical dimension as fine grids, and the refinement ratio, which is the ratio of coarse to fine cell width, is 2. Suppose $\rho_{i,j}^f$ is represented on the fine grid with dimensions $2N_x$ by $2N_y$ and $\rho_{i,j}^c$ on the coarse grid with dimensions N_x by N_y. (See figure 2.4.) Values of $\rho_{i,j}^c$ are computed from $\rho_{i,j}^f$ by the averaging operator A^M by

$$\rho_{i,j}^c = A^M(\rho^f)_{i,j} = \frac{\rho_{2i,2j}^f + \rho_{2i-1,2j}^f + \rho_{2i,2j-1}^f + \rho_{2i-1,2j-1}^f}{4}. \tag{2.48}$$

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,

$$\xi_{2i,2j}^f = \frac{9\xi_{i,j}^c + 3\xi_{i+1,j}^c + 3\xi_{i,j+1}^c + \xi_{i+1,j+1}^c}{16} \tag{2.49}$$

while

$$\xi_{2i-1,2j}^f = \frac{9\xi_{i,j}^c + 3\xi_{i-1,j}^c + 3\xi_{i,j+1}^c + \xi_{i-1,j+1}^c}{16}.$$

The notation

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

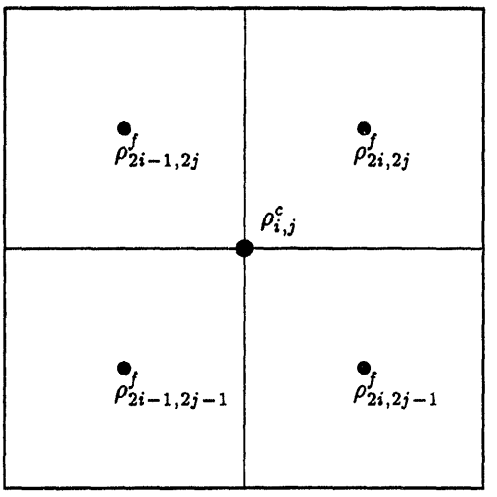


Figure 2.4: Section of a grid showing multigrid refinement.

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, ψ^k, h)
  $\psi^k = G^M(\psi^k, f, h)$
  if $(h < htol)$
    $\rho^f = R^M(\psi^k, f, h)$
    $\rho^c = A^M(\rho^f)$
    $\xi^c = MGV(\rho^c, 0, 2h)$
    $\psi^{k+1} = \psi^k + I^M(\xi^c)$
    $\psi^{k+1} = G^M(\psi^{k+1}, f, h)$
  endif
  return ψ^{k+1}
END MGV

For the operator appearing in the Poisson problem (2.41) for the projection, a single V-cycle will usually reduce the size of the residual by a factor of five to ten. The full multigrid

algorithm consists of repeatedly performing V-cycles to improve an initial guess to the solution until the norm of the residual is smaller than some stopping criterion. In pseudo-code, this procedure takes the form

MGFULL(f, ψ^0, h)

$k = 0$

$\rho = R^M(\psi^k, f, h)$

while ($\|\rho\| > tol$)

$\psi^{k+1} = \text{MGV}(f, \psi^k, h)$

$\rho = R^M(\psi^{k+1}, f, h)$

$k = k + 1$

return ψ^{k+1}

END MGFULL

For the computations in this thesis, the norm used for defining the size of the residual is the infinity (maximum) norm, and the error tolerance is set to 10^{-8}. The number of points in the computational grid is in all cases restricted to be a power of two so that the recursion in the V-cycles continues until a grid with a single point is reached.

The residual and relaxation operators for the multigrid method must be modified near boundaries. In all cases, the form of these operators at boundaries are derived from the form of DG near the boundary. In practice, additional computational cells are formed around the boundary of the domain, and values are set in these cells so that if the form of the relaxation and residual operators that is used in the interior of the grid is used at the boundary, the correct value will result. These boundary values are also used to define the interpolation near boundaries with the same stencil as in the interior.

Chapter 3

A Vorticity-Based Method for a Single Grid

Recall from Chapter 1 that for incompressible flow a useful form of the Navier-Stokes equations (1.1) can be derived for the vorticity, which is defined as the curl of the velocity field. In two dimensions, the vorticity is simply the scalar

$$\omega = \nabla \times U = v_x - u_y.$$

Taking the curl of both sides of equation (1.1) yields the vorticity form of the Navier-Stokes equations for incompressible flow:

$$\begin{aligned} \omega_t &= -u\omega_x - v\omega_y + \nu\Delta\omega \qquad (3.1) \\ \nabla \cdot U &= 0. \end{aligned}$$

The boundary conditions associated with equations (3.1) at physical boundaries are specified on the velocities and not on the vorticity. Therefore, the no-flow condition (1.3) is enforced for Euler flow, and additionally the no-slip condition (1.4) for viscous flow. There are no explicit physical boundary conditions for the vorticity at solid walls.

For incompressible flow, the velocities can be recovered from the vorticity by way of the stream function ψ which satisfies the following equations:

$$\psi_y = u \qquad (3.2)$$

$$-\psi_x = v \qquad (3.3)$$

and

$$\Delta\psi = -\omega. \tag{3.4}$$

Equation (3.4) is a Poisson problem and hence requires a boundary condition. For Euler flow, the no-flow condition at solid walls translates into the condition that the tangential derivative of the stream function along each wall vanishes. This implies that the stream function is constant along a physical boundary. In other words, the no-flow condition on the velocities translates into homogeneous Dirichlet boundary conditions for the stream function:

$$U\cdot\hat{n} = 0 \Leftrightarrow \psi = 0 \text{ on } \partial\Omega.$$

Unfortunately, if the flow is viscous and the no-slip condition also applies at solid wall boundaries, it is easy to see that ψ must also satisfy homogeneous Neumann boundary conditions at solid walls:

$$U = 0 \Leftrightarrow \frac{\partial\psi}{\partial\hat{n}} = 0 \text{ on } \partial\Omega.$$

Since either of these two boundary conditions on ψ are sufficient to fully specify the solution to equation (3.4), the equation is overdetermined in the viscous case.

The lack of boundary conditions for the vorticity and the surplus of boundary conditions for the stream function present a considerable challenge in the design of finite difference methods for integrating the vorticity stream-function form of the Navier-Stokes equations in two dimensions. Recent progress in designing methods has been made, however. Quartapelle and Valz-Gris have presented a way to replace one of the boundary conditions for the stream function with a constraint on the evolution of the vorticity [44] [43]. Anderson [5], Reider [45], and Hou and Wetton [34] have presented methods based on similar ideas. A three-dimensional extension to the two-dimensional results has yet to be completed, and this may be the biggest drawback to using vorticity-stream function methods for realistic engineering problems.

The method presented here is a fairly straightforward discretization of the vorticity equations. Its purpose is to provide a means to validate the results gained from the projection method on vortex patch problems and to demonstrate the usefulness of the machinery developed for the refined grid projection method. Because the method is used only for modeling vortex patches with free-space boundary conditions, the difficulties concerning the over-specification of boundary conditions for the stream function that are discussed above can be avoided. In short, the value of the vorticity and the stream function can be

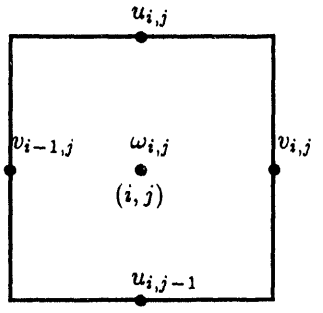


Figure 3.1: Cell variable locations for the vorticity method.

specified far from the patch. It is certainly not the case that the vorticity stream-function method presented here is immediately applicable to problems concerning viscous flow in the presence of boundaries.

The numerical method presented here was designed so that it could be easily extended to refined grids using the same machinery developed for the refined grid projection method described in Chapter 4. The method is also designed to have two important properties. First, because the method is to be used to model the evolution of vortex patches, it is desirable to have the method conserve the total amount of vorticity in the flow (in the absence of viscosity). Conservation of vorticity for Euler flow ensures that a patch of vorticity is really only changing shape and not shrinking (or growing) in size. The second property built into the method is commonly referred to as that of being free-stream preserving. Loosely defined, a method that is free-stream preserving will not change the value of areas of constant vorticity during a particular time step. The property of being free-stream preserving ensures that a constant patch of vorticity will essentially remain constant away from the edges of the patch. Also, if the patch is surrounded by a constant but nonzero vorticity field, then the value of the vorticity away from a patch will also remain constant.

3.1 Spatial Discretization

As in the projection method discussed in the previous chapter, a staggered computational grid is employed for the vorticity method. However, the vorticity and stream function are represented at the center of grid cells, and the velocities are represented at cell edges. The location of the u and v velocities are opposite as in the projection method,

u is located at the top and bottom of cells and v at the sides (see figure 3.1). This data arrangement allows one to define the discrete vorticity as a simple difference much like a MAC divergence:

$$\omega_{i,j} = \frac{v_{i,j} - v_{i-1,j}}{h} - \frac{u_{i,j} - u_{i,j-1}}{h}. \tag{3.5}$$

If the vector $U^\star$ is defined as $U^\star_{i,j} = (v_{i,j}, -u_{i,j})$, then

$$\omega_{i,j} = D(U^\star)_{i,j}. \tag{3.6}$$

Cell-edge velocities $u_{i,j}$ and $v_{i,j}$ can be defined from cell-centered values of the stream function $\psi_{i,j}$ by

$$\begin{aligned} v_{i,j} &= -\frac{\psi_{i+1,j} - \psi_{i,j}}{h} \\ u_{i,j} &= \frac{\psi_{i,j+1} - \psi_{i,j}}{h}. \end{aligned} \tag{3.7}$$

This implies that

$$G(\psi)_{i,j} = -U^\star_{i,j} \tag{3.8}$$

where G is the staggered grid gradient. Therefore, $\psi_{i,j}$ is determined by $\omega_{i,j}$ by the equation

$$DG(\psi)_{i,j} = -\omega_{i,j} \tag{3.9}$$

with appropriate boundary conditions. This evaluation of the stream function and the velocities from the vorticity closely resembles the form of the MAC projection presented in Section 2.3 with the vorticity taking the place of the MAC divergence and the velocities resembling the gradient part of the Hodge decomposition. In terms of the numerical implementation, the two problems are virtually identical and hence the same numerical algorithm developed for the MAC projection can be used for this problem as well.

Boundary conditions must be supplied for equation (3.9). In this paper, only two kinds of boundary conditions are implemented: periodic conditions and free-space conditions for vortex patch problems. Periodic domains pose absolutely no difficulties for specifying either the boundary conditions for the vorticity or for the vorticity-stream function Poisson problem (3.9). The vorticity-stream function form of the Navier-Stokes equations are readily implemented on a doubly-periodic computational domain simply by specifying that operators use periodic conditions at grid boundaries.

For vortex patch problems, the vorticity outside the patch is assumed to be a constant value, and the edge of the physical domain which the computational grid represents

is arranged to be a very large distance from the patch. A consequence of this is that one can specify the value of the velocities at the edge of the computational domain to be exactly those that would result from a perfectly circular patch of vorticity (for which the exact velocity field is known) instead of the slightly irregular patches actually being modeled. The size of the domain is chosen so that the errors resulting from this slight change in the boundary conditions is smaller than the discretization error near the patch.

Because of the staggered grid arrangement of the data, the value of the tangential component of the velocity is the value that can be specified at the grid boundaries. This translates into specifying Neumann boundary conditions for the stream function which are the same conditions used for the Poisson problem associated with the MAC projection. In the implementation of the vorticity method, the Poisson problem involves the update to the vorticity and the update to the stream function, hence the boundary conditions are homogeneous Neumann. It will be shown that the solvability constraint for the Neumann problem is guaranteed by the fact that the method is conservative.

As previously mentioned, this method was designed to conserve the total amount of vorticity in Euler flow. The way in which this is accomplished is by approximating the advective derivatives by finite differences in conservative form. Recall from Chapter 1 that for Euler flow, the vorticity equations can be written in terms of a divergence of the vector $\omega U = (\omega u, \omega v)$:

$$\omega_t = -\nabla \cdot (\omega U). \tag{3.10}$$

Equation (3.10) can be discretized using a MAC divergence to provide a discrete analog to the conservation of vorticity. Specifically, given cell-edge values of the quantity $(\omega U)_{i,j}$ as represented in figure 3.2, the advective derivatives are approximated by

$$(\omega u)_x + (\omega v)_y \approx \frac{(\omega u)_{i,j} - (\omega u)_{i-1,j}}{h} + \frac{(\omega v)_{i,j} - (\omega v)_{i,j-1}}{h}. \tag{3.11}$$

The right-hand side of equation (3.11) is the MAC divergence of $(\omega U)_{i,j}$, hence by applying the discrete divergence theorem (2.3.1), it can be shown that the sum of the advective derivatives only depends on the boundary conditions of $(\omega U)_{i,j}$. For the problems presented in this paper, the boundary conditions are such that the total sum of the advective derivatives is zero, i.e.

$$\sum_{i,j} D(\omega U)_{i,j} = 0. \tag{3.12}$$

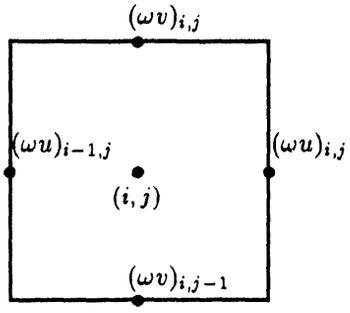


Figure 3.2: Locations of the edge values ωU.

As will be seen in the next section, updates to the vorticity in the implementation of this method are of the form

$$\frac{\omega^* - \omega^n}{\Delta t} = -D(\omega U)_{i,j}$$

where ω^* is a prediction of the updated vorticity. Since the relationship between the stream-function and the vorticity is linear, one can compute the update of the stream-function from the equation

$$L^5(\frac{\psi^* - \psi^n}{\Delta t}) = -\frac{\omega^* - \omega^n}{\Delta t}$$

with homogeneous Neumann boundary conditions. The discrete solvability condition for this Neumann problem is then guaranteed by equation (3.12). Once the update for the stream function is known, an update for the velocities can be explicitly computed.

3.2 Computation of Advective Derivatives

As previously mentioned, in order to make the method described here conserve the total sum of vorticity in the problem, the advective terms are discretized using a conservative MAC divergence. Specifically, updates to the vorticity take the form

$$\frac{\omega^* - \omega^n}{\Delta t} = -D(\omega U)_{i,j}.$$

The way in which the cell-edge values of ωU are computed is motivated by the desire to make the overall method free-stream preserving.

Suppose that $(\omega U)_{i,j}$ is the product of the cell-edge values $\hat{U}_{i,j}$ and $\hat{W}_{i,j} = (\hat{\omega}_{1i,j}, \hat{\omega}_{2i,j})$. The first component of $\hat{W}_{i,j}$ corresponds to a value of the vorticity at the right edge of cell

(i,j), and likewise $\hat{\omega}_{2i,j}$ corresponds to the value of the vorticity at the top edge of cell (i,j). Now, if the form of ωU is given by

$$\begin{aligned}(\omega u)_{i,j} &= \hat{u}_{i,j}\hat{\omega}_{1i,j} \\ (\omega v)_{i,j} &= \hat{v}_{i,j}\hat{\omega}_{2i,j},\end{aligned}$$

and if in a certain area of the domain the values of $\hat{W}_{i,j}$ are some constant value C, then the value of the advective derivative would be given by

$$\frac{(\hat{u}\hat{\omega}_1)_{i,j} - (\hat{u}\hat{\omega}_1)_{i-1,j} + (\hat{v}\hat{\omega}_2)_{i,j} - (\hat{v}\hat{\omega}_2)_{i,j-1}}{h} = C(\frac{\hat{u}_{i,j} - \hat{u}_{i-1,j} + \hat{v}_{i,j} - \hat{v}_{i,j-1}}{h}). \quad (3.13)$$

Note that the right-hand side of equation (3.13) is exactly the form of the MAC divergence of $\hat{U}$. This implies that for the value of the advective derivative to be zero and hence the vorticity to remain constant in an area of constant vorticity, the edge values $\hat{U}$ should have MAC divergence zero. A simple calculation shows that if the values of $\hat{U}$ are given by an average of the four nearest values of U, the resulting value of $\hat{U}$ will be discretely divergence free (writing u and v in the form of differences of the stream function causes the terms in the divergence to cancel). Therefore $\hat{U}$ is defined as

$$\begin{aligned}\hat{u}_{i,j} &= \frac{u_{i,j} + u_{i,j-1} + u_{i+1,j} + u_{i+1,j-1}}{4} \\ \hat{v}_{i,j} &= \frac{v_{i,j} + v_{i-1,j} + v_{i,j+1} + v_{i-1,j+1}}{4}.\end{aligned}$$

With $\hat{U}$ so defined, the form of the advective derivative becomes

$$(\omega u)_x + (\omega v)_y \approx D(\omega U) = \frac{(\hat{u}\hat{\omega}_1)_{i,j} - (\hat{u}\hat{\omega}_1)_{i-1,j} + (\hat{v}\hat{\omega}_2)_{i,j} - (\hat{v}\hat{\omega}_2)_{i,j-1}}{h}. \quad (3.14)$$

Hence, all that remains is to define the edge values $\hat{W}$ and the definition of the advective derivatives is complete.

The method for computing the edge values $\hat{W}$ is an upwind interpolation procedure that resembles the differencing procedure used in some ENO (essentially non-oscillatory) schemes for hyperbolic conservation laws. (See e.g. Shu and Osher [48].) Edge values are computed by polynomial interpolation where the points used in the interpolation stencil vary from point to point and depend on the local smoothness of the vorticity.

For example, consider the edge value $\hat{\omega}_{1i,j}$. An rth degree polynomial interpolation of the $r+1$ values $\omega_{i_o,j}, \omega_{i_o+1,j}, \ldots \omega_{i_o+r,j}$ is used to interpolate $\hat{\omega}_{1i,j}$. Note that these values are consecutive and each is vertically centered around $\hat{\omega}_{1i,j}$. The procedure for choosing

which $r+1$ values are used is inductively defined as follows: Choose the initial value of i_o to be the index of the upwind value of $\omega_{i,j}$, i.e.

$$i_o^0 = \begin{cases} i+1 & \text{if } u_{i,j} < 0 \\ i & \text{otherwise.} \end{cases}$$

Next, for $k = 1, r$, add one point to the stencil depending on the smoothness of the vorticity as measured by the divided differences of the vorticity $f[\omega_{i,j}]$ (For an introduction to divided differences, see e.g. [26].) Specifically,

$$i_o^{k+1} = \begin{cases} i_o^k + 1 & \text{if } |f[\omega_{i_o^k,j}, \omega_{i_o^k+1,j}, \ldots \omega_{i_o^k+k,j}]| < |f[\omega_{i_o^k-1,j}, \omega_{i_o^k,j}, \ldots \omega_{i_o^k+k-1,j}]| \\ i_o^k & \text{otherwise.} \end{cases}$$

With the definition of $\hat{W}$ complete, the advective derivative is given by equation (3.14).

In the implementation of this method, r is taken to be 3. When the vorticity field consists of a constant patch of vorticity with a constant background vorticity, the above interpolation procedure produces edge values that are either equal to the value of the patch or to the background vorticity. In essence, the interpolation procedure does not smear discontinuous jumps in the vorticity field. When coupled with the TVD time integrator described in the next section, the resulting scheme can be used to model the evolution of vortex patches without introducing new maxima or minima in the solution and without excessive smearing of patch boundaries.

3.3 Temporal Discretization

The time-stepping strategy for the vorticity stream function method is based on an explicit Runge-Kutta integrator applied to

$$\omega_t = -(u\omega)_x - (v\omega)_y + \nu\Delta\omega. \tag{3.15}$$

The form of the integrator is taken from Shu and Osher [47] where the authors develop Runge-Kutta methods for use with conservation laws that are total variation diminishing (TVD). In the present setting it is observed that the TVD Runge-Kutta schemes prevent the numerical solutions of the vorticity field from oscillating near the boundary of vortex patches where the vorticity field is discontinuous.

Given the values of the velocities and vorticities at a given time step n, namely ω^n, u^n, and v^n, if F is defined as

$$F(\omega, u, v) = -(u\omega)_x - (v\omega)_y + \nu\Delta\omega, \tag{3.16}$$

then a third-order TVD Runge-Kutta method for computing ω^{n+1} is given by

$$\begin{aligned} \omega^1 &= \omega^n + \Delta t F(\omega^n, u^n, v^n) \\ \omega^2 &= \frac{3}{4}\omega^n + \frac{1}{4}\omega^1 + \frac{\Delta t}{4}F(\omega^1, u^1, v^1) \\ \omega^{n+1} &= \frac{1}{3}\omega^n + \frac{2}{3}\omega^2 + \frac{2}{3}\Delta t F(\omega^2, u^2, v^2). \end{aligned} \tag{3.17}$$

This Runge-Kutta method is TVD for values of $C \leq 1$ under the CFL restriction

$$\Delta t \leq Ch\lambda_o$$

where λ_o is such that the first-order method

$$\omega^{n+1} = \omega^n + \Delta t F(\omega^n, u^n, v^n) \tag{3.18}$$

is assumed to be TVD whenever

$$\Delta t \leq h\lambda_o.$$

For the method described by equation (3.18) to be TVD for the linear advection equation it is necessary that

$$\Delta t \leq h \max_{i,j}(|u_{i,j}| + |v_{i,j}|)$$

where the maximum is over all cells. Therefore, in practice the time step for the full method is set every time step to

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

with $C = 0.9$.

