A Fast Adaptive Vortex Method using Local Corrections

By

Ann Stewart Almgren

B.A. (Harvard University) 1984
M.S. (University of California) 1987

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ENGINEERING
Mechanical Engineering

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA at BERKELEY

Approved:

Chair: ..... 4/25/91 .....
Date
..... 4/15/91 .....
..... 4/22/91 .....

* *

A Fast Adaptive Vortex Method using Local Corrections

by

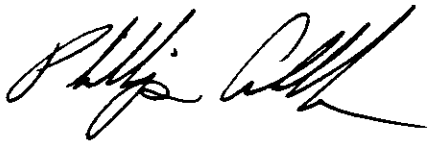
Ann S. Almgren

Abstract

Vortex methods are particle methods for solving time-dependent incompressible flow problems by discretizing the vorticity into vortex elements and following these elements in time. The main difficulty with vortex methods as originally formulated is that the cost of the evaluation of the velocity field induced by N vortices is $O(N^2)$. This is expensive, particularly in three dimensions where the number of elements can increase rapidly in time due to vortex stretching. Several methods have been developed for reducing this cost by exploiting the fact that the velocity induced by a vortex element is harmonic; the method of local corrections (MLC) is one such method.

The MLC is a particle-particle particle-mesh method, in which the calculation of the velocity field induced by a collection of vortices is split into two parts: (i) a finite difference velocity field calculation using a fast Poisson solver on a uniform grid superimposed on the vorticity field, and interpolation of that velocity from the grid points to the vortices; (ii) a local corrections step in which local interactions are computed directly. We present a fast adaptive vortex method which adds adaptive mesh refinement to the MLC.

In many calculations the vorticity is concentrated in subregions of the domain. By adding adaptive mesh refinement, which allows for finer grids in regions of concentrated vorticity, we reduce the number of local corrections where the savings are largest without introducing unnecessary additional grid points in regions of few vortices. Among the achievements of this work are the development of an efficient multigrid Poisson solver on an adaptively refined mesh, a hierarchical version of the local corrections algorithm, and an optimal refinement algorithm for adaptively choosing the local mesh density so as to minimize the cost of the algorithm for a given distribution of vorticity. This MLC with AMR has been implemented in two and three space dimensions. Calculations with a vortex ring in three dimensions show the breakeven point between the MLC with AMR and the direct method is $N \approx 3000$; for $N \approx 64,000$ MLC with AMR is approximately twelve times faster than the direct method.



Dedication

I dedicate this thesis to the many people who have encouraged me and supported me and taught me through the years. I would like especially to recognize two teachers, in whose classrooms I had the privilege to be a student: Mr. Jim Messersmith, a science teacher at John Witherspoon Middle School in Princeton, New Jersey; and Mr. Bob Komada, a physics teacher at Princeton High School.

I took middle school science from Mr. Messersmith ("Mr. M.") for three years. He taught me not just to ask the right questions, but how to look for the answers myself. Bob Komada was my high school physics teacher for two years. He valued thinking and learning, not only in his students but in himself, and his example was contagious. These teachers have helped me become who I am today, and I thank them.

Contents

Dedication ii

Table of Contents iii

List of Figures vi

List of Tables x

Acknowledgements xii

Introduction 1

1 Standard Vortex Method 4

1.1 The Navier-Stokes Equations 4

1.2 Vorticity - Stream Function Formulation 5

1.3 Standard Vortex Method 7

1.3.1 Two Dimensions 7

1.3.2 Three Dimensions 12

2 Review of Previous Work 19

2.1 History of the Standard Vortex Method 19

2.2 Fast Vortex Methods 21

2.2.1 Cloud-in-Cell Method 21

2.2.2 PPPM Method 22

2.2.3 Method of Multipoles 23

3 Method of Local Corrections (MLC) 26

3.1 Description of the Algorithm 26
3.2 Parameter Analysis 32
3.3 Comparison of Fast Vortex Methods 40

4 MLC with Adaptive Mesh Refinement **45**
4.1 Adaptive Mesh Refinement 45
4.2 MLC with AMR 50
4.3 Error and Timing Results 56
4.4 Optimal Refinement Criteria 59