Note that at each Runge-Kutta substep, the values of the velocities must be updated from the value of the intermediate vorticity field by solution of the stream-function equation (3.4). In the numerical implementation of this method, the stream function is recomputed at every Runge-Kutta substep solving a Poisson equation for the update in the stream function. In other words, the equation

$$\Delta(\delta\psi) = -\delta\omega \tag{3.19}$$

is solved where $\delta\omega$ is the the change in ω for that particular substep. Once $\delta\psi$ is computed, it is added to ψ, and updated velocities are computed from equation (3.2). For the vortex patch problems studied in this thesis, the boundary conditions for equation (3.19) are homogeneous Neumann. This is discussed further in Section 5.4.

A numerical convergence study of this method appears in Chapter 6.

Chapter 4

A Projection Method on Refined Grids

The accuracy of a numerical method for modeling fluid flow depends on the number of computational elements used. For finite difference methods, accuracy of the method depends on the accuracy of the finite difference operators used to approximate derivatives, the distance between grid points, and the smoothness of the solution. This dependence of the accuracy on the smoothness of the solution indicates that a finite difference method on a uniform grid will be least accurate where the solution is least smooth. This implies that, in order for errors in regions of large variation in the solution to be the same size as those in regions where the solution is smoother, the grid spacing in the regions of large variation must be smaller. Unfortunately, these regions of large variation are often the parts of the flow that are of most interest. However, if a uniform grid is used, and the grid spacing is set so that the errors associated with the least smooth parts of the flow are small, then grid points are in a sense being wasted in smooth regions of the flow . An increase in the efficiency of a finite difference method will result if the grid size in smooth regions of the flow is larger than that in regions of large variation.

No finite difference method can resolve features of a flow that are smaller than the distance between grid points. Therefore, if one is interested in completely resolving every feature of the flow, the grid spacing at any point must be smaller than the smallest feature of the flow at that point. Henshaw et al. [33] have shown that for the Navier-Stokes equations, there is a lower limit to scales of the flow which depends on the viscosity of the

fluid. The smaller the amount of viscosity, the smaller the important scales become. For Euler flow, arbitrarily small scales can form in the flow. From a computational point of view, this implies that the smaller the viscosity is in the flow, the smaller the grid spacing must be to fully resolve the flow.

The method presented here attempts to reduce errors and improve resolution near small scales in the flow by refining the computational grid in these locations. Similar methods using this type of grid refinement have been successfully developed for modeling compressible flow and systems of hyperbolic conservation laws in general (See for example, Bell et al. [7], Berger and Colella [12], or Berger and Oliger [14].)

4.1 Grid Refinement Terminology and Notation

The grid refinement strategy used in this thesis is based on a cell-centered discretization in which a rectangular block of grid cells is subdivided by some refinement ratio so that fine grid cell edges line up with coarse grid cell edges. (See figure 4.1.) 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. Child grids are restricted to lie inside parent grids, so each child has only one 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. Any particular grid can have any number of non-overlapping child grids each of which must have the same refinement factor.

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 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 or in the case where the parent grid is contiguous to another grid with the same cell size, the required extra layer of

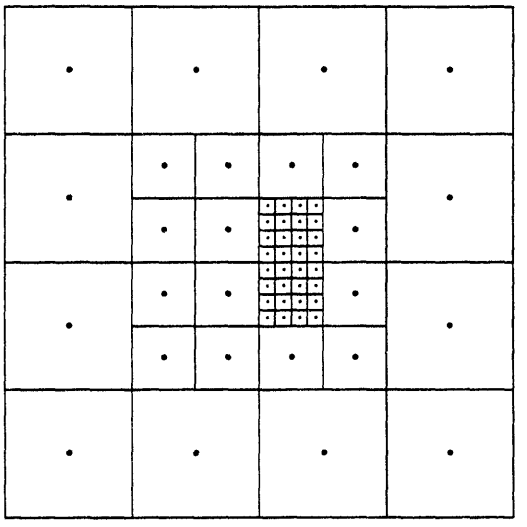


Figure 4.1: Cell centered grid refinement with refinement factors 2 and 4.

cells surrounding a child grid can be the boundary cells of the parent grid.

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 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.

Two superscripts on variables will indicate the grid level and particular grid within a level. When a time step is also indicated, the time superscript will always follow the variable name directly, while grid superscripts and subscripts will be placed to the outside. For instance $(U^n)^{l,k}$ 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 $n+1/2$. 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 equal to 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.

4.2 Time-stepping Procedure

In recent methods for gas dynamics which employ grid refinement, solutions are updated using time refinement as well as spatial grid refinement [12],[14]. 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. One obvious reason 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 they do at higher ones, 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. Since a majority of the grid points lie in the finest grid levels for the grid structures used in this thesis, the computational savings associated with temporal refinement would be modest. Also, since the flow is incompressible, there is not a finite limit to the speed at which disturbances can propagate in the flow. Therefore, it is not clear whether pressure effects can be adequately modeled with local projections. Almgren et al. [1] implement a projection method that does refine in time. Velocities on finer grids are updated with a smaller time step than coarser grids and hence more frequently. Projections on refined grids for time steps that do not correspond to coarse grid time steps are done using Dirichlet boundary data interpolated in time and space from coarse grid values. Only at time steps where coarse and fine grids coincide in time are projections done on several levels simultaneously. It is well know that solving a standard Poisson problem on locally refined grids by first solving the problem on coarse

grids then on refined regions using interpolated boundary data results in a global error on the order of the coarse grid error. In essence, it destroys the benefit of local grid refinement. Since no careful analysis has yet appeared for a method like the one in [1], it is not clear what the effect of temporal refinement for incompressible flow simulations is in general. It is possible that the temporal smoothness of the pressure correction for many problems might keep the interpolation errors at an acceptably low level when temporal refinement is employed.

The issue of how viscous terms are incorporated into a method also has an impact on the appropriateness of temporal refinement. In a method like the vorticity-stream function method described in Chapter 3, the viscous terms can be handled in an explicit manner. For problems with little viscosity, no serious restriction on the time step is introduced by this. Explicit treatment allows easy inclusion of the viscous terms in the refined grid updates even if temporal refinement is being used. Conversely, if the viscous terms are handled in an implicit manner as in the projection method used in this thesis, then it is not clear what the effect of solving the implicit equations locally would be. The projection method presented here is only implemented for inviscid flow, but the author hopes to investigate the issues of temporal refinement, projections, and viscous terms in future work.

A single step of the refined grid projection method closely mimics the single grid method presented in Section 2.5. The advective derivatives are computed sequentially on each grid in each level. At the boundaries of child grids, difference operators are modified to use both parent and child grid data. The MAC projection and approximate projection steps in the single grid case are performed on the entire grid structure. The following is an outline for one time step. Each item is described in a subsequent subsection.

1. Given cell-centered values of U^n and $\nabla p^{n-1/2}$, velocity values for a single row of boundary cells around each child grid are set so that difference operators can be used at coarse-fine interfaces.

2. The Taylor series extrapolation procedure from the single grid case is used to compute provisional time-centered edge velocities.

3. The provisional edge velocities are corrected with a MAC projection to enforce incompressibility.

4. The time-centered edge velocities are differenced to compute an approximation to the

advective terms which in turn is used to compute a provisional update of the velocities.

5. An approximate projection is performed on the provisional velocity update to yield U^{n+1} and $\nabla p^{n+1/2}$.

4.2.1 Interpolation of Boundary Values

As mentioned at the beginning of this chapter, in the implementation of the refined grid projection method, it is convenient to create extra rows of cells around child grids so that finite difference operators near grid boundaries can be defined with the same stencil as in grid interiors. One can view the values in these boundary cells in the same manner as boundary cells outside a physical domain; their value is only significant in that it allows finite difference operators to take the same form at grid boundaries as they do in the interior.

The procedure for interpolating the values for boundary cells in this method is motivated by the form of the finite difference approximation to the normal derivative across the boundary. A single row of boundary cells around a grid is set so that a simple centered difference at the child grid boundary will give a second-order approximation to the normal derivative at the boundary. For example, consider the section of a refined grid shown in figure 4.2. The circle in the figure marks the location of the center of a boundary cell of the fine grid, and the solid dots mark the center of actual grid cells. Let the value of some function ψ at each of the marked locations be denoted by the corresponding subscripts ψ_A,ψ_B, et cetera. 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. One could compute some value for ψ_F and then approximate the value of the derivative with

$$\frac{\psi_D - \psi_F}{h} \tag{4.1}$$

where h is the grid spacing of the fine grid.

It is possible to define a value of ψ_F that only depends on coarse grid values of ψ, so that the difference equation (4.1) only depends on one value from the interior of the child grid, namely ψ_D. The disadvantage of this method is that in order to produce a second-order difference operator, the coarse grid points on which the the value of ψ_F depends must be spread around the point F in both the horizontal and vertical directions. Since boundary values for a child grid can lie next to the grid boundary of its parent, in some situations there may not be enough coarse grid points available to handle a wide coarse grid stencil.

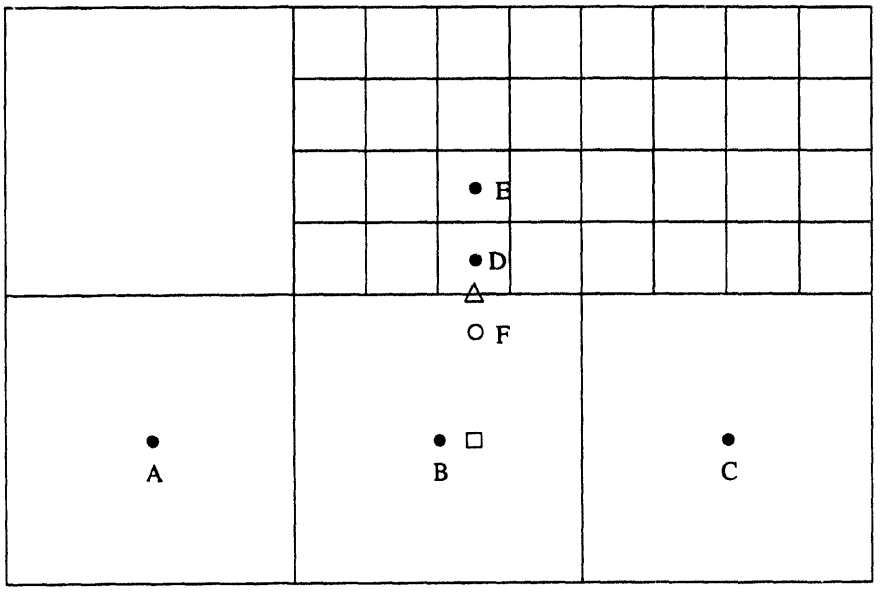


Figure 4.2: Normal derivative or interpolation stencil with $r = 4$.

To avoid potential problems with wide stencils, the derivative operator (and hence the interpolation stencil) used in this thesis depend only 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. Additional values from the interior of child grids are used to yield the second-order operator. In all, five points are used in the interpolation or derivative operator stencil—three coarse and two fine.

Figure 4.2 shows the stencil for the case $r = 4$. The formula for the derivative of ψ normal to the boundary at the point marked by the triangle is given by

$$\frac{\frac{1}{5}\psi_E + \frac{1}{3}\psi_D + \frac{1}{20}\psi_A - \frac{1}{2}\psi_B - \frac{1}{12}\psi_C}{h} \tag{4.2}$$

where h is the cell size of the fine region. This is equivalent to

$$\frac{\psi_D - \psi_F}{h} \tag{4.3}$$

where

$$\psi_F = -\frac{1}{5}\psi_E + \frac{2}{3}\psi_D - \frac{1}{20}\psi_A + \frac{1}{2}\psi_B + \frac{1}{12}\psi_C. \tag{4.4}$$

The form of the derivative of ψ is also equivalent to a second-order approximation which depends on the 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 figure 4.2 and the value of which is interpolated from the coarse grid values ψ_A, ψ_B, and ψ_C.

The method of interpolation presented above is used for all child-grid boundary value interpolation. In Section 4.2.3 it will also be used to define the form of the projection operator near a child grid boundary.

Whenever a child grid is contiguous to another grid in the same level, boundary cells that correspond to interior cells of the neighboring grid can simply be given the value of the corresponding neighbor cells. This is equivalent to ignoring the boundary between the two contiguous grids and using the usual difference operator.

The practice of setting boundary values for grids based on the form of difference operators across coarse-fine interfaces is analogous to what is done at the boundary of physical domains where boundary data do not represent true physical values but only serve to enable finite difference operators to take a convenient form at boundaries. Since boundary cells around child grids are usually in the physical domain, it is possible to assign values of physical variables to these cells through interpolation of surrounding values. Finite difference operators could be defined simply by using these interpolated values with the usual operator stencil to compute derivatives at coarse-fine interfaces. The difference between these two approaches is that in the method defined here, it is not always possible to assign a unique value to grid cells that are at once boundary cells for more than one grid. Consider, for example, the section of grid shown in figure 4.3. The fine grid boundary cell marked by the dotted edges is used in the calculation of the derivatives at both of the cell edges marked by the squares. The interpolation stencil for each of these derivative operators uses different points, hence a different value for the boundary cell must be used for each.

4.2.2 Calculation of Provisional Time-Centered Edge Values

Recall from Section 2.5 that Taylor series expansions are used to extrapolate cell-centered velocities in time and space to yield provisional time-centered edge velocities. This is done in two steps. First, extrapolation is done from each cell center to yield values of the velocities at each cell edge. This is accomplished by computing approximations to all the first-order terms in the Taylor series expansions given in equations (2.32) and (2.32) except the pressure term. The pressure term is later approximated with a MAC projection. The second step is to use a procedure similar to solving a Riemann problem at each cell

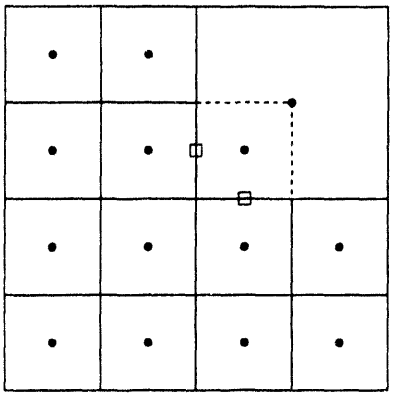


Figure 4.3: Ambiguous boundary value location.

edge. This produces a single value from the two values that result from the extrapolation procedure.

In the refined grid version, the extrapolation to cell edges proceeds in a similar manner to the single grid method. After a single row of boundary cells around child grids is given velocity values by the procedure described above, difference approximations to the derivatives are computed in exactly the same manner as in the single grid method to produce cell-edge values $\tilde{U}$. For example, extrapolation to the top edge of a cell has the form

$$\tilde{U}^T_{i,j-1/2} = U^n_{i,j} + \frac{1}{2}[1 - s_L \tfrac{\Delta t}{h} v^n_{i,j}]\delta^4_y(U^n_{i,j}) + \tfrac{\Delta t}{2}[-(uD^u_x(U))_{i,j} + \nu L^5(U^n)_{i,j}]. \qquad (4.5)$$

Since evaluating the fourth-order slope approximations δ^4 defined in equation (2.36) near boundaries would require two rows of boundary cells, the second order approximation δ^1 is used at cells next to child grid boundaries. (δ^1 is defined in equation (2.35).)

Extrapolation to grid edges from cell centers produces two values of $\tilde{U}$ at all cell edges in the interior of a grid. However, 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 $\tilde{U}$ is computed by the above procedure. The second value of $\tilde{U}$, which corresponds to extrapolation to cell edges at the grid boundary from cells just outside the grid, is interpolated from parent grid values of $\tilde{U}$. Third-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.

After two values of $\tilde{U}$ are computed at each cell edge, a single provisional time-centered value for each velocity at each edge is selected by the process described by equation (2.5.2). Once a single value of $\tilde{U}$ is computed at each cell edge on all levels, parent grid values of the provisional velocities are then assigned the average of child values in preparation for the subsequent MAC projection.

4.2.3 The Refined Grid MAC Projection

The values of the time-centered provisional velocities $\tilde{U}$ are calculated without incorporating the pressure term in the Taylor series extrapolation. To correct for the omission of the pressure term, a MAC projection is performed on the provisional values.

Unlike the evaluation of the advective terms in the Navier-Stokes equations, the projection is inherently non-local and must be performed on the entire computational domain. Nevertheless, it is possible to utilize a relaxation scheme based on the standard multigrid machinery that performs relaxations on individual grids. Such a method is described in detail in Section 4.3

To begin, the procedure for defining the projection operator detailed in Section 2.2 is extended to refined grids. Recall that given a discrete divergence operator D on the refined grid, a discrete gradient operator G can be constructed from D using an adjointness condition:

$$\langle G(\phi), U\rangle_V = -\langle \phi, D(U)\rangle_S.$$

The inner products for the single grid case are the usual vector inner products (scaled for cell size), so that $G = -D^T$. In the refined grid case, the inner products will have to be specified more carefully in order to produce a consistent form for G. It will also be shown that the adjointness condition must be relaxed if G is to be more than zeroth-order accurate at coarse-fine grid interfaces.

The discrete divergence operator for the refined grid structure is defined on each grid section in exactly the same manner as in the single grid case. The divergence is defined at the center of each cell and is the sum of the centered differences of cell-edge velocities. For coarse cell edges that are also fine grid cell edges, the value of the velocity used in the coarse grid divergence is the average of the fine grid velocities. 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. Second, a

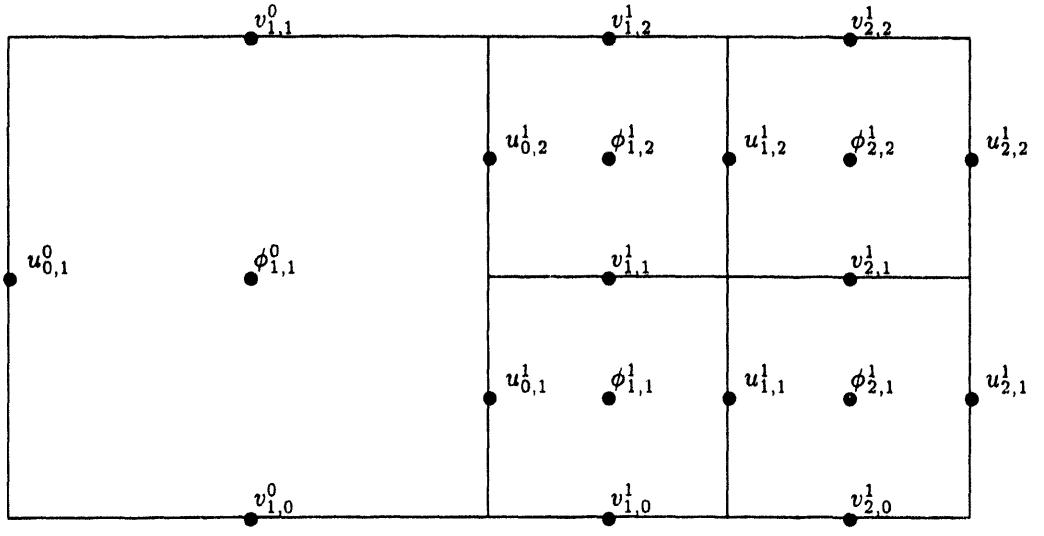


Figure 4.4: Partial grid example with $r = 2$.

refined grid version of the discrete divergence theorem 2.3.1 holds. 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.

The form of the scalar inner product for refined grids is also a straightforward extension of the single grid case. The 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 exactly the rectangle rule for the approximation of $\int_\Omega \psi(\bar{x})\phi(\bar{x})\, d\bar{x}$.

To illustrate, consider $\langle \phi, D(U) \rangle_S$ for the small section of refined grid shown in figure 4.4. Let h be the grid spacing on the coarse grid. For this example

$$\begin{aligned} \langle \phi, D(U) \rangle_S &= \phi^0_{1,1} \left(\frac{\frac{u^1_{0,1}+u^1_{0,2}}{2} - u^0_{0,1}}{h} + \frac{v^0_{1,1} - v^0_{1,0}}{h} \right) h^2 + \\ &\quad \sum_{i,j=1}^{2} \phi^1_{i,j} \left(\frac{u^1_{i,j} - u^1_{i-1,j}}{\frac{h}{2}} + \frac{v^1_{i,j} - v^1_{i,j-1}}{\frac{h}{2}} \right) \frac{h^2}{4}. \end{aligned} \tag{4.6}$$

In order to determine the correct form of the vector inner product and subsequently the form of G, consider first the simpler case in which grid refinement is only done in one dimension.

Figure 4.5 shows a section of a grid which is only refined in the horizontal direction.

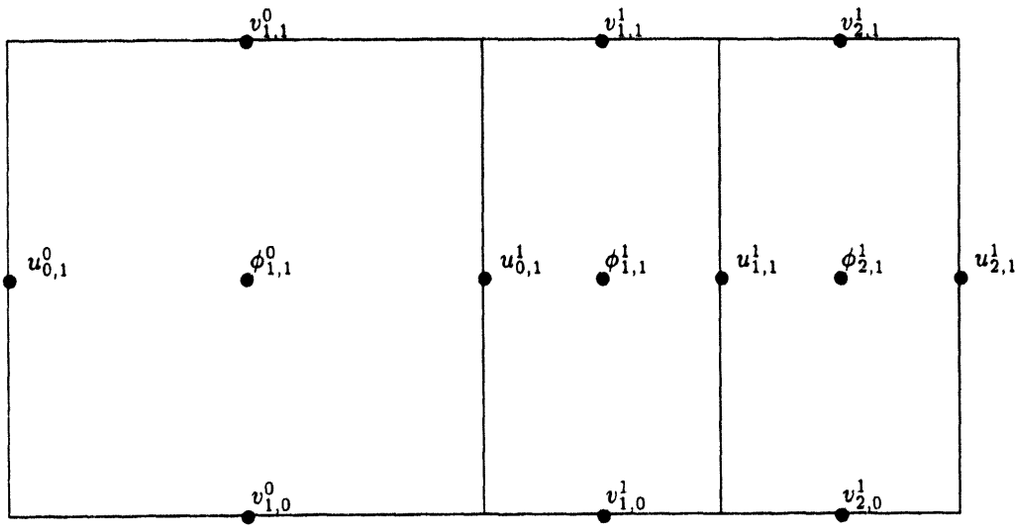


Figure 4.5: Partial grid example with refinement in one dimension only.

Here the form of $\langle \phi, D(U) \rangle_S$ becomes

$$\begin{aligned} \langle \phi, D(U) \rangle_S &= \phi^0_{1,1} \left(\frac{u^1_{0,1} - u^0_{0,1}}{h} + \frac{v^0_{1,1} - v^0_{1,0}}{h} \right) (h^2) \\ &+ \sum_{i=1}^{2} \phi^1_{i,1} \left(\frac{u^1_{i,1} - u^1_{i-1,1}}{\frac{h}{2}} + \frac{v^1_{i,1} - v^1_{i,0}}{h} \right) \left(\frac{h}{2} \right) h. \end{aligned} \tag{4.7}$$

The question of interest is: What form will the operator G take at the coarse-fine interface? If the adjointness condition is to hold, $G_1(\phi)^1_{0,1}$ must depend exclusively on the values of $\phi^0_{1,1}$ and $\phi^1_{1,1}$ since only these terms appear paired with $u^1_{0,1}$ in the definition of $\langle \phi, D(U) \rangle_S$. It is immediately apparent that the approximation to the gradient will at best be first-order accurate since the two points on which it depends are not centered about the evaluation point. In order for the approximation to be first order, it must take the form

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

By rearranging terms in (4.7) it is clear that the term corresponding to $G_1(\phi)^1_{0,}$ in the vector inner product $\langle U, G(\phi) \rangle_V$ must be

$$u^1_{0,1} \left(\frac{\phi^1_{1,1} - \phi^0_{1,1}}{\frac{3h}{4}} \right) \frac{3h}{4} h. \tag{4.8}$$

This form of the vector and scalar inner products defines G to be the usual second-order difference in grid interiors, and a first-order approximation similar to equation (4.8) at coarse-fine interfaces.

The reduction to first order of the gradient in the above case is worsened when one refines in both spatial dimensions. Returning to figure 4.4, the same argument indicates that the form of $G_1(\phi)^1_{0,j}$ must only depend on values of $\phi^0_{1,1}$ and $\phi^1_{1,j}$. Besides not being horizontally centered, the two values used in each derivative are also not centered in the vertical direction. The best one can produce is gradients that take the form:

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

which are only exact for functions linear in x and constant in y. The form of the terms from the vector inner product are again arrived at by matching terms in equation (4.6), and are given by

$$u^1_{0,j}\left(\frac{\phi^1_{1,j} - \phi^0_{1,1}}{\frac{3h}{4}}\right)\frac{3h}{4}\frac{h}{2}. \tag{4.9}$$

It is apparent that the terms in the vector inner product should be scaled by the product of the edge length and the distance between adjacent cell centers. As with the scalar case, this form is the rectangle rule for a continuous integral over rectangles arranged as in figure 4.6. The dotted lines in this figure are edges of the rectangles the areas of which are the scaling factors for the terms located at the square symbols.

Away from grid edges, the terms of the inner product are scaled exactly as in the single grid case, and subsequently G takes the usual centered difference form in the interior of grids.

The coarse grid value of the gradient at interfaces is defined as the average of the fine grid gradients as is the case for velocities. For the example shown in figure 4.4, averaging gives

$$\begin{aligned} G_1(\phi)^0_{1,1} &= \frac{\frac{\phi^1_{1,2} - \phi^0_{1,1}}{\frac{3h}{4}} + \frac{\phi^1_{1,1} - \phi^0_{1,1}}{\frac{3h}{4}}}{2} \\ &= \frac{\frac{\phi^1_{1,2} + \phi^1_{1,1}}{2} - \phi^0_{1,1}}{\frac{3h}{4}}. \end{aligned} \tag{4.10}$$

When viewed as a coarse grid operator, the gradient at the interface is first-order accurate, but this is really no consolation. Note that the form of $G_1(\phi)^0_{1,1}$ cannot be written as

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

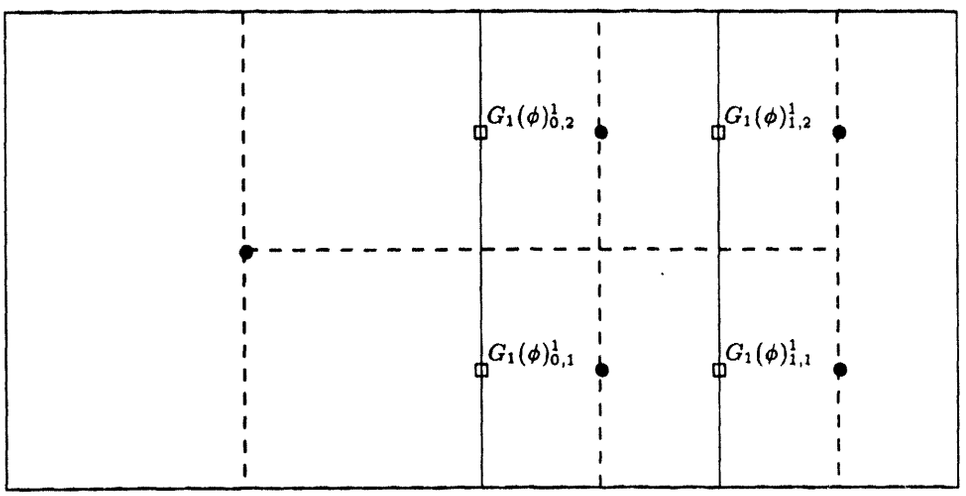


Figure 4.6: Rectangles the areas of which give the scaling factors for the vector inner product.

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 loss of accuracy for the gradient at grid interfaces is clearly unacceptable. Two remedies to the problem exist. The form of the divergence can be changed to one the adjoint of which is a more accurate gradient, or the adjointness condition at the edge of grids can be ignored. A divergence operator that depends on additional velocity values will define a gradient through the adjoint relationship 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.

The alternative, relaxing the adjointness condition at the boundaries, is the method by which a higher-order gradient is defined here. The gradient at grid edges is defined in terms of child grid boundary values as is done in Section 4.2.1. For example, consider the

gradient that results from the adjointness condition in the example shown in figure 4.4.

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

Values of ϕ for the first row of boundary cells around the fine grid region can be defined so that the same form of the gradient can be used at the grid edges as in the interior. A simple manipulation shows that

$$G_1(\phi)^1_{0,j} = \frac{\phi^1_{1,j} - \phi^1_{0,j}}{\frac{h}{2}}.$$

where the boundary value $\phi^1_{0,j}$ is computed by

$$\phi^1_{0,j} = \frac{2}{3}\phi^0_{1,1} + \frac{1}{3}\phi^1_{1,j}.$$

This indicates that the correct interpolation scheme for setting the boundary value for this form of the gradient is to use linear interpolation between the coarse and fine grid values. These values are not centered in the vertical direction which is exactly the cause of the reduction to zeroth-order accuracy for this form of the gradient. Any form of the gradient can be described in this manner and, likewise, any interpolation scheme for boundary values defines a gradient.

The method used for interpolating boundary values for the gradient is exactly that which is used to interpolate cell-centered velocities during the evaluation of the advective derivatives. Details of the interpolation stencil are presented in Section 4.2.1. The interpolation is third-order accurate which results in second-order accuracy of the gradient at grid edges.

Recall that the adjointness condition is used in the single grid method to show that the projection is well defined and norm reducing. Consequently, a gradient that is not adjoint to the divergence could lead to an ill-posed projection. It should be noted however that violating the adjointness condition is not the same as defining an approximate projection. The velocity field resulting from the projection will still be discretely divergence free if the Laplacian in the projection Poisson problem is the composition of D and G (and a solution to this problem exists). In the absence of the adjointness condition, other methods must be used to show that the projection is well posed and norm reducing. It can be shown directly for very simple refined grids that the MAC projection described above is well posed, although a general proof of this for an arbitrary fine grid structure has not been completed.

Computational experience suggests that for a variety of problems the MAC projection is well posed. No examples of a Poisson problem without a solution were encountered during the numerical experiments contained in this thesis.

4.2.4 Evaluating the Advective Terms

Once the MAC projection has been performed on the time-centered cell-edge values, the advective derivatives can be computed exactly as in equation (2.29), suppressing the $n + 1/2$ superscript

$$\begin{aligned}(uU_x + vU_y)_{i,j} &= \frac{(u_{i+1/2,j} + u_{i-1/2,j})}{2}\frac{(U_{i+1/2,j} - U_{i-1/2,j})}{h} \\ &+ \frac{(v_{i,j+1/2} + v_{i,j-1/2})}{2}\frac{(U_{i,j+1/2} - U_{i,j-1/2})}{h}.\end{aligned} \tag{4.11}$$

The values of the advective derivatives are then used to compute provisional updates to the velocity on each grid level from the formula: (assuming $\nu = 0$)

$$\frac{U^* - U^n}{\Delta t} + \nabla p^{n-1/2} = -[(U \cdot \nabla)U]^{n+1/2}. \tag{4.12}$$

4.2.5 The Approximate Projection

The final step in the method is to decompose the provisional update by an approximate projection to yield an update to the velocity and pressure terms. As in the single grid method, an approximate projection $\tilde{P}$ based on a MAC projection is used for the decomposition. The form of the update for the velocity and pressure are

$$\frac{U^{n+1} - U^n}{\Delta t} = \tilde{P}(\frac{U^* - U^n}{\Delta t})$$

and

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

The form of the approximate projection is again very similar to what is used in the single grid method. A MAC projection is performed on cell-edge values that are interpolated from the cell-centered values of the provisional update. The cell-edge gradient that results from the projection is then interpolated back to cell centers to yield $\nabla p^{n+1/2}$ and the updated velocity U^{n+1}.

Interpolation is done in exactly the same manner as in the single grid approximate projection discussed in Chapter 2.5.3. When cell-centered values are being interpolated to

the edges of child grids, the additional value needed for the interpolation stencil is computed from coarse grid values using the same interpolation scheme developed in Section 4.2.1. In practice, a single row of boundary cells are set by interpolation so that cell-edge values at the edge of child grids can be computed with the interpolation scheme that is used in the grid interior.

4.3 A Multigrid Method for the Refined Grid MAC Projection

The multigrid algorithm discussed in Section 2.6 can be extended to refined grids. In [3], Almgren presented a multigrid method for solving the Poisson problem associated with the method of local corrections. 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 [35]. Two important features of the method presented in this thesis make it a significant improvement over the methods in [1] and [35]. 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 resulting velocity 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. Details of how intermediate grids are used and how this procedure differs from earlier multigrid algorithms are discussed in Section 4.3.4.

The multigrid algorithm discussed in Section 2.6 has a natural extension to a refined grid structure with a refinement factor of two. The basic idea is that relaxation is performed 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. 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 the previous section. The details of the relaxation, averaging, and interpolation operators will be given after an outline of a single V-cycle on the refined grid structure is

presented. Modifications of the method to handle refinement factors which are greater than two are covered in Section 4.3.4.

Given a refined mesh with levels 0 through $lmax$, suppose that an initial guess to the solution $\phi^{l,k}$ is available on each grid. The following uses the operator notation of Section 2.6 to describe a single multigrid sweep of the entire refined grid structure when the refinement ratio between levels is two in each case:

- Calculate the residual $\rho^{l,k} = R^M(\phi^{l,k}, D(U)^{l,k}, h^k)$ on each grid of each level including those regions in which child grids exist.
- Correct the residuals for coarse grid cells that border regions of refinement.
- For $l = lmax$,1, step -1
  1. Perform relaxation on residual problems, $\xi^{l,k} = G^M(\xi^{l,k}, \rho^{l,k}, h^l)$. At child grid boundaries the parent grid values used in the relaxation operators are set to zero.
  2. Update the value of $\rho^{l-1,k}$ at coarse cell locations that border refined regions. The new values of the residual should be consistent with the updated solution $\phi^{l,k} + \xi^{l,k}$ on refined grids.
  3. Form new residual problems $\tilde{\rho}^{l,k} = R^M(\xi^{l,k}, \rho^{l,k}, h^k)$.
  4. Average $\tilde{\rho}^{l,k}$ onto the parent grids at level $l-1$ which completes the computation of $\rho^{l-1,k}$.
- Do one multigrid V-cycle for the residual problem on the coarsest level. $\xi^0 = \mathrm{MGV}(\xi^0, \rho^0, h^0)$ Note that this problem could instead be solved completely, but computational experiments show that performing a single V-cycle is more efficient than completely solving this residual problem.
- For $l = 1, lmax$
  1. Interpolate the solution of the residual problems from parent grids and add to residual problem solutions; $\xi^{l,k} = \xi^{l,k} + I^M(\xi^{l,k+1})$.
  2. Relax again to improve solution $\xi^{l,k}$; $\xi^{l,k} = G^M(\xi^{l,k}, \rho^{l,k}, h^k)$.
  3. Add $\xi^{l,k}$ to $\phi^{l,k}$.

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^{-8}. A single application of the above procedure on the entire grid structure typically reduces the norm of the residual by a factor of five to ten which is approximately the same size reduction as for a V-cycle on a single grid problem.

4.3.1 Averaging the Residuals

The residual problem that is constructed on each grid must be averaged onto its parent grid, and the form of the divergence operator determines how this is done. The residual operator R^M has the form

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

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

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

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. This seemingly trivial point has a significant consequences: The composite residual problems that are formed in the refined grid multigrid procedure are Poisson problems that are specified to have homogeneous Neumann boundary conditions. Hence, there is a solvability condition on the right-hand side of these problems, namely

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

But since the right-hand sides of composite residual problems are always in the form of a divergence, the solvability condition for the Poisson equation is automatically satisfied by theorem 2.3.1. A different method of averaging residuals—even a more accurate method, could lead to a violation of the solvability condition for composite residual problems.

The form of the residual at grid boundaries depends only on the form of the gradient there. In the implementation, boundary cells are set at grid edges so that the residual can be computed with the same stencil as in the grid interior. Since the residual depends on the form of DG, values for the boundary cells are set in the same manner as for the gradient operator.

4.3.2 Interpolating Corrections

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

In the implementation on refined grids, 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.

4.3.3 Relaxation on Refined Grids

As in the single grid method, the relaxation scheme used on individual child grids is Gauss-Seidel with red-black ordering. On child grids, the relaxation operator 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 take the same form as in equation (2.47)

$$G^M(\psi)_{i,j} = \frac{\psi_{i+1,j} + \psi_{i-1,j} + \psi_{i,j+1} + \psi_{i,j-1} - h^2 f_{i,j}}{4}. \tag{4.13}$$

Since boundary values depend on interior cells, the red and black subgrids are coupled at the boundary and, technically, a change in either grid will change the correct value in boundary cells. In the refined grid multigrid 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 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.

4.3.4 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. A multigrid method similar to the one presented here appears in [1] which, for refinement factors greater than two, replaces 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 grid 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. Divergences or residual problems never need to be adjusted to satisfy the solvability constraints associated with the Neumann problem as is done in [35].

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 again, and a new residual problem is computed which is then again averaged to the next level. This procedure continues until the next coarsest physical level is reached.

The form of a residual problem on a physical grid is simply

$$DG(\xi) = \rho$$

where the residual ρ is

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

When intermediate levels are used, the residual problem calculated on these levels is actually the residual of a residual problem or

$$\tilde{\rho} = \rho - DG(\xi).$$

But using the definition of ρ, this is simply

$$\tilde{\rho} = D(U) - DG(\phi) - DG(\xi) = D(U - G(\phi + \xi)). \tag{4.14}$$

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 solution of the residual problem ξ.

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. It is also apparent from equation (4.14) that the right-hand side of the composite residual problem on each grid level is always in the form of a divergence. This ensures that composite residual problems satisfy the solvability condition for the discrete Poisson problem with Neumann boundary conditions. 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.

Chapter 5

A Refined Grid Vorticity-Stream Function Method

The vorticity-stream function method for a single grid described in Chapter 3 was designed so that it could easily be extended to refined grids using the machinery developed for the refined grid projection of Chapter 4. There are two main purposes for developing the refined grid vorticity method. First, the method provides a way to validate the results from the refined grid projection method. If these two methods, which are based on different forms of the equations of motion, can be shown to converge numerically to the same solution for the same problem, it is difficult to argue that this solution is incorrect. Second, although the method here uses simplified boundary conditions, it is hoped that the machinery developed for the vorticity method can be used in more complicated vorticity-based methods that allow viscous flows around physical boundaries.

The refined grid method described here shares the two important qualities of the single grid method: It is conservative and free-stream preserving. It is based on the same form of the vorticity equation of motion:

$$\omega_t = -\nabla \cdot (\omega U) \tag{5.1}$$

which is discretized using a conservative MAC divergence.

The temporal discretization of the refined grid method is the same as the single grid method: A third order TVD Runge-Kutta scheme is used. This method is described by equations (3.17). For each Runge-Kutta substep, the following steps are performed:

1. Staggered grid values of the velocities are computed by differencing the cell-centered values of the stream function. This produces a staggered grid velocity field that is discretely divergence free in grid interiors but not necessarily zero near coarse-fine interfaces because of the asymmetry of the difference operator there.

2. To make the divergence of the staggered velocity field zero in all cells of the refined grid structure, a MAC projection is performed on the edge velocities.

3. The cell-centered vorticity values are interpolated to cell edges using the upwind interpolation scheme discussed in Section 3.1.

4. A conservative difference on edge quantities is performed to yield the advective derivatives.

5. The vorticity-stream function Poisson equation is solved to yield updated values of the stream function and velocities.

Each of the above steps closely resembles the single grid method of Chapter 3 and is described in further detail below.

5.1 Computation of Staggered Velocities

Recall from the single grid method that equation (5.1) is discretized using a MAC divergence operator D applied to cell edge quantities

$$D(\omega U) = \frac{(\hat{u}\hat{\omega}_1)_{i,j} - (\hat{u}\hat{\omega}_1)_{i-1,j} + (\hat{v}\hat{\omega}_2)_{i,j} - (\hat{v}\hat{\omega}_2)_{i,j-1}}{h}. \tag{5.2}$$

The values of the edge velocities $\hat{U}$ are computed by averaging values of u and v, and the resulting values are such that $D(\hat{U}) = 0$. The fact that divergence of $\hat{U}$ is zero is a consequence of the fact that the velocities are computed from the stream function using symmetric differences which cause the terms in the divergence to cancel.

In this refined grid method, cell-edge velocities $\hat{U}$ are computed by averaging nearest-neighbor velocities as is done in the single grid method. At child grid boundaries, values of the velocities in boundary cells are computed using interpolated values of the stream function and these velocity values are then used to compute $\hat{U}$ on the grid boundary. This procedure yields values of $\hat{U}$ in the interior of grids that are discretely divergence free. At coarse-fine grid interfaces however, the difference operators used to compute the velocities

U from the stream function are not symmetric, and hence the terms in the divergence of $\hat{U}$ do not quite cancel. In order to ensure that the overall refined grid method is free-stream preserving, the MAC projection described in 4.2.3 is applied to the computed values $\hat{U}$ so that they have a MAC divergence of zero. In practice, it requires about four to six iterations of the multigrid procedure described in 4.3 to reduce the divergence of $\hat{U}$ to be of the order of 10^{-9} for the problems in this thesis.

5.2 The Interpolation of Vorticity

Once divergence-free staggered values of $\hat{U}$ are computed, cell-edge values $\hat{W} = (\omega_1, \omega_2)$ of the vorticity must be interpolated from cell-centered values. The adaptive interpolation scheme described in Section 3.2 is employed with a slight modification at the boundaries of child grids. At these boundaries, values of the vorticity are interpolated from coarse grid values for a single row of boundary cells. Only these boundary values, along with interior cells, are available for use in the interpolation procedure. This effectively limits the number of different interpolation stencils that can be used at the boundary to two. For example, consider the interpolation for the edge value $\hat{\omega}_{0,j}$ (see figure 5.1). For rth order interpolation, $r+1$ values will be used for the interpolation and, in this case, they will be either $\omega_{0,j}, \omega_{1,j}, \ldots \omega_{r,j}$ or $\omega_{1,j}, \omega_{2,j}, \ldots \omega_{r+1,j}$. Note that this does not reduce the order of accuracy of the interpolation at the boundary, it simply changes which points the interpolation stencil includes. As in the single grid method, $r = 3$ for the interpolation order.