5 Grid-Related Topics **66**
5.1 Finite Difference Stencils 66
5.1.1 Discrete Laplacian 66
5.1.2 Interpolation Stencils 67
5.2 Multigrid 71
5.3 Multigrid with AMR 75
5.4 Boundary Conditions 77
5.4.1 Infinite Domain 78
5.4.2 Solid Wall (No Flow) on Regular Domain 81
5.4.3 Fully Periodic 82
5.4.4 Walls with Periodic 83
5.4.5 Mixed Periodic-Infinite Domain Conditions 84
5.4.6 Mixed Wall-Infinite Domain Conditions 90

6 Three Dimensional Results: Stability of a Vortex Ring **92**
6.1 Vortex Rings 92

6.2 Presentation of Data 95

6.2 Summary of Results 97

6.2 Individual Calculations 99

6.3 Discussion and Conclusions 108

References **131**

List of Figures

3.1 Grid showing $B^{\mathbf{i}}, R^{\mathbf{i}}, R_0^{\mathbf{i}}$ for creation of $\mathbf{g}^{\mathbf{i}}$ in two dimensions. 44

3.2 Grid showing $B^{\mathbf{i}}, S^{\mathbf{i}}$ for local corrections in two dimensions. 44

4.1 Sample composite grid at level 2. 65

6.1 (a) Vortex ring and axes. (b) Locations of vortex filaments in the core for discretizations KG:II and KG:III. 112

6.2 Ring with $N = 2040$ (Mesh KG:II), $n = 9$ perturbation, at times $t = 0, 10, 40, 70, 100$, calculation done using MLC. Shown at angle $\pi/3$ from the z-axis. 113

6.3 (a) Amplitude vs. wavenumber for ring with $n = 9$ perturbation at $t = 0, 10, 40, 70, 100$; (b) time evolution of $n = 9$ mode from $t = 0$ to $t = 100$. Discretization as in Figure 6.2. 114

6.4 Ring with $N = 2040$ (Mesh KG:II), $n = 12$ perturbation, circulation calculated by solving the system of linear equations (6.1), at times $t = 30, 60, 90, 120$, calculation done using MLC. Shown at angle $\pi/3$ from the z-axis. 115

6.5 Ring from Figure 6.4 at time $t = 140$ shown at angles $0, \pi/3, \pi/2$ from the z-axis. 116

6.6 (a) Amplitude vs. wavenumber at $t = 0, 30, 60, 90, 120, 140$, for ring with $N = 2040$, $n = 12$ perturbation; (b) time evolution of $n = 12$ mode. Discretization as in Figure 6.4. 117

6.7 Intersection of 17 filaments with the $\theta = 0$ plane for ring with $N = 2040$, $n = 12$ at times $t = 0, 10, 20, 30, 40, 50, 60, 70$; discretization as in Figure 6.4. 118

6.8 (a) Ring with 17 filaments, $N = 2040$ (Mesh KG:II) at $t = 40$. Calculation done using MLC. (b) Ring with 61 filaments, $N = 7320$ (Mesh KG:III) at $t = 40$. Calculation done using MLC with AMR. Both have $n = 12$ perturbation, circulation calculated by solving the system of linear equations (6.1). 119

6.9 Time evolution of $n = 12$ mode for calculations of ring with $N = 7320$ using (a) MLC with AMR, (b) direct method; discretization as in Figure 6.8b. 120

6.10 (a) Amplitude vs. wavenumber for calculations of $N = 7320$ ring with $n = 12$ perturbation, at times $t = 0, 5, 10, 15, 20, 25, 30, 35, 40$, using MLC with AMR. Discretization as in Figure 6.8b. 121

6.10 (b) Amplitude vs. wavenumber for calculations of $N = 7320$ ring with $n = 12$ perturbation, at times $t = 0, 5, 10, 15, 20, 25, 30, 35, 40$, using the direct method. Discretization as in Figure 6.8b. 122

6.10 (c) Amplitude vs. wavenumber for calculations of $N = 7320$ ring with $n = 12$ perturbation, at times $t = 0, 5, 10, 15, 20, 25, 30, 35, 40$, using MLC on a uniform grid. Discretization as in Figure 6.8b. 123