For child grids that are contiguous to another grid of the same refinement, the values of the neighboring grid are used in the interpolation procedure, and no modification of stencil selection is required.

5.3 Computation of Advective Derivative

The approximation to the advective derivative given by equation (5.2) is computed on each cell in the refined grid structure using the previously computed cell-edge values of $\hat{U}$ and $\hat{W} = (\omega_1, \omega_2)$. Before the difference approximation is taken, the value of the product $\hat{U}\hat{W}_{i,j}$ on each coarse grid cell edge that also corresponds to fine grid values is assigned the average of those fine grid values of the product. This ensures that the method will be

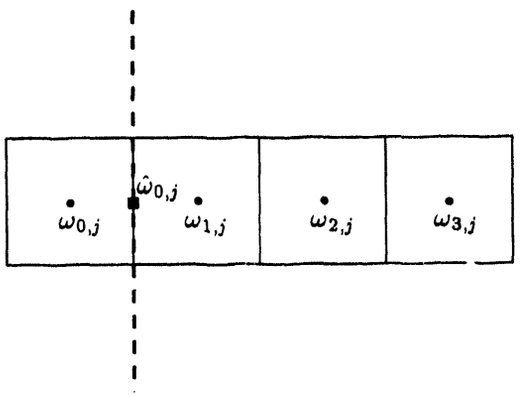


Figure 5.1: Interpolation stencil for vorticity at a grid edge.

conservative since the discrete form of the divergence theorem (2.3.1) holds for the refined grid structure.

5.4 Solution of the Vorticity-Stream Function Equation

Recall from the single grid method that at each Runge-Kutta substep, the update to the stream function $\delta\psi$ must be computed from the update of the vorticity $\delta\omega$ by

$$\Delta(\delta\psi) = -\delta\omega. \tag{5.3}$$

The multigrid method described in the Section 4.3 is used to solve equation (5.3). For the patch problems studied in this thesis, the boundary conditions on the computational domain are imposed on the velocities, and these boundary conditions do not change in time. Therefore, the correct boundary conditions for equation (5.3) are homogeneous Neumann. Again, the solvability condition for this problem is guaranteed by the fact that the method is conservative; the total sum of the update $\delta\omega$ is zero.

Once the stream function has been computed at each Runge-Kutta substep, the updated staggered grid velocities are computed by

$$v_{i,j} = -\frac{\psi_{i+1,j} - \psi_{i,j}}{h}$$
$$u_{i,j} = \frac{\psi_{i,j+1} - \psi_{i,j}}{h}.$$

Coarse grid values of the velocities are then assigned the average of fine grid values where child grids exist.

Chapter 6

Numerical Results

6.1 Single Grid Convergence Results

6.1.1 Convergence of the Single Grid Projection Method

In order to validate the second-order convergence rate of the projection method, two example problems are run and a convergence study is done for both. The first problem is one with a stationary solution in a periodic domain. The initial conditions are given by

$$\begin{aligned} u(x,y) &= -\cos(2m\pi x)\sin(2m\pi y) \\ v(x,y) &= \sin(2m\pi x)\cos(2m\pi y), \end{aligned} \tag{6.1}$$

where m is some integer and the domain is the periodic unit square. The general solution at time t for these initial conditions and given viscosity ν is

$$\begin{aligned} u(x,y,t) &= -\cos(2m\pi x)\sin(2m\pi y)e^{-2\nu(2m\pi)^2 t} \\ v(x,y,t) &= \sin(2m\pi x)\cos(2m\pi y)e^{-2\nu(2m\pi)^2 t}. \end{aligned}$$

In the absence of viscosity, the solution is constant in time.

Since the exact solution of this problem is known, the error of a computed solution can be calculated directly. Table 6.1 contains convergence results for these initial conditions. The same problem is run on grids with grid spacing ranging from 1/32 to 1/256, and error information is gathered at the end of each run. For this example, the number of modes in the initial condition is two, i.e. $m = 2$ in equation (6.1). The end time for each run is 1.0, the CFL number is 0.9, and $\nu = 0$. Table 6.1 shows the L_2 and infinity norm of the error

Error of	32	rate	64	rate	128	rate	256
$\|u\|_2$	1.86E-2	2.02	4.58E-3	2.24	9.69E-4	2.13	2.21E-4
$\|u\|_\infty$	4.17E-2	1.45	1.52E-2	1.82	4.29E-3	1.88	1.16E-3

Table 6.1: Convergence rates for the projection method on the stationary problem.

in the u-component of the velocity (since the problem is symmetric this is the same as the error in the v velocity). Here the L_2 norm for u on the grid is defined as

$$\|u\|_2 = \sqrt{\frac{\sum_{i,j} u_{i,j}}{nx \cdot ny}}.$$

The errors are given in the columns labeled by the number of grid points in each dimension for the run (e.g. "32"). Convergence rates are computed by taking the $\log_2$ of the ratio of errors between grids the cell sizes of which differ by a factor of two and are listed in the columns labeled "rate". It is apparent from the data in table 6.1 that the method converges to the exact solution at a second-order rate for this problem.

The second example is that of a double shear layer in a periodic domain. The initial conditions are given by

$$\begin{aligned} u(x,y) &= \begin{cases} tanh(\rho(y-0.25)), & \text{for } y \le 0.5 \\ tanh(\rho(0.75-y)), & \text{for } y > 0.5 \end{cases} \\ v(x,y) &= \delta sin(2\pi x), \end{aligned} \tag{6.2}$$

where ρ is the shear layer width parameter, and δ is the perturbation size. Since the initial shear layers are unstable, the small perturbation in v eventually causes the layers to roll up into strong vortical structures.

The exact solution for these initial condition is unknown, so convergence rates are calculated by a procedure similar to Richardson extrapolation which compares a solution computed with a given grid cell size h to one computed with cells of size $h/2$. If one assumes that the error in a solution at time t can be adequately represented by $h^r\xi(x,y,t)$ where r is the rate of convergence and ξ does not depend on h, then the difference $D^h_{h/2}$ between a solution computed with grid size h and one with $h/2$ is given by

$$D^h_{h/2} = h^r(1-1/2^r)\xi(x,y,t).$$

If one then computes $D^{h/2}_{h/4}$, then $r = \log_2(D^h_{h/2}/D^{h/2}_{h/4})$. To compute $D^h_{h/2}$, an average of the a solution with grid spacing $h/2$ is computed, and $D^h_{h/2}$ is defined as the L_2 norm of the

Time	32-64	rate	64-128	rate	128-256
0.4	1.92E-2	1.95	4.95E-3	2.18	1.09E-3
0.8	7.15E-2	1.86	1.97E-2	2.15	4.46E-3
1.2	7.85E-2	1.63	2.53E-2	1.96	6.50E-3

Table 6.2: Convergence rates for the projection method for the u-component of the velocity on the inviscid shear layer problem.

difference between this average solution and the solution with resolution h. Specifically, if u^c is a solution on an $nx \times ny$ grid with grid spacing h, and u^f is a solution on a $2nx \times 2ny$ grid with grid spacing $h/2$, then

$$(D^h_{h/2})^2 = \frac{\sum_{i,j}(u^c(i,j) - \frac{u^f(2i,2j)+u^f(2i-1,2j)+u^f(2i,2j-1)+u^f(2i-1,2j-1)}{4})}{nx \cdot ny}.$$

Table 6.2 gives convergence rates for grid cell sizes ranging from 1/32 to 1/256 computed at times 0.4, 0.8, and 1.2. The quantities $D^h_{h/2}$ are given in the columns labeled by the the number of grid points in each dimension of the coarse and fine runs. For example, $D^{1/32}_{1/64}$ is given in the column labeled "32-64". For these runs, $\rho = 30$, $\delta = 0.05$, $\nu = 0$, and the CFL number is 0.9. Contour plots of the vorticity distribution and u-component of the velocity at time 1.2 generated by the run with $h = 1/256$ are shown in the left side of figure 6.1. The top left picture in figure 6.1 is the vorticity field for the projection method and the bottom left is the u-component of the velocity. The vorticity for these plots is computed from the velocities using standard fourth-order finite differences. The right side of figure 6.1 displays the computed solutions from the vorticity-stream function for comparison.

6.1.2 Convergence of the Single Grid Vorticity-Stream Function Method

Table 6.3 gives the convergence rates for the vorticity-stream function method on a single grid for the stationary problem given by the initial conditions (6.1). Errors and convergence rates are calculated for the vorticity ω by comparing the computed solution to the exact solution. The end time for this run is 1.0, the CFL number = 0.9, and $\nu = 0$. It is evident from the data in 6.3 that the method is converging at a third-order rate for this problem.

The convergence test used for the projection method for the double shear layer is also repeated for the vorticity-stream function method for the u-component of the velocity. While the stationary problem is certainly one for which a finite difference method should

give accurate results, the periodic double shear layer problem given by the initial conditions (6.2) is in a sense the most difficult type of problem for a finite difference method based on vorticity variables to solve. The vorticity field in this problem initially consists of two spikes of vorticity resembling delta functions that roll up into vortical structures. The evolution of these structures is the part of the flow that is of interest, but finite difference approximations to derivatives of the vorticity will be least accurate near these spikes of vorticity. Nevertheless, results of the convergence analysis of the vorticity-stream function method for this problem shown in table 6.4 reveal that the method is converging at a second-order rate or better for the u-component of the velocity up to time 0.8. In order to compute the differences $D^h_{h/2}$ for the u-component of the velocity, at each coarse grid cell edge, two fine grid cell-edge velocities were averaged and compared to the coarse grid velocity value.

A convergence test for the values of the vorticity was also done for the double shear layer problem and the results are displayed in in table 6.5. It is evident from the data that at short times the rate of convergence for the vorticity is nearly two, but as the vorticity field becomes more complicated, the rate drops off. Since this calculation is for Euler flow, there is no lower limit to the size of vortical features in the actual physical flow, so it is not surprising that the convergence rate of the method drops below two as the method begins to seriously under-resolve the features of the flow. Nonetheless, table 6.5 gives evidence that even later in the evolution of the flow, the method is converging asymptotically at nearly a second-order rate.

The right side of figure 6.1 displays a contour plot of the vorticity and the u-component of the velocity computed by the vorticity-stream function method for the double shear layer problem. The results shown are from a run with grid size $h = 1/256$. Note the close agreement between the vorticity-stream function solution shown on the right and the solution computed by the projection method shown on the left. Despite the fact that the two methods use different formulations of the Navier-Stokes equations, they compute nearly identical solutions to this problem. The contour plots of the velocities for these two methods shown in the bottom row of figure 6.1 are nearly indistinguishable.

Error of	32	rate	64	rate	128	rate	256
$\|\omega\|_2$	2.82E-1	2.63	4.54E-2	2.92	6.01E-3	2.99	7.56E-4
$\|\omega\|_\infty$	5.33E-1	2.89	7.17E-2	2.97	9.17E-3	3.00	1.15E-3

Table 6.3: Convergence rates for the vorticity-stream function method on the stationary problem.

Time	32-64	rate	64-128	rate	128-256	rate	256-512
0.4	2.39E-2	2.29	4.90E-3	2.44	9.04E-4	2.04	2.20E-4
0.8	8.28E-2	1.66	2.62E-2	2.16	5.85E-3	2.11	1.25E-3
1.2	8.57E-2	1.19	3.74E-2	1.56	1.27E-2	1.67	4.00E-3

Table 6.4: Convergence rates of the u-component of the velocity for the vorticity-stream function method on the inviscid shear layer problem.

Time	32-64	rate	64-128	rate	128-256	rate	256-512
0.4	9.27E-1	1.79	2.68E-1	2.12	6.24E-2	2.08	1.44E-2
0.8	2.74	0.87	1.49	1.25	6.27E-1	1.51	2.20E-1

Table 6.5: Convergence rates of the vorticity for the vorticity-stream function method on the inviscid shear layer problem.

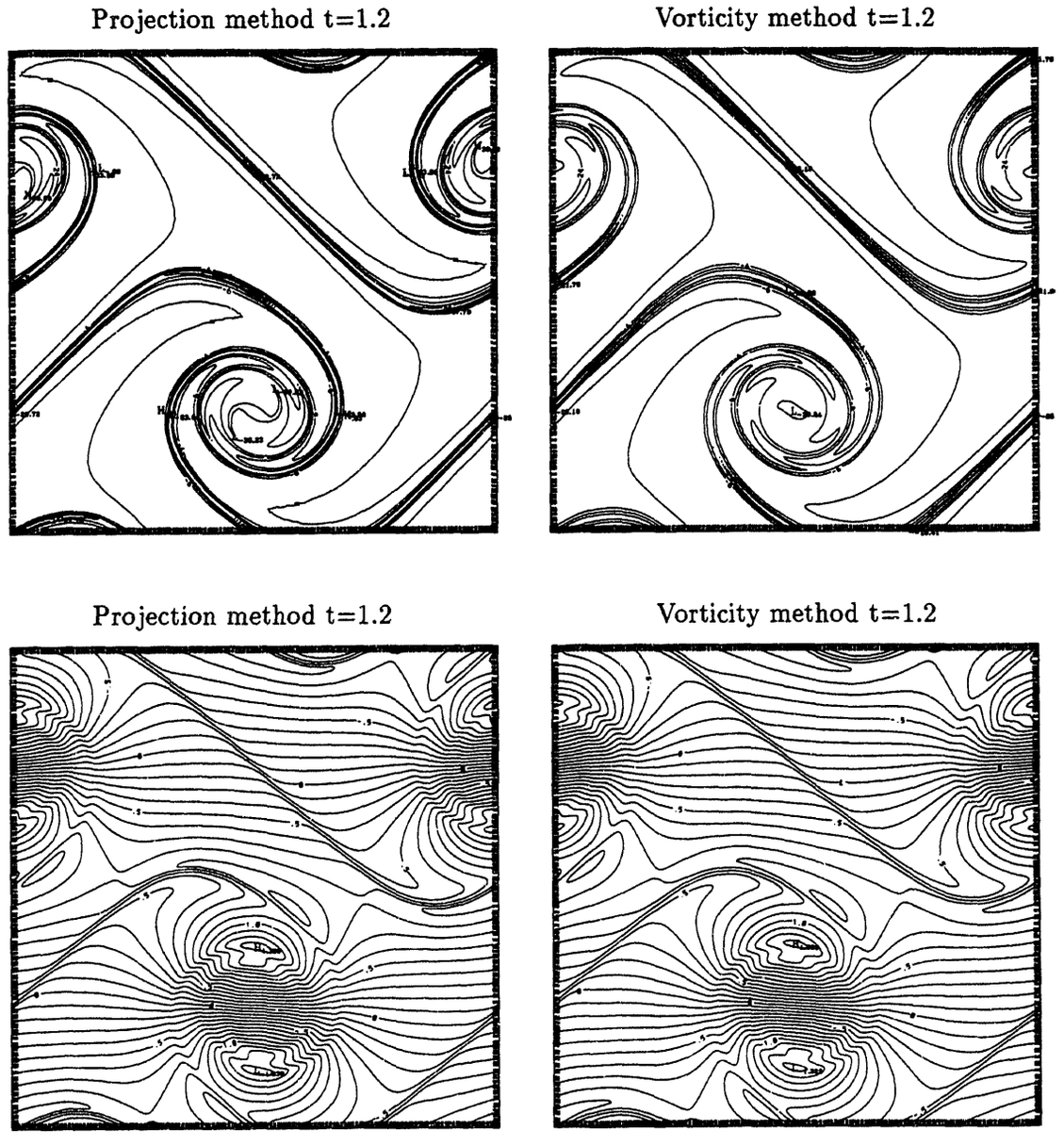


Figure 6.1: Double shear layer at time 1.2 for 256×256 grid. The vorticity is shown in the top row, the u-component velocity on the bottom.

6.1.3 Behavior of the Single Grid Schemes at Low CFL

In the locally refined grid version of the projection method, all grids are updated at every time step. A consequence of this is that for regions of the grid with coarse grid spacing, the effective CFL number can become very small. It is not obvious what effect a small CFL number has on the accuracy or convergence rate of a method. In order to investigate the effect of low CFL on the convergence rates of the refined grid methods, the convergence test on the stationary solution problem was repeated with CFL = 1/16. The stationary solution problem is used because it is assumed that in regions of a refined grid containing coarse grid points, the solution will be smooth and vary little in time. The data appearing in the tables below support the assertion that using a low CFL number in smooth regions of the flow has little effect on the overall accuracy of the method.

The convergence rates for the projection method are shown in table 6.6. The rate computed with the L_2 norm agrees closely with the results from the normal CFL run shown in 6.1. The convergence rate for the infinity norm is slightly lower than second order for this run. This is probably caused by the fact that the projection method employs slope limiters at the maxima and minima of the velocity field which in turn causes larger errors near these extrema.

The results from the vorticity-stream function method show astounding agreement between the two runs with CFL=0.9 and CFL=1/16. The vorticity field of these two runs agree to four decimal places, hence the convergence rates listed in table 6.7 agree with those in 6.3 exactly. It is apparent that the temporal errors for the vorticity stream-function method for this problem are negligible compared to the spatial errors.

6.2 Numerical Issues for the Refined Grid Methods

6.2.1 Numerical Approximation of Free Space Boundary Conditions

The vortex patch problems studied in this thesis are set in an unbounded physical domain. Since the computational domain is of course finite, some approximation to the free-space boundary conditions must be made at the edges of the computational domain. The boundary conditions for the patch problems are set in the following way: Since the vorticity distribution for each patch problem resembles a circular patch, at a distance from the patch, the solution will resemble the flow induced by a circular patch. The flow outside

Error of	32	rate	64	rate	128
$\|u\|_2$	9.95E-3	1.81	2.83E-3	2.12	6.49E-4
$\|u\|_\infty$	2.27E-2	1.52	7.87E-3	1.41	2.97E-3

Table 6.6: Convergence rates for the projection method on the stationary problem run with CFL number 1/16.

Error of	32	rate	64	rate	128
$\|w\|_2$	2.82E-1	2.63	4.54E-2	2.92	6.01E-3
$\|w\|_\infty$	5.33E-1	2.89	7.17E-2	2.97	9.17E-3

Table 6.7: Convergence rates for the vorticity-stream function method on the stationary problem run with CFL number 1/16.

of a circular patch of constant vorticity is given by the stream function

$$\psi = \frac{1}{2\pi}\bar{\omega}\log R$$

where R is the radial distance from the center of the patch and

$$\bar{\omega} = \int \omega d\bar{x}$$

with the integral taken over the support of ω . If ω_o is the value of the vorticity in the patch, and the patch has radius r, then $\bar{\omega} = \pi r^2 \omega_o$.

For a patch centered at the origin, this stream function yields the velocity field

$$\begin{aligned} u(x,y) &= -\frac{y}{2\pi}\frac{\bar{\omega}}{R^2} \\ v(x,y) &= \frac{x}{2\pi}\frac{\bar{\omega}}{R^2}. \end{aligned} \tag{6.3}$$

For the problems presented here, the velocities given in equation (6.3) 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 (6.3) instead of using an approximation to free-space boundary conditions affects the solution very little.

Given any vorticity distribution $\omega(\bar{\mathbf{x}})$ with compact support and total vorticity $\bar{\omega}$ corresponding to the velocity field U, suppose the center of mass of the vorticity distribution is given by the point $\bar{\mathbf{x}}$. One can define a point vortex located at $\bar{\mathbf{x}}$ as a vorticity distribution, i.e.

$$\tilde{\omega}(\mathbf{x}) = \bar{\omega}\delta(\bar{\mathbf{x}}). \tag{6.4}$$

Surely outside of some circle with large radius $\tilde{\omega}(\mathbf{x}) = \omega(\mathbf{x}) = 0$. One can also show that the velocity field $U(\mathbf{x})$ will converge to the velocity field $\tilde{U}(\mathbf{x})$ induced by (6.4) as the distance between $\mathbf{x}$ and $\bar{\mathbf{x}}$ goes to infinity. This is analogous to the gravitational effect of a cluster of stars such as a galaxy. At a great distance, the gravitational effect of the galaxy will be indistinguishable from that of a single body with a mass equal to that of the galaxy.

It is therefore tempting to try to approximate free-space boundary conditions for any problem with a compact vorticity distribution by imposing the velocities given by (6.3) at a great distance from $\bar{\mathbf{x}}$. In theory, this could be done efficiently by creating ever larger and coarser grids around the area of non-zero vorticity until the computational domain becomes large enough so that the error from the imposed boundary conditions is sufficiently small. Unfortunately, the author's computational experience shows that using a grid structure with many levels, which produces grid cell sizes that vary over several degrees of magnitude, leads to convergence problems for the multigrid routines used to solve the Poisson problems in the methods presented here. The problem lies in the fact that it becomes very difficult to enforce the solvability conditions for the Poisson equation for the composite residuals that occurs in the multigrid methods. The right-hand side of these problems contains residuals averaged from every grid level with each successive averaging causing a loss of precision. Because these composite problems have Neumann boundary conditions, the sum of the right-hand side should equal zero. (See Section 4.3.1.) It is the author's experience that using more than approximately a dozen multigrid levels can cause a large enough mismatch in the solvability condition for the composite residual problem on the coarsest grid to keep the overall method from converging. This is assuming the multigrid convergence criterion is that the residuals are uniformly reduced to less than 10^{-9} and using the 48-bit precision of Cray single precision floating point arithmetic. It was observed that increasing the accuracy of the floating point arithmetic to Cray double precision (96 bits) eliminated convergence difficulties to multigrid tolerances of 10^{-15} for some large problems, but the high computational cost of using double precision arithmetic on the Cray machines makes this a very unattractive alternative.

For free-space problems where the vorticity distribution does not resemble a circular patch, free-space boundary conditions would have to be explicitly computed. An interesting numerical method for approximating free-space boundary conditions using potential theory similar to a method proposed by Anderson [4] is presented by Almgren in [3], and could be readily implemented on refined grids.

In the numerical implementation of the methods, only the values of the velocities ever need to be computed at the boundaries. In the vorticity method, values of the vorticity at the boundary are zero. 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 for this problem, it must be true that the sum of the imposed boundary velocity values equal the total mass of the vorticity in the patch. 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 usually on the order of 10^{-5} or less. This adjustment is performed only during the computation of the initial conditions; since the method conserves the total mass of vorticity the adjustment never needs to be repeated.

6.2.2 Initial Conditions

The initial conditions for the remaining problems examined in this thesis are given in terms of the vorticity. For the projection method, initial conditions are required for the velocities and the pressure gradient. The initial velocities for the projection method are thus computed by first solving

$$\Delta\psi_{i,j} = -\omega_{i,j} \tag{6.5}$$

where ω is the initial vorticity and ψ the initial stream function. Centered finite difference approximations to the derivative of ψ are then used to compute the velocities. The Poisson equation (6.5) is solved exactly the same way as the one which is solved in the vorticity-stream function method. The Neumann boundary conditions for (6.5) translate to specifying the velocities at the computational boundary and are set as described in the previous section. The initial conditions for the gradient of the pressure are computed by iterating the time-stepping procedure with the initial velocities as is done for the single grid method.

For the vorticity method, the initial conditions for the stream function are computed by equation (6.5) in the same manner. The staggered grid velocity initial conditions are then computed from the initial stream function as in equation (3.7).

6.2.3 Location of Refined Grids

For each of the refined-grid computations described below, the computational grid contains areas of grid refinement. In each case, the position of the refined regions of the grid

are selected at the beginning of the run and do not change during the run. For the problems presented here, a general knowledge of the evolution of the solution is known beforehand. This allows the regions of grid refinement to be initially positioned in regions of the flow to reduce the discretization errors near large gradients and to capture interesting features as they develop in the flow.

For general problems with solutions that are not well understood, it is convenient to have a method for automatically determining where regions of grid refinement should be positioned. Adaptive mesh refinement (AMR) methods developed by Berger [11] have been applied to systems of partial differential equations by Berger and Oliger [14], to the Euler equations by Berger and Jameson [13], and to the equations of gas dynamics by Berger and Colella [12]. Attempts are currently underway to apply AMR to methods for the two- and three-dimensional Navier-Stokes equations [1]. These AMR methods use local error estimates of the solution to determine where refined grids should be placed.

It is likely that a combination of physical insight and AMR techniques will be necessary to effectively determine where refined grid regions are needed for complicated engineering applications using refined grid methods. In some instances, numerical viscosity caused by under-resolution 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. Of course, preliminary numerical experiments as well as physical insight can be used to identify such occurrences. It seems unlikely to the author that any AMR algorithm will be developed in the foreseeable future that makes it unnecessary for the user to help determine where refined grid regions are needed for all problems.

In this thesis, grid refinement is used to increase the resolution in some regions of the flow and to increase the size of the physical domain being used. The necessary size of the domain and resolution of the finest grid level were determined by blind guessing and numerical experiment. No further analytical justification of the choice of grid arrangements is provided.

6.3 Convergence Results for the Refined Grid Methods

To verify the convergence of the refined grid versions of the projection and vorticity-stream function methods, two convergence tests are presented. In the first test, an entire grid structure is refined, and convergence is demonstrated for a refined region of the grid.

Error	16	rate	32	rate	64
$\|u\|_2$	6.93E-2	3.18	7.63E-3	1.86	2.106E-3
$\|u\|_\infty$	1.21E-1	2.64	1.94E-2	1.72	5.89E-3

Table 6.8: Convergence rates for the projection method on the stationary problem on a refined grid with refinement factor 2.

Specifically, the inviscid stationary problem given by the initial conditions (6.1) is solved on the periodic unit square containing a single section of refined grid. For each example listed below, three runs are performed with the grid spacing on the coarsest grid being 1/32, 1/64, and 1/128, respectively. The length of the run for each case is 1.0, and the CFL number is 0.9. The refinement ratio for the refined section of the grid remains constant, 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 and infinity norm is computed for the error of the refined region only. For the projection method, the refined region is the square defined by the points $(0.1875, 0.1875)$ and $(0.3125, 0.3125)$, where the first point is the lower left corner of the refined grid and the second is the upper right corner. Table 6.8 contains the convergence results for the case when the refinement factor between the coarse grid and the refined region is 2, and table 6.9 for factor 4.

For the vorticity-stream function method, the refined patch is bounded by the square defined by $(0.3125, 0.3125)$ and $(0.4375, 0.4375)$. The convergence results for a refinement factor of 2 are given in table 6.10 and for factor 4 in 6.11. For both the projection and vorticity methods, it is apparent that the error in the refined patch is converging to zero in both the L_2 and infinity norms at a second-order rate or better. 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 factor 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. This paradox is caused by the fact that the errors in the refined region are 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, but at least for this example, grid refinement is only marginally advantageous. 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.

Error	16	rate	32	rate	64
$\|u\|_2$	3.23E-2	2.99	4.07E-3	2.53	7.06E-4
$\|u\|_\infty$	8.44E-2	2.11	1.96E-2	1.88	5.28E-3

Table 6.9: Convergence rates for the projection method on the stationary problem on a refined grid with refinement factor 4.

Error	16	rate	32	rate	64
$\|\omega\|_2$	1.66E-0	2.16	3.70E-1	2.41	6.98E-2
$\|\omega\|_\infty$	2.32E-0	2.28	4.76E-1	2.34	9.41E-2

Table 6.10: Convergence rates for the vorticity method on the stationary problem on a refined grid with refinement factor 2.

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 AMR must address both of these concerns.

In order to give a more convincing convergence example for a refined grid method, the projection method is applied to a vortex capture problem, and convergence of refined grid solutions is illustrated. A discussion of vortex capturing and an analysis of physical insight gained from the computations is contained in Section 6.5.

For this convergence test, three runs are done on a refined grid structure containing five grid levels, each consisting of a single grid. The coarsest two grids remain the same for the three runs, while the finest three levels are refined by increasing the refinement factor between levels one and two from 2 to 4 to 8. Table 6.12 gives the grid locations and cell sizes for each level for each of the three runs. The column labeled "Grid location" gives

Error	16	rate	32	rate	64
$\|\omega\|_2$	1.65E-0	2.02	4.09E-1	2.41	7.72E-2
$\|\omega\|_\infty$	1.92E-0	1.74	5.73E-1	2.48	1.03E-1

Table 6.11: Convergence rates for the vorticity method on the stationary problem on a refined grid with refinement factor 4.

Level	Grid location	run 1	run 2	run 3
0	(-512,-512) (512,512)	32	32	32
1	(-64,-64) (64,64)	4	4	4
2	(-16,-16) (16,16)	2	1	1/2
3	(-4,4) (4,4)	1/2	1/4	1/8
4	(-2,-2) (2,2)	1/8	1/16	1/32

Table 6.12: Grid cell sizes for the three runs in the vortex capture convergence study.

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 vortex capture 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 figure 6.3. The vorticity is contoured between 0.5 and 6.0 with a 0.5 increment between contour lines. Given these initial conditions, the two patches of vorticity will wind around each other to form a larger vortex patch centered at the origin.

For each of the three grid structures described in table 6.12, solutions were computed to time $t = 10$. Computed values of the u-component of velocity were compared and convergence rates were computed using the Richardson's procedure described above. The graph in figure 6.2 plots the computed rate at each time interval. The convergence rate is initially almost exactly 2 but begins to vary as the run progresses. In all, the method appears to be converging on the finest grid at nearly a second-order rate for this problem. Figure 6.3 also shows the vorticity contours for the solution on the finest grid for the three runs at time 5.0. 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 picture, respectively. For these plots, the vorticity is contoured every $\pi/2$, beginning at zero. A detailed analysis of this problem appears in Section 6.5.

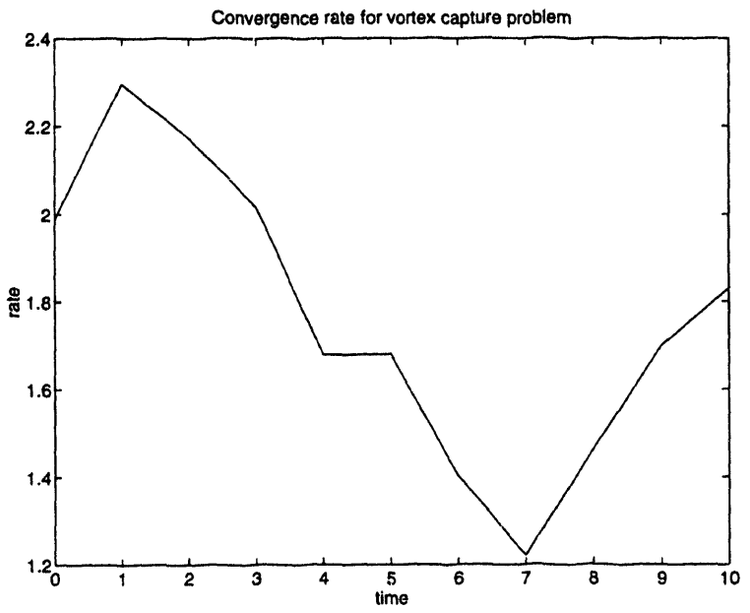


Figure 6.2: Convergence rate for vortex capture problem.

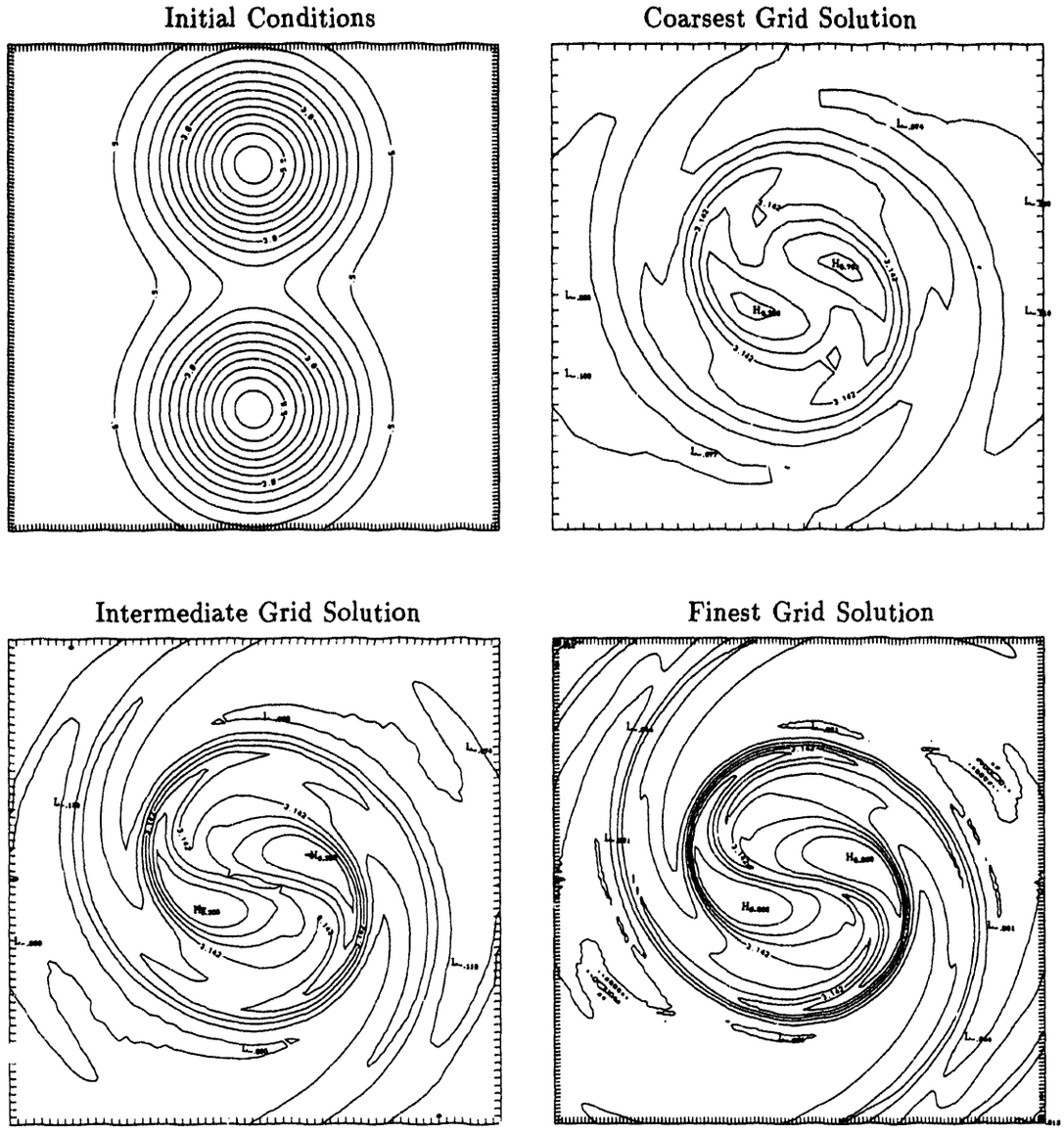


Figure 6.3: Vorticity contours for the vortex capture convergence runs. The initial conditions appear in the top left corner. The physical domain shown represents the square given by (-2,-2) and (2,2).

2 of 2

6.4 A Study of Vortex Patch Evolution and Filamentation

In this section, the two methods presented in this paper are applied to the modeling of the evolution of vortex patches. In the first example, both methods are shown to be accurate on an elliptical patch of vorticity. In the second example, the methods are used to model the filamentation of a patch of constant vorticity. Finally, the methods are used to illustrate the evolution of a smoothed patch of vorticity.

6.4.1 Elliptical Patches of Vorticity

The first example presented is designed to show that the projection and vorticity-stream function methods correctly compute the evolution of a vortex patch. Both methods are applied to an elliptical patch of constant vorticity. Elliptical patches of vorticity in a flow with background vorticity zero evolve by simply rotating in the plane. They are a simpler form of the more general ellipses studied by Kida [37] which rotate and change aspect ratio in time.

The initial boundary $s(\theta)$ can be parameterized by

$$s(\theta) = a\cos(\theta) + b\sin(\theta). \tag{6.6}$$

For a patch of constant vorticity ω_o the boundary of which is given by equation (6.6), the rotation rate Ω of the patch is given by

$$\Omega = \omega_o \frac{ab}{(a+b)^2},$$

or equivalently if $\gamma = a/b$ is the aspect ratio of the ellipse

$$\Omega = \omega_o \frac{\gamma}{(1+\gamma)^2}. \tag{6.7}$$

To validate the ability of the projection and vorticity-stream function methods to accurately model patches of vorticity, each is applied to model an elliptical patch of constant vorticity with value 2π and aspect ratio $\gamma = 0.9$ ($a = 0.9$ and $b = 1.0$). By equation (6.7), this patch will complete one rotation in time 361/90. The computations were carried out to time 361/360, the time for the patch to make one quarter of a revolution.

The computational grid for this problem consists of five levels of grid refinement with the finest level containing eight grids arranged around the perimeter of the patch. The computation is run twice with the second run having twice the resolution of the first. Table

6.13 contains the positions and sizes of each grid used in the coarse run, and figure 6.4.1 shows the position of the finest two grid levels superimposed over the initial patch location. Contour plots of the vorticity for the right-most refined grid region in figure 6.4.1 at time 361/360 are shown for the vorticity method in figure 6.5 and for the projection method in figure 6.6. For each figure, three plots from the coarser run are shown in the top row, and three from the fine run are shown on the bottom. The left-most plots are of the computed vorticity at time 361/360, while the right-most pictures are of the exact solution at the same time. The vorticity is contoured from 0 to 2π by intervals of $\pi/4$. Note that for both the projection and vorticity methods, the initially sharp vortex patch boundary is smeared during the run. For the vorticity method, the solution is very nearly constant inside and outside the patch boundary. The projection method however produces a vorticity field that deviates from being constant to a much greater extent, a fact illustrated by the complicated vorticity contours at the 0 and 2π level. This is not surprising since for the projection method the vorticity is computed by second-order finite differences of the velocities rather than being the primary variable as in the vorticity method. Note that neither method produces any type of unphysical oscillations near the patch boundaries.

To better illustrate where the patch boundary is located for each run, a patch of constant vorticity is created from the vorticity fields produced by the methods by assigning the value of zero to the patch if the value of the computed vorticity is less than π, and 2π if the computed vorticity is greater than π. This has the effect of sharpening the vorticity boundary to once again approximate a discontinuous vorticity field. Contour plots of these vortex patches are shown in the middle frames of figures 6.5 and 6.6. Note that although both methods smear the patch boundaries, the middle figures agree closely with the exact solutions to the granularity of the mesh. This is an indication that the methods can be used to accurately represent the boundary of a patch that has a discontinuous jump in vorticity, at least for relatively simple boundaries.

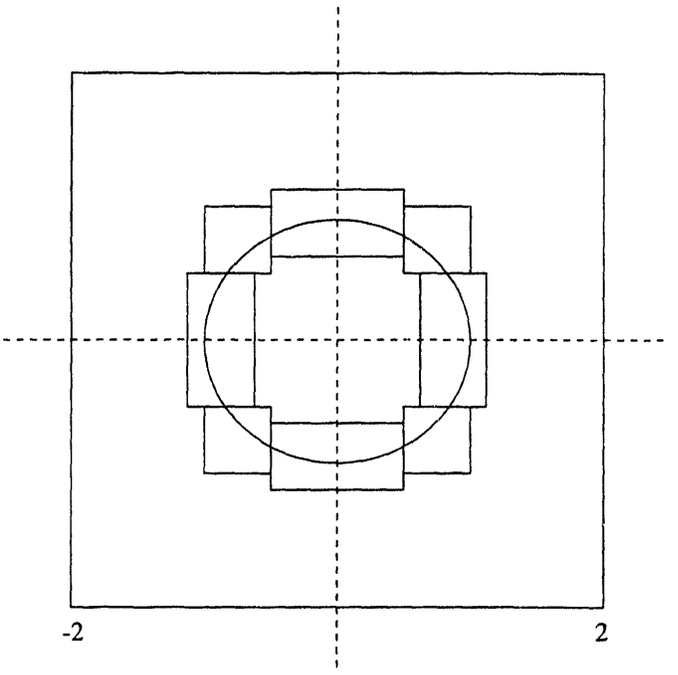


Figure 6.4: Finest two levels of grid refinement for elliptical patch test problem.

Location	level	size	grid spacing
(-128,-128) (128,128)	0	32×32	8
(-16,-16) (16,16)	1	32×32	1
(-4,-4) (4,4)	2	32×32	1/4
(-2,-2) (2,2)	3	64×64	1/16
(0.625,-0.5) (1.125,0.5)	4	32×64	1/64†
(0.5,0.5) (1.0,1.0)	4	32×32	1/64
6 Symmetries of above of above 2	4	-	1/64

Table 6.13: Twelve grids used in the elliptical vortex patch problems. † denotes the grid shown in the figures.

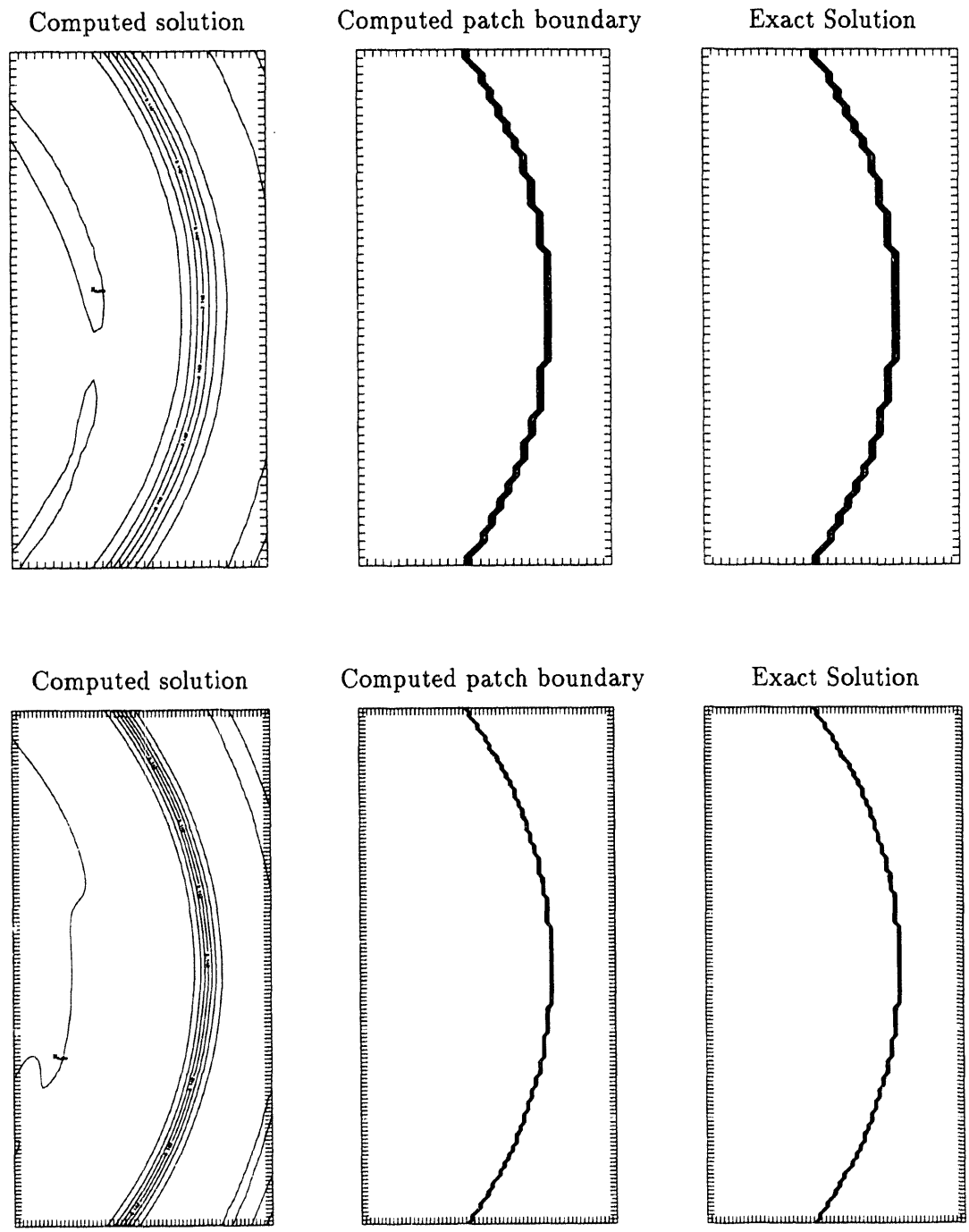


Figure 6.5: Final solution of elliptical patch calculation for the vorticity stream function method.

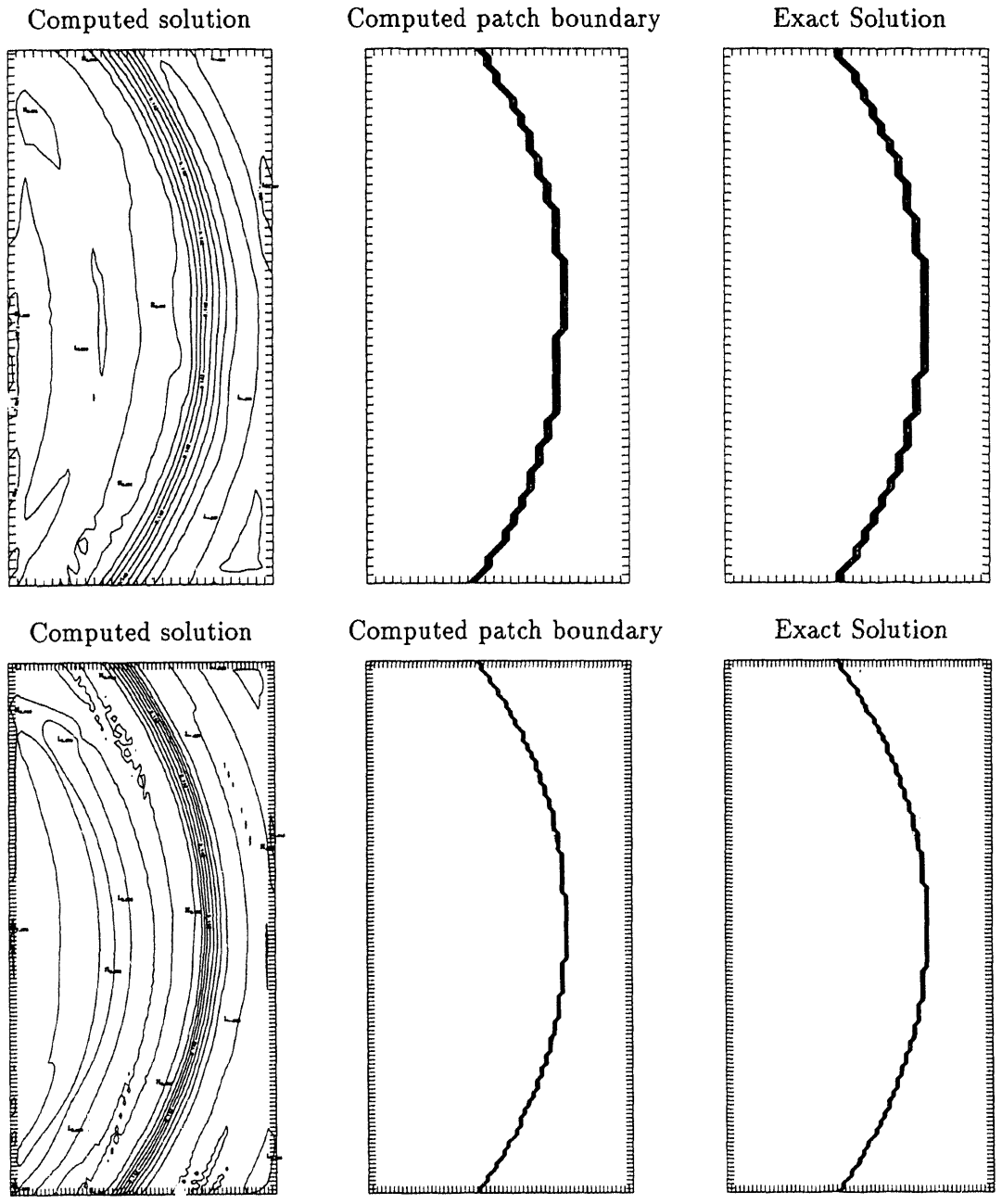


Figure 6.6: Final solution of elliptical patch calculation for the projection method.

6.4.2 Filamentation of Patches of Constant Vorticity

In order to study the filamentation of a vortex patch, numerical experiments are performed on a circular patch of constant vorticity the boundary of which is initially given a small perturbation. The experiments presented here are similar to those performed by Dritschel [27]. Each vortex patch is centered at the origin and the radial distance $r(\theta)$ from the origin to the patch boundary is parameterized in polar coordinates by

$$r(\theta) = 1 + a(\theta/\theta_o)e^{-\frac{1}{2}(\theta/\theta_o)^2} \tag{6.8}$$

for $-\pi \leq \theta \leq \pi$. Here a represents the perturbation amplitude and θ_o defines the radial extent of the disturbance. In this experiment, as in [27], $\theta_o = \pi/m$ for some integer m. Figure 6.4.2 shows a close-up view of an initial disturbance given by $a = 0.1$ and $m = 100$. In [27] the perturbations used are smaller and less sharp than the ones presented here. Also, the perturbations in [27] have the opposite sign as the perturbation in (6.8). Dritschel's calculations utilize a contour surgery algorithm which is capable of resolving very small disturbances and performing extended calculations but only for patches of constant vorticity (or piecewise constant). The perturbations for this experiment were chosen to be small, but large enough to be readily resolved, and to be steep enough in nature to exhibit filamentation in a relatively short time.

The refined grid structure used for the circular patches is similar to the one used for the elliptical patches studied in the last section. A twelve-grid structure like the one used for the elliptical patch problem is used first, and then an additional run with an extra level of grid refinement is performed. This second calculation uses an additional forty-eight grids located around the perimeter of the patch to increase refinement near the boundary and to reduce the amount of smearing of the vorticity discontinuity. Table 6.14 contains the sizes and locations of the grids for the finest grid example.

Figure 6.9 through 6.12 display the results of the numerical runs. The first two figures show the results from the twelve-grid examples. For each time shown, the projection method results appear on the right while the results from the vorticity-stream function method appear on the left. As the calculation progresses, the perturbation travels around the boundary of the patch as it evolves, making approximately one complete cycle in time $t = 2$. The perturbation is shown on the finest grid level at times $t = 0.5, 1.0, 1.5$, and 2.0.

The first thing to note is that as the patch filaments, the vorticity-stream function method dissipates the filament at a greater rate than the projection method. In each case,

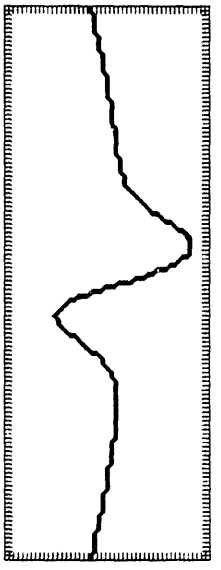


Figure 6.7: Close-up view of perturbation to circular patch given by $a = 0.1$ and $m = 100$.

the filament appears longer and thinner in the picture produced from the projection method. This is to be expected since the amount of numerical dissipation for the projection method is based on the variations of the gradients of the velocities while in the vorticity-stream function method it is based on the much steeper gradient of the vorticity. Qualitatively however, the two methods agree.

By time $t = 2$, both methods show that the filament has merged with the patch boundary and the initial perturbation has been effectively eliminated. If one believes that numerical dissipation is in any way modeling actual physical viscosity, this would lead one to believe that vortex filaments of this sort in nearly inviscid flow would also eventually be dissipated. Of course numerical dissipation is not an exact representation of physical viscosity, but each acts to remove small scales from the flow and each has the most effect near large gradients in the flow.

The second set of calculations, the results of which are shown in figures 6.11 and 6.12, have a finest grid resolution of twice that of the first calculation. This will presumably reduce the amount of numerical dissipation and the figures support this presumption. In both the projection and vorticity methods, the filaments become longer and thinner than they did in the first example. However, even with the greater resolution, the filaments have

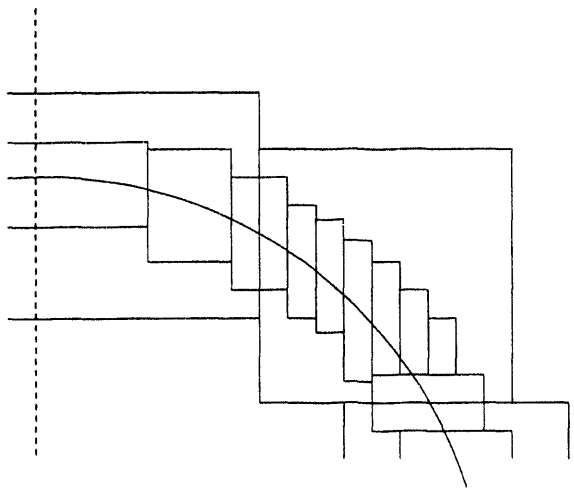


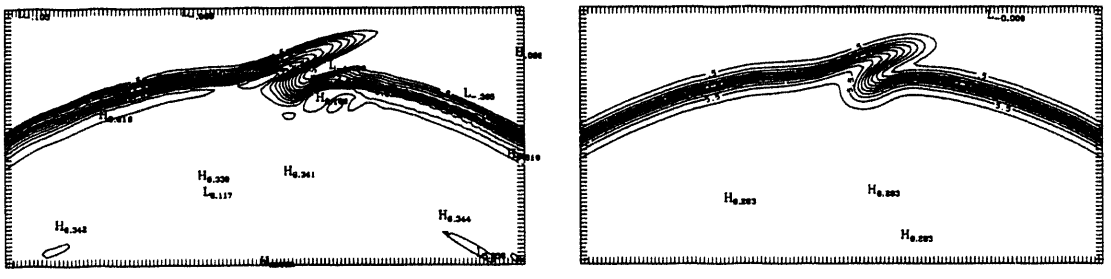
Figure 6.8: Section of finest two levels of grid refinement for patch filamentation problem.

nearly completely merged with the patch boundary by time $t = 2$. It seems completely reasonable that no matter how fine a grid resolution used for this problem, as the filaments become longer and thinner, numerical dissipation will eventually smear the separation between the filaments and the patch boundary. It is also reasonable to presume that a small viscosity in physical flow will have the same effect. If true, this would imply that the filamentation process for nearly stationary patches of vorticity in flow with an arbitrarily small viscosity will eventually be removed and that the patch will return to a nearly stationary state.

Location	level	size	grid spacing
(-128,-128) (128,128)	0	32×32	8
(-16,-16) (16,16)	1	32×32	1
(-4,-4) (4,4)	2	32×32	1/4
(-1.25,-1.25) (1.25,1.25)	3	64×64	1/16
(0.5625,-0.5) (1.1875,0.5)	4	32×64	1/64
(0.5,0.5) (1.0625,1.0625	4	32×32	1/64
6 Symmetries of above	4	-	1/64
of above 2			
(0.890625,-0.25) (1.078125,0.25)	5	48×128	1/256
(0.8125,0.25) (1.0625,0.4375)	5	64×48	1/256
(0.75,0.4375) (1.0,0.5625)	5	64×32	1/256
(0.875,0.5625) (0.9375,0.6875)	5	16×32	1/256
(0.8125,0.5625) (0.875,0.75)	5	16×48	1/256
(0.75,0.5625) (0.8125,0.8125)	5	16×64	1/256
(0.6875,0.546875) (0.75,0.859375)	5	16×80	1/256
(0.625,0.65625) (0.6875,0.90625)	5	16×64	1/256
(0.5625,0.6875) (0.625,0.9375)	5	16×64	1/256
(0.4375,0.75) (0.5625,1.0)	5	32×64	1/256
(0.25,0.8125) (0.4375,1.0625)	5	48×64	1/256
33 symmetries of	5	-	1/256
above 11			

Table 6.14: Sixty grids used in perturbed circular vortex patch problems.

Projection method t=0.5 Vorticity method t=0.5



Projection method t=0.5 Vorticity method t=1.0

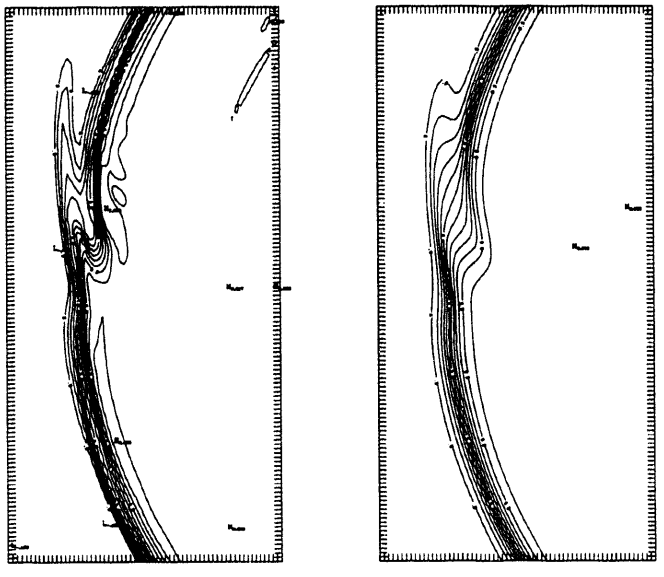
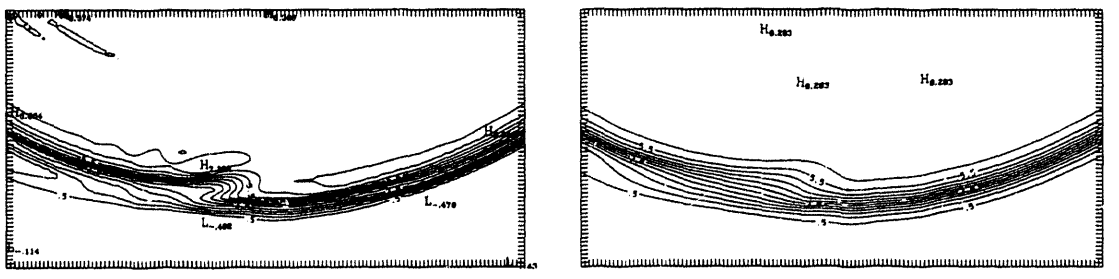


Figure 6.9: Vortex patch filamentation from the twelve grid example at times $t = 0.5$ and $t = 1.0$.

Projection method t=1.5 Vorticity method t=1.5



Projection method t=2.0 Vorticity method t=2.0

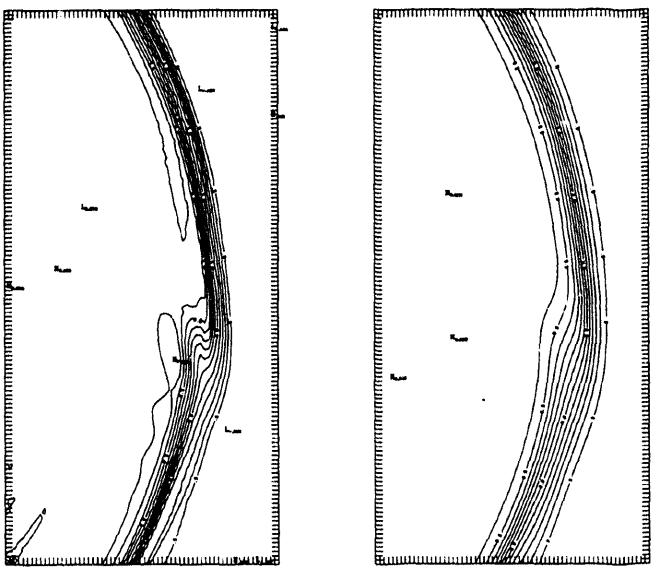
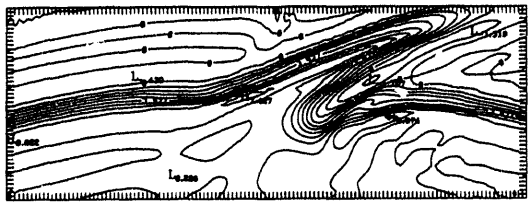
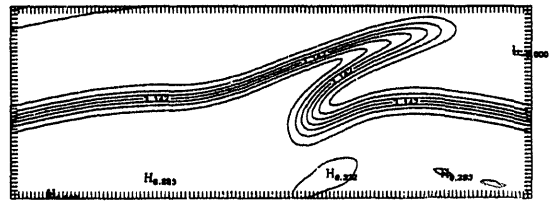


Figure 6.10: Vortex patch filamentation from the twelve grid example at time $t = 1.5$ and $t = 2.0$.

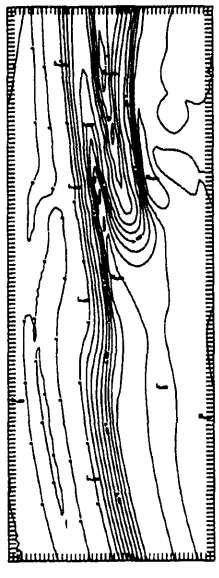
Projection method t=0.5



Vorticity method t=0.5



Projection method t=1.0



Vorticity method t=1.0

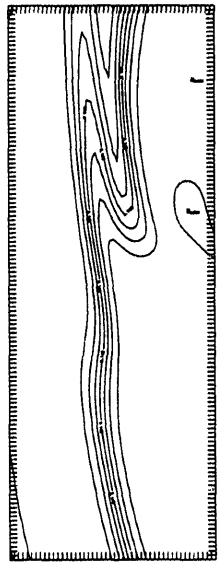


Figure 6.11: Vortex patch filamentation from sixty grid computation at times $t = 0.5$ and $t = 1.0$.

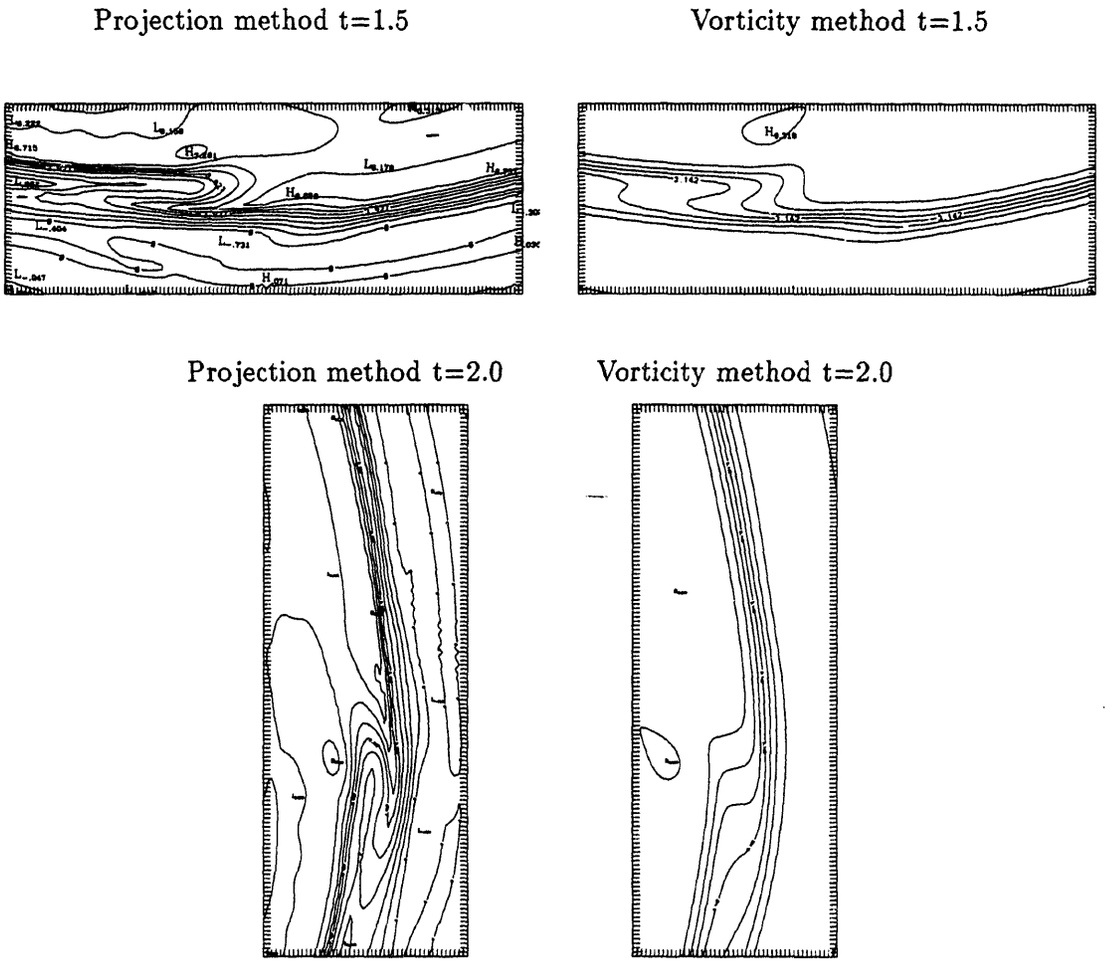


Figure 6.12: Vortex patch filamentation from sixty grid computation at times $t = 1.5$ and $t = 2.0$.

6.4.3 Filamentation of Smooth Vortex Patches

The vortex patch calculations presented in Section 6.4.2 are specified as patches of vorticity with discontinuities in the vorticity field at the patch edges. As seen in the above figures, finite difference methods tend to smear sharp discontinuities. Even the projection method, for which the discontinuity occurs in the derivative of the velocities, smooths out the patch boundary. Also, finite difference calculations produce unavoidable numerical dissipation in the flows which in general has the greatest effect near discontinuities or regions of large gradients.

Patches of vorticity in which the vorticity field at the boundary of the patch is not discontinuous but instead has a large gradient have not been studied nearly as extensively as the constant valued patch problem. This is mainly because methods like contour dynamics or vortex patch methods, which are very effective for constant patch problems, are not readily applicable to patches with smooth boundaries. For Euler flow, it is generally accepted that unsteady patches undergo repeated filamentation as they evolve toward equilibrium states. It is not immediately obvious how this process changes when the boundary of the patch is smooth.

Arnold [6] has proven that certain stationary distributions of vorticity are stable, i.e. the growth of perturbations to the stationary distributions is bounded in time. Smooth, radially symmetric patches in which the vorticity monotonically decreases from the center of the patch, satisfy the conditions of the proof and are hence stable. On the other hand, any voriticity distribution which is constant on an open set on which the stream function varies does not satisfy the conditions of the proof. In particular, the theorem says nothing about patches of constant vorticity with smooth edges. Radially symmetric patches of vorticity in which the vorticity increases with radial distance presumably are not stable. This places patches with some constant region of vorticity at at a transition point between stable and non-stable patches. When and if these patches filament is not clearly understood.

The initial conditions for the smooth vortex patch problem are given by a smoothing of the initial conditions for the constant patch problems by a hyperbolic tangent function. Specifically, the vorticity is given in polar coordinates by

$$\omega(r,\theta) = \pi(1 - \tanh(\rho(r - r_b))) \tag{6.9}$$

where

$$r_b(r,\theta) = 1 + a(\theta/\theta_o)e^{-\frac{1}{2}(\theta/\theta_o)^2},$$

and a and θ_o are defined as in the constant patch problem. The value of ρ in equation (6.9) determines the thickness of the the patch boundary. This distribution of vorticity is essentially constant near the origin.

The projection and vorticity stream-function method are applied to the smooth patch problem using the 12-grid refined grid structure described by table 6.13. Two calculations are done with each method, one with $\rho = 20$ and the other with $\rho = 10$. As in the constant patch problem $a = 0.1$ and $m = 100$ for both runs. The calculations are carried out to time $t = 2.0$. Figures 6.13 and 6.14 display the evolution of the smoothed patch perturbation as the patch rotates.

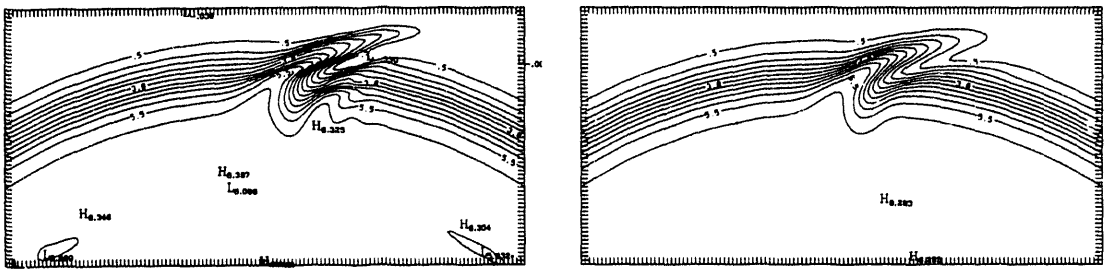
It is obvious from the figures that the numerical dissipation of the vorticity-stream function method is having a greater effect on the evolving filament than that of the projection method. This is expected since the velocity fields for this problem do not contain the large gradient that the vorticty field has near the patch boundary. Still, the two methods agree qualitatively. The filament illustrated by the stretching of the lowest vorticity contours quickly diffuses and the patch boundary becomes more regular as the calculation progresses. For slightly viscous flows, it seems unlikely that filamentation is an important mechanism of the transition to equilibrium for nearly steady smooth patches. In the inviscid case, however, the stretching of the vorticity contours seen in the solutions of the smooth patch problem could continue without dissipating. The filament arms would then eventually wrap around the core of the vortex patch as they become longer and thinner. This could lead to an eventual sharpening of the gradient of the patch boundary similar to what is seen at the core of the vortex capture problem examined in Section 6.5. Once filamentation steepens the patch boundary, the core itself could filament, eventually sending arms of the higher valued vorticity found in the vortex core into the surrounding flow.

It is also possible that for this form of perturbation a sufficiently smooth vortex patch, i.e. one such that the magnitude of the gradient never reaches some critical value, filamentation of the vortex core will not occur. One can readily see that for the smoother vortex patch (with $\rho = 10$), the amount of filamentation seen at the innermost contour line is less than for the patch with the steeper boundary ($\rho = 20$). Whether or not there is a level of smoothness in a patch boundary sufficiently steep to prohibit filamentation is a very difficult question to answer numerically. If a finite difference method is being used for the calculation, one cannot be sure whether or not the failure of a patch to filament is due to the geometry of a patch, or to numerical dissipation. If a contour dynamics or vortex

method is being used which represents a continuous vorticity field as piecewise constant patch, one must be concerned with whether or not a piecewise constant vorticity exhibits the same filamentation behavior as a continuous one.

Projection method t=0.5

Vorticity method t=0.5



Projection method t=1.0

Vorticity method t=1.0

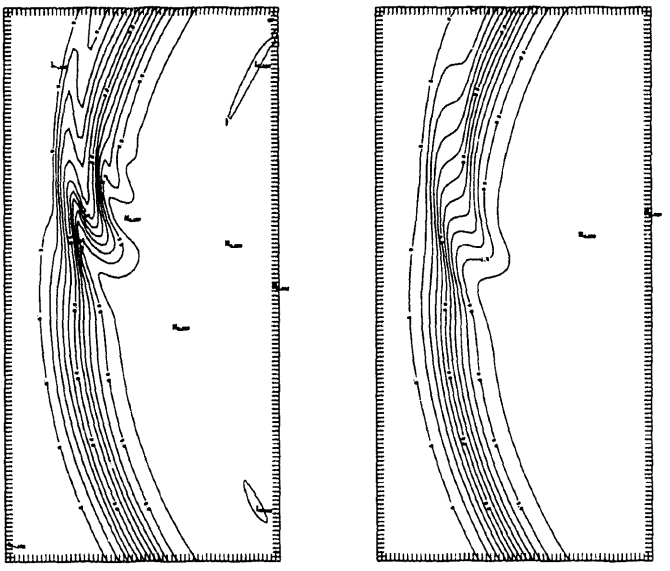
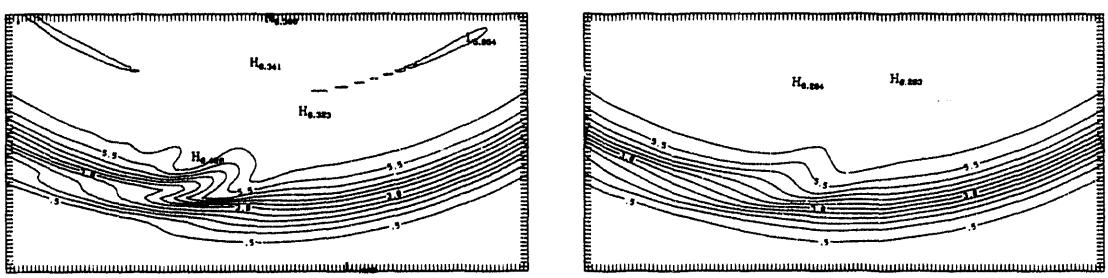


Figure 6.13: Initial filamentation of smooth vortex patch with $\rho = 20$.

Projection method t=1.5

Vorticity method t=1.5



Projection method t=2.0

Vorticity method t=2.0

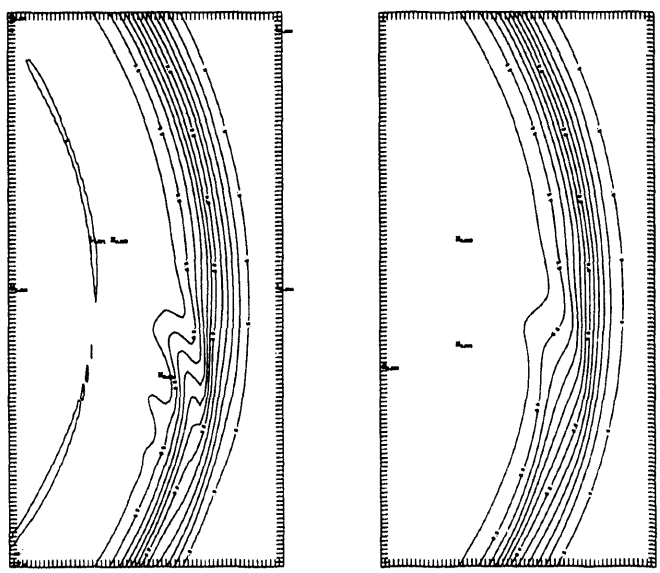
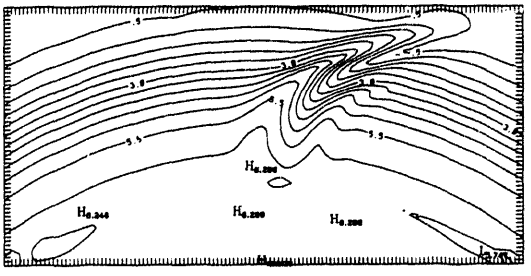
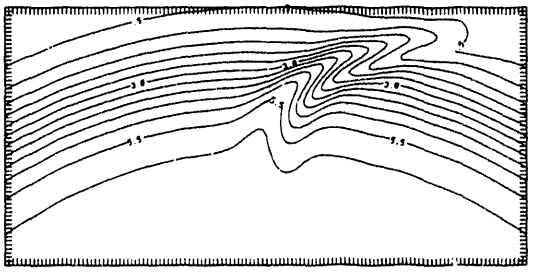


Figure 6.14: Filamentation of smooth vortex patch with $\rho = 20$.

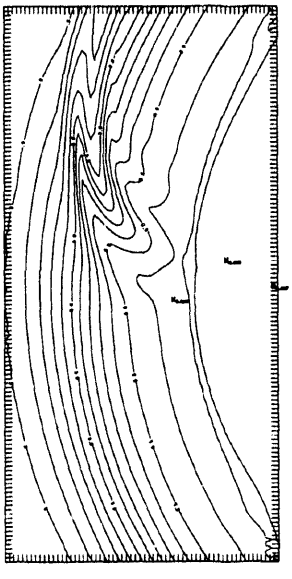
Projection method t=0.5



Vorticity method t=0.5



Projection method t=1.0



Vorticity method t=1.0

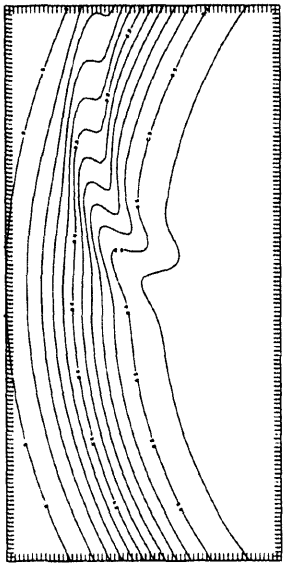
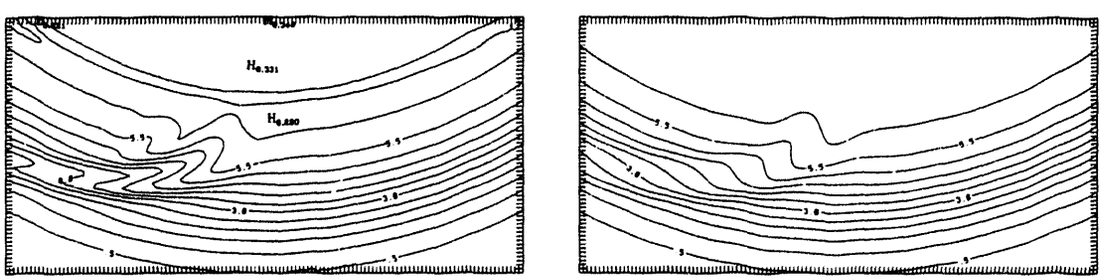


Figure 6.15: Initial filamentation of smooth vortex patch, $\rho = 10$.

Projection method t=1.5

Vorticity method t=1.5



Projection method t=2.0

Vorticity method t=2.0

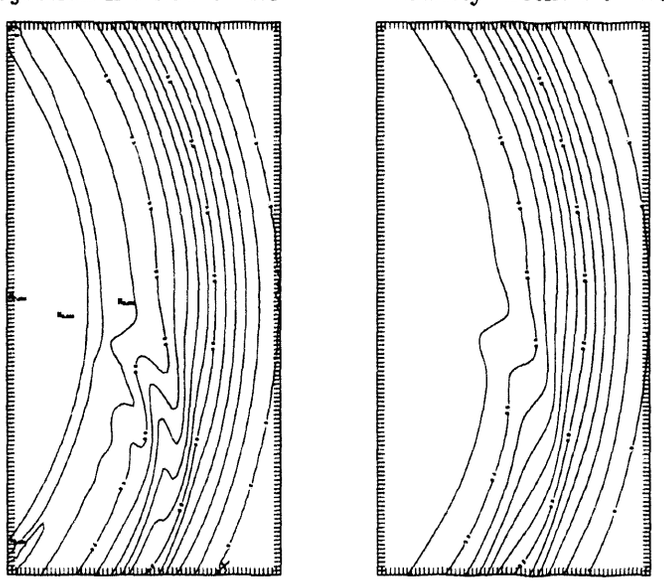


Figure 6.16: Filamentation of smooth vortex patch, $\rho = 10$.

6.5 An Analysis of Vortex Capture

The evolution of two circular patches of constant vorticity has been extensively studied. (See Buttke [18], Wang [51], and references therein.) These simulations show that the two patches will move toward and wrap around each other as they rotate about their center of mass. The shear flow produced by each vortex patch causes thin arms of vorticity to be pulled from the opposite patch and wrap around the vortex cores as they move together. In time the filaments become extremely complicated, but in Euler flow, the patches will never merge; there will always be a layer of zero vorticity between the boundaries of the two patches. It is believed however that the distribution of vorticity will in time approach an equilibrium state which corresponds to a solution of the Joyce-Montgomery equations [23]. One can visualize this by imagining one patch being red and one patch blue. After a long time, the plane will be full of long red and blue filaments circling out from a minuscule vortex core. The red and blue will never combine, but viewed on a slightly cruder scale, the vorticity will appear "green" with the "deepness" of the green being approximated by the equilibrium distribution.

Calculations that begin with an arbitrary random vorticity distribution in a periodic domain are presented by Montgomery et al. in [41]. The numerical results show that as the flow progresses, the vorticity field begins to organize into isolated groups of same-signed smoothed vortex patches. Eventually all vortex patches merge, and the flow consists of two counter-rotating smooth vortex patches of opposite sign. The approach to this equilibrium state consists of many instances of "vortex capture" in which one vortex patch absorbs another of like sign, or more simply two vortex patches merge into one. In the following example, evidence is presented that the vortex capturing phenomenon resembles the example of two constant patches in that filamentation is the dominant mechanism for moving toward the equilibrium state.

To model vortex capture, the evolution of two identical smooth patches of vorticity is computed. The initial conditions, which appeared earlier in the convergence study of Section 6.3, are given by

$$\omega(x,y) = 2\pi(e^{-2(x^2+(y-1)^2)} + e^{-2(x^2+(y+1)^2)}). \tag{6.10}$$

The pictures shown in figure 6.3 confirm that as in the case for patches of constant vorticity, as the flow evolves, the two patches move together with vortex filaments forming

and wrapping around the central cores. It is apparent from these runs that a very steep gradient of the vorticity forms between the two vorticity maxima as they move closer together. To better resolve this region, a sixth grid level is added to the grid that was used in the convergence study. Table 6.15 displays the locations and sizes of the grids for this computation.

The projection and vorticity-stream function methods are used to compute the evolution of the vortex patches to time $t = 10$ on the refined grid structure described by table 6.15. The results are shown in figures 6.17 through 6.19. Figures 6.17 and 6.18 show contours of the vorticity from the second finest grid level which represents the square region of the plane defined by the points $(-2,-2)$ and $(2,2)$. The final figure shows close-up views of the region around the origin that was computed on the finest grid. In all cases the vorticity contours begin at 0.5 and extend to 6.0 by 0.5 increments.

The first thing to note from the figures is that even at the extremely small scales shown in figure 6.19, the projection method solution and the vorticity-stream function solution agree in several important respects. The rate of roll up for the two vortices is nearly identical for the two methods, and even dissipative effects occurring on a small scale such as the breaking of contour lines at the origin manifest themselves at comparable times. It is especially apparent from figures 6.17 and 6.18 that there is no loss of accuracy in the method occurring at coarse fine grid interfaces. Figures 6.17 and 6.18 show vorticity contours computed from the second coarsest grid. At the center of this grid is a refined grid region, and any loss of accuracy at this coasre-fine interface would manifest itself by disturbing the vorticity contours as they cross the interface. Even though a region of sharp vorticity gradients is rapidly spinning across the refined regions, the vorticity contour lines remain smooth and unbroken.

At time $t = 2.0$ one can see from figure 6.17 that the two vortex patches are moving toward each other as each begins to elongate. By time $t = 4.0$, the region between the vorticity maxima of the two patches has been squeezed together so that the gradient of the vorticity in this region is very large. In the absence of viscosity, the contour lines of vorticity running between the two maxima would not break, they would continue to be squeezed together until a virtual discontinuity in the vorticity field would form. The numerical viscosity in the finite difference methods prohibit the formation of such steep gradients and hence the contour lines between the maxima do break. Also, one can see in the figures that the tails of the vortex filaments which, in Euler flow would remain distinct

Level	Grid location	cell size
0	(-512,-512) (512,512)	32
1	(-64,-64) (64,64)	4
2	(-16,-16) (16,16)	1/2
3	(-4,-4) (4,4)	1/8
4	(-2,-2) (2,2)	1/32
5	(-0.5,-0.5) (0.5,0.5)	1/128

Table 6.15: Grid cell sizes for the three runs in the vortex capture convergence study.

are merging together to form a smoother distribution of vorticity. By the end time of the run, the distribution of vorticity resembles a smoothed version of the expected equilibrium state for the inviscid constant patch problem.

The fact that the solution of the vortex capture problem presented here resembles the equilibrium solution of the inviscid constant patch problem is by no means a coincidence. Statistical arguments prove that the equilibrium states for the two problems will both satisfy the Joyce-Montgomery equations, although at different values of the temperature. It is conjectured here that very small amounts of viscosity have little effect on the equilibrium solution of the vortex capture problem for moderate or "meso-scopic" time scales. This conjecture could in theory be validated numerically by first computing the equilibrium statistical temperature of the flow by comparing the distribution of the vorticity and the stream function of the computed solution to a solution of the Joyce-Montgomery equation. This computed temperature could then be compared to the temperature of the initial conditions computed by variational methods. Turkington and Whitaker [50] present the details of such a variational procedure.

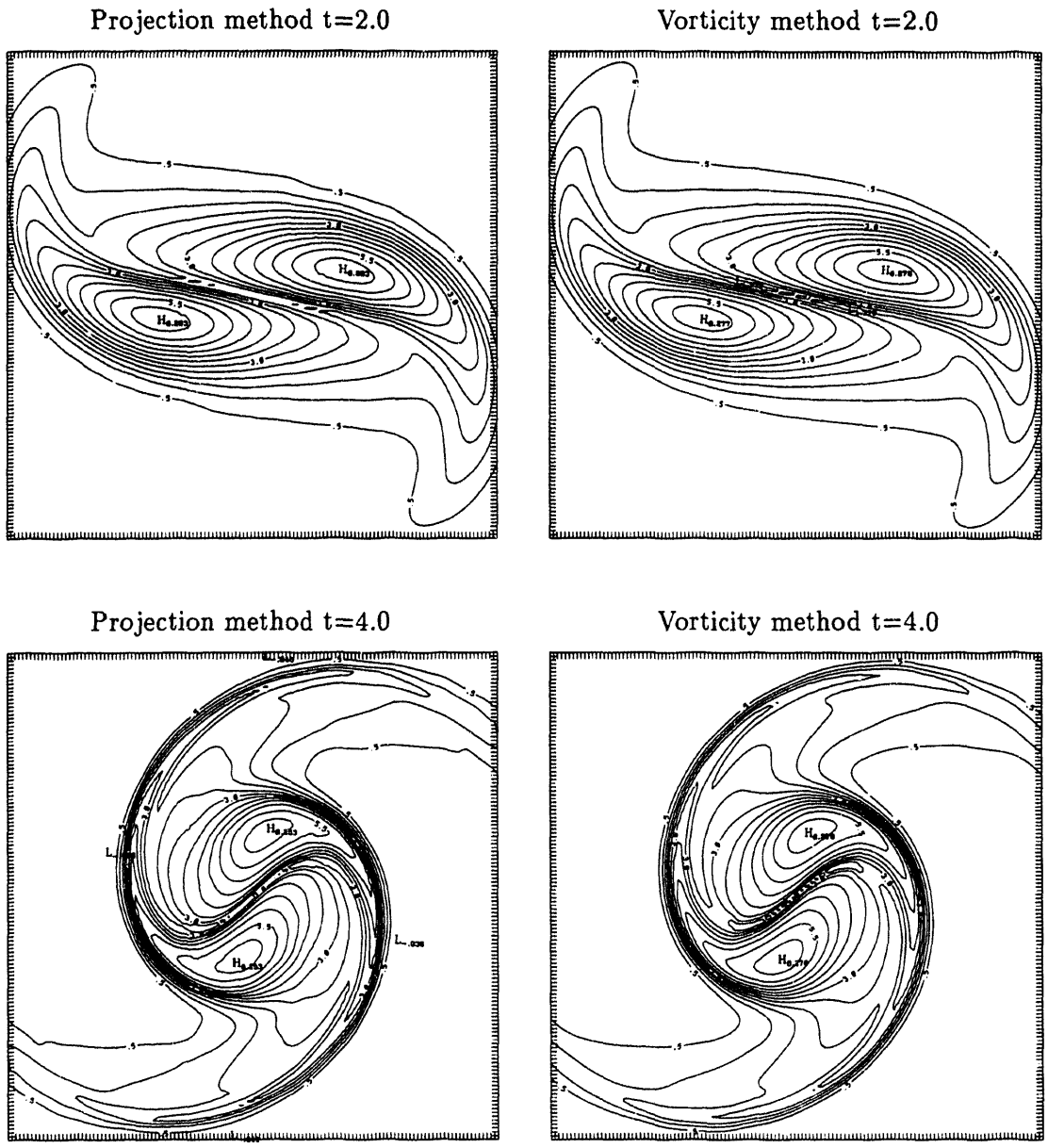


Figure 6.17: Comparison of projection method and vorticity-stream function method on the vortex capture problem at times $t = 2$ and $t = 4$.

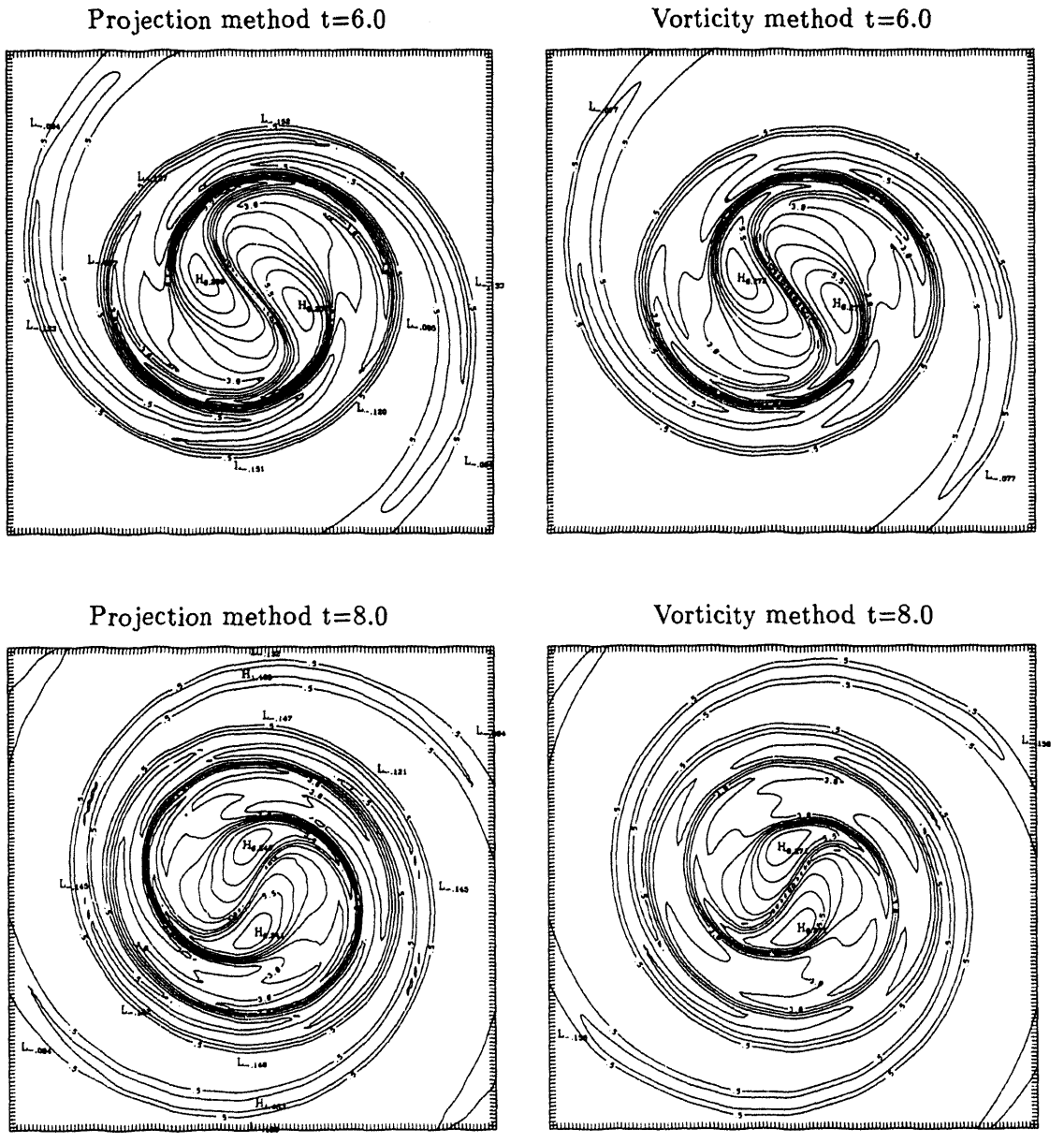


Figure 6.18: Comparison of projection method and vorticity-stream function method on the vortex capture problem at times $t = 6$ and $t = 8$.

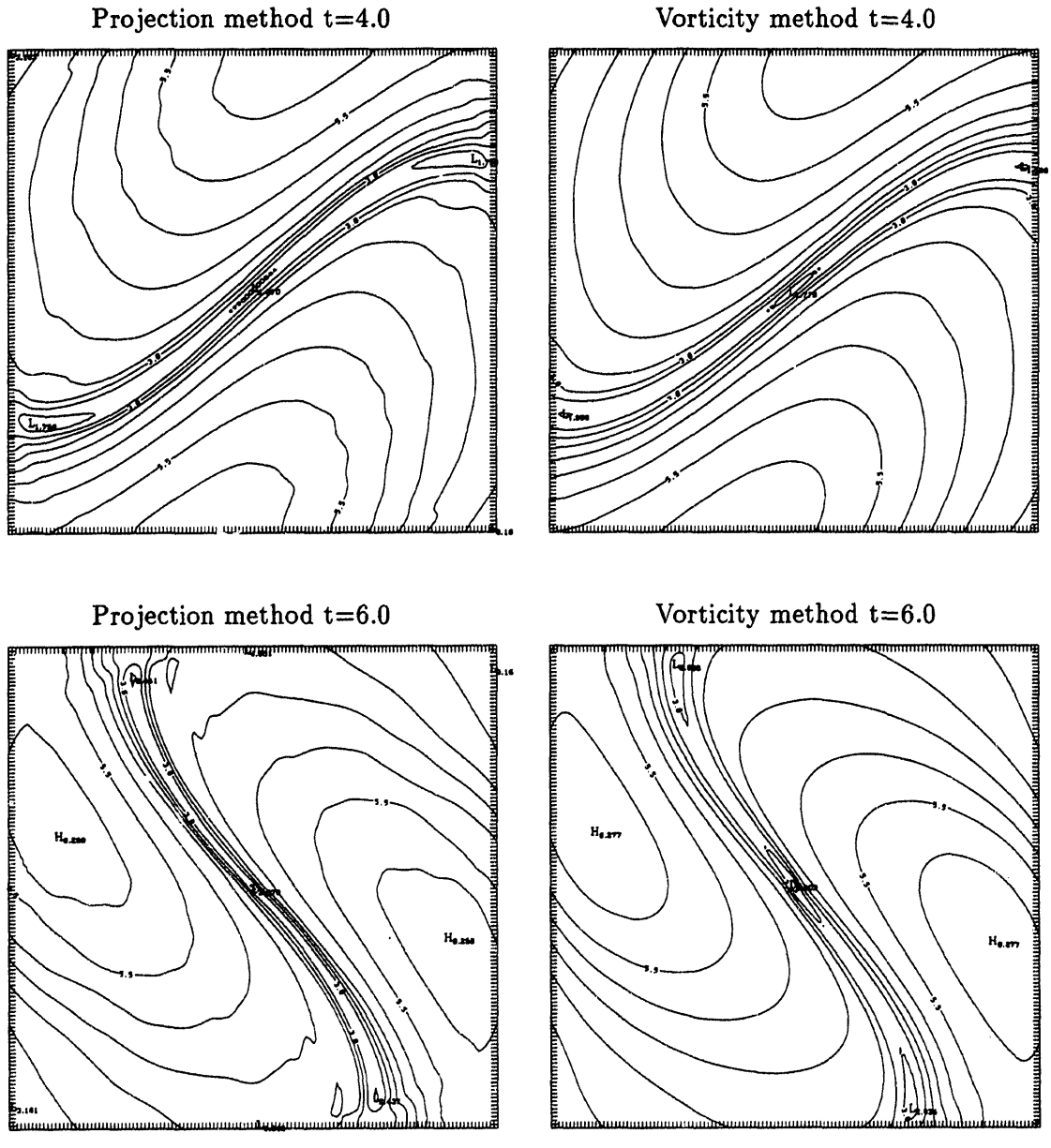


Figure 6.19: Finest grid comparison of projection method and vorticity-stream function method on the vortex capture problem at times $t = 4$ and $t = 6$.

6.6 Concluding Remarks

Some general remarks can be made regarding the results from the patch filamentation and the vortex capture problems. The close agreement of the two methods on a variety of problems gives very strong evidence that they are producing physically meaningful results. The experiments performed on the vortex patches also help illustrate the shortcomings of finite difference methods for problems of this sort. Both the projection and vorticity-stream function methods produce numerical dissipation that makes the study of exceedingly small structures like vortex filaments difficult. The question of whether or not a certain initial distribution of vorticity will in time undergo filamentation appears to be particularly difficult to address with finite difference methods. The increase in resolution given by these methods can capture the origins of the filamentation process, but the experiments suggest that numerical dissipation will eventually dissipate filaments regardless of the level of resolution. Nevertheless, the computations presented here shed some light into the behavior of vortex filaments for smooth patches of viscosity and possibly help explain the role of small amounts of dissipation in these systems. Further studies with even greater resolution are possible. The projection method computation of the perturbed circular patch of vorticity implemented with sixty individual grids took less than four hours to complete on a Cray YMP. Improving the resolution of this run by a factor of four or even eight is possible with current supercomputers.

The vortex capture problem demonstrates the usefulness of finite difference methods on locally refined grids for studying the transition of flows to equilibrium states. For two dimensional problems, where the theory of statistical equilibrium states is well developed, refined grid methods could be used to numerically validate statistical theory. The question of whether or not small amounts of dissipation significantly affect the transition to equilibrium is just one problem that could be readily investigated with these methods. Accurately incorporating the effect of physical viscosity into refined grid methods will surely be an active area of research in this field in the near future and will help to better explain the role of viscosity in the transition to statistical equilibrium states. If refined grids methods are also extended to three-dimensional flows, high resolution studies of local phenomena in three dimensions such as the behavior of vortex lines will also be possible.

Perhaps more important than any single insight gained from the numerical experiments presented in this chapter is the more general and basic result that it is possible

to construct stable, uniformly accurate methods for solving the Euler equations on locally refined grids. It is particularly significant that, when taken as a whole, the problems presented illustrate that a uniformly accurate, efficient projection can be implemented on refined grids. Among the problems that will undoubtably be the subject of future research will be to extend refined grid projection methods to viscous three-dimensional flows in general boundaries, to flows with complex chemistry, and to methods developed for complicated computational domains such as overlapping, composite, or body fitted grids. It is hoped that the analysis of the projection on refined grids presented in this work will be an important contribution to the current research towards these goals.

Bibliography

[1] A. S. Almgren, J. B. Bell, P. Colella, and L. H. Howell. An adaptive projection method for the incompressible Euler equations. In *Proceedings of the Eleventh AIAA Computational Fluid Dynamics Conference*, pages 530–539. AIAA, June 1993.

[2] A. S. Almgren, J. B. Bell, and W. G. Szymczak. A numerical method for the incompressible Navier-Stokes equations based on an approximate projection. LLNL unclassified report UCRL-JC-112842, Lawrence Livermore National Laboratory, 1993.

[3] Ann Stewart Almgren. *A Fast Adaptive Vortex Method Using Local Corrections.* PhD thesis, University of California, Berkeley, May 1991.

[4] Christopher R. Anderson. Domain decomposition techniques and the solution of poisson's equation in infinite domains. In *Proceedings of the Second International Symposium on Domain Decomposition Methods*, pages 129–139, 1988.

[5] Christopher R. Anderson. Vorticity boundary conditions and boundary vorticity generation for two-dimensional viscous incompressible flows. *J. Comp. Phys.*, 80:72–97, 1989.

[6] V. I. Arnold. *Mathematical Methods of Classical Mechanics.* Springer-Verlag, New York, 1988.

[7] J. B. Bell, M. J. Berger, J. S. Saltzman, and M. Welcome. Three dimensional adaptive mesh refinement for hyperbolic conservation laws. LLNL unclassified report UCRL-JC-108794, Lawrence Livermore National Laboratory, 1991.

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

[9] J. B. Bell, P. Colella, and L. H. Howell. An efficient second-order projection method for viscous incompressible flow. In *Proceedings of the Tenth AIAA Computational Fluid Dynamics Conference*, pages 360–367. AIAA, June 1991.

[10] J. B. Bell and D. L. Marcus. A second order projection method for variable density flows. *J. Comp. Phys.*, 101(1):1–24, 1992.

[11] M. J. Berger. *Adaptive Mesh Refinement For Hyperbolic Partial Differential Equations.* PhD thesis, Stanford University, 1982.

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

[13] M. J. Berger and A. Jameson. Automatic adaptive grid refinement for the euler equations. *AIAA J.*, 23:561–568, 1985.

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

[15] W. L. Briggs. *A Multigrid Tutorial.* SIAM, Philadelphia, PA, 1987.

[16] D. L. Brown and M. Minion. Performance of underresolved two-dimensional incompressible flow simulations. LANL unclassified report LA-UR-94-9999, Los Alamos National Laboratory, 1994.

[17] Jacob Burbea. Motions of vortex patches. *Letters in Mathematical Physics*, 6:1–16, 1982.

[18] T. Buttke. A fast adaptive vortex method for patches of constant vorticity in two dimensions. *JCP*, 89:161–186, 1990.

[19] A. J. Chorin. Numerical solution of the Navier-Stokes equations for an incompressible fluid. *Bull. of the American Math. Soc.*, 73:928–931, 1967.

[20] A. J. Chorin. Numerical solution of incompressible flow problems. *Studies in Numerical Analysis*, 2:64–71, 1968.

[21] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Math. Comp.*, 22:742–762, 1968.

[22] A. J. Chorin. On the convergence of discrete approximations to the Navier-Stokes equations. *Math. Comp.*, 23:341–353, 1969.

[23] A. J. Chorin. *Vorticity and Turbulence.* Springer-Verlag, New York, 1993.

[24] A. J. Chorin and J. E. Marsden. *A Mathematical Introduction to Fluid Mechanics.* Springer-Verlag, New York, 1979.

[25] P. Colella. Multidimensional upwind methods for hyperbolic conservation laws. *J. Comp. Phys.*, 87:171–200, 1990.

[26] S. D. Conte and Carl de Boor. *Elementary Numerical Analysis.* McGraw-Hill, New York, NY, 1980.

[27] David D. Dritschel. Contour surgery: A topological reconnection scheme for extended integrations using contour dynamics. *Journal of Computationl Physics*, 77, 1988.

[28] David D. Dritschel. The repeated filamentation of two-dimensional vorticity interfaces. *J. Fluid Mech.*, 194:511–546, 1988.

[29] David D. Dritschel. Contour dynamics and contour surgery: Numerical algorithms for extended, high-resolution modelling of vortex dynamics in two dimensional, inviscid, incompressible flows. *Comput. Phys. Rep.*, 10:77, 1989.

[30] Weinan E and Chi-Wang Shu. A numerical resolution study of high order essentially non-oscillatory schemes applied to incompressible flow. *J. Comp. Phys.*, 110:39–46, 1994.

[31] P. M. Gresho. On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. *Int. J. Numer. Methods Fluids*, 11:587, 1990.

[32] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluids with free surfaces. *Physics of Fluids*, 8(12), 1965.

[33] W. D. Henshaw, H.-O. Kreiss, and L.G.M. Reyna. On the smallest scale for the incompressible Navier-Stokes equations. *Theoretical and Computational Fluid Dynamics*, 1:65–95, 1989.

[34] T.Y. Hou and B.T.R. Wetton. Convergence of a finite difference scheme for the Navier-Stokes equations using vorticity boundary conditions. *Theoretical and Computational Fluid Dynamics*, 1:65–95, 1989.

[35] L. H. Howell. A multilevel adaptive projection method for unsteady incompressible flow. In *6th Copper Mountain Conference on Multigrid Methods*, April 1993.

[36] Fritz John. *Partial Differential Equations.* Springer-Verlag, New York, 1982.

[37] S. Kida. Motion of an elliptical vortex in a uniform shear flow. *Journal of The Physical Society of Japan*, 50:35–7–3520, 1981.

[38] O. A. Ladyzhenskaya. *The Mathematical Theory of Viscous Incompressible Flow.* Gordon and Breach, New York, 1963.

[39] M. F. Lai. *A Projection Method for Reacting Flow in the Zero Mach Number Limit.* PhD thesis, University of California, Berkeley, September 1993.

[40] M. F. Lai, J. Bell, and P. Colella. A projection method for combustion in the zero Mach number limit. In *Proceedings of the Eleventh AIAA Computational Fluid Dynamics Conference*, pages 776–783. AIAA, June 1993.

[41] D. Montgomery, W. Matthaeus, W. Stribling, D. Martinez, and S. Oughton. Relaxation in two dimensions and the "sinh-poisson" equation. *Phys. Fluids*, A, 4:3–6, 1992.

[42] R. Peyret and T. D. Taylor. *Computational Methods for Fluid Flow.* Springer-Verlag, New York, 1983.

[43] L. Quartapelle. Projection conditionins on the vorticity in viscous incompressible flows. *Int. J. Num. Math. Fluids*, 1:129–144, 1981.

[44] L. Quartapelle. Vorticity conditioning in the computation of two-dimensional viscous flows. *J. Comp. Phys.*, 40:453–477, 1981.

[45] Mark Reider. *Development of Higher Order Numerical Methods for Two-Dimensional Incompressible Flow With Applications to Flow Around Circular Cylinders and Airfiols.* PhD thesis, University of California, Los Angeles, 1992.

[46] G. Russo and J. Strain. Fast triangulated vortex methods for the two-dimensional euler equatios. *J. Comp. Phys.*, 111:291–323, 1994.

[47] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *J. Comp. Phys.*, 77:439–471, 1988.

[48] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes,II. *J. Comp. Phys.*, 83:32–78, 1989.

[49] J. C. Simo and F. Armero. Unconditional stability and long-term behavior of transient algorithms for the incompressible Navier-Stokes and Euler equations. *Comp. Meth. Appl. Mech. Eng.*, 111:111–154, 1994.

[50] B. Turkington and N. Whitaker. Variational approach to steady flow in two dimensions. to appear.

[51] Hong Yun Wang. A high order vortex method for patches of constant vorticity in two dimensions. Report PAM-534, University of California Department of Mathematics, 1991.

[52] N.J. Zabusky, M. H. Hughes, and K. V. Roberts. Contour dynamics for the Euler equations in two dimensions. *J. Comp. Phys.*, 30:96, 1979.

DATE
FILMED
8/11/94
END