6.11 Amplitude vs. wavenumber for calculations of ring with $N = 15600$ (Mesh KG:II), $n = 12$ perturbation, circulation calculated as product of point value of vorticity and area represented by the filament, at time $t = 0$ and (a) at $t = 10$ using MLC with AMR on 8-32 grid, (b) at $t = 10$ using MLC with AMR on 16-32 grid, and (c) at $t = 10$ using the direct method. 124

6.12 (a) Amplitude vs. wavenumber for calculations of a ring with N = 15600, $n = 12$ perturbation, at times $t = 0, 10, 20, 30, 40$, using MLC with AMR on a 8-64 grid. Discretization as in Figure 6.11. 125

6.12 (b) Amplitude vs. wavenumber for calculation of a ring with N = 15600, $n = 12$ perturbation, at times $t = 0, 10, 20, 30, 40$, using MLC with AMR on a 8-32 grid. Discretization as in Figure 6.11. 126

6.13 (a) Amplitude vs. wavenumber for calculations of a ring with N = 2280 (Mesh KG:III), $n = 12$ perturbation, circulation calculated by integrating the vorticity over the area represented by each filament (Equation 6.3), at times $t = 0, 10, 20, 30, 40, 50, 60, 70, 80$, using MLC and the direct method. (b) Time evolution of $n = 12$ mode for calculations using MLC and the direct method. 127

6.14 Intersection of 19 filaments with the $\theta = 0$ plane for ring with N = 2280, $n = 12$ perturbation, at times $t = 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75$. Calculation done using MLC; discretization as in Figure 6.13. 128

6.15 (a) Amplitude vs. wavenumber for calculation of ring with N = 14640 (Mesh KG:III), $n = 12$ perturbation, circulation calculated by integrating the vorticity over the area represented by each filament (Equation 6.3), at times $t = 0, 10, 20, 30, 40, 50, 60, 70$, using MLC with AMR (8-64) with $C = D = 2.5$. This is contrasted with the $N = 2280$ MLC calculation. (b) Time evolution of $n = 12$ mode for the calculation with $N = 14640$ using MLC with AMR and the calculation with $N = 2280$ using MLC. 129

6.16 Intersection of 61 filaments with the $\theta = 0$ plane for ring with $N = 14640, n = 12$ perturbation, at times $t = 0, 5, 10, 15, 20, 25, 30, 35, 40, 45$ $50, 55, 60, 65, 70$. Discretization as in Figure 6.15. 130

List of Tables

3.1 Relative point error in u at $(.75, .75)$ due to a single vortex of unit strength at $(.505, .505)$ as a function of spreading distance D and mesh spacing h. 35

3.2 Relative L_2 norm of error in u due to a single vortex of unit strength at $(.505, .505)$ as a function of spreading distance D and mesh spacing h. 35

3.3 Relative point error in u at $(.75, .75)$ due to a patch of vorticity centered at $(.5, .5)$, radius .2, as a function of spreading distance D and mesh spacing h. 36

3.4 Relative L_2 norm of error in u due to a patch of vorticity centered at $(.5, .5)$, radius .2, as a function of spreading distance D and mesh spacing h. 36

3.5 Relative L_2 norm of error in velocity at the vortex locations. Parameters are as in Table 3.3. 37

3.6 L_2 norm of error in vortex locations at time $t = 1$ using two different core functions. Anderson [1]. 38

3.7 L_2 norm of error in vortex locations after finite time as a function of number of vortices N, mesh spacing h, and time step Δt .Baden [6]. 39

4.1 Relative L_2 norm of error for different levels of mesh refinement. Calculations for a three-dimensional vortex ring of radius .1 around the z-axis and cross-sectional radius .02. 57

4.2 Timings for a single velocity evaluation using the direct method, MLC and MLC with AMR for calculations of a three-dimensional vortex ring. Timings are on a Cray Y-MP with the cft77 compiler. 58

4.3 Timings for a single velocity evaluation with different possible refinements of the grid for calculations of a three-dimensional vortex ring with $N = 17641$. Timings are on a Cray Y-MP with the cft77 compiler. 60

4.4 Operation count for the MLC on a uniform grid with M total grid points, M_v bins containing vortices, and N vortices. 60

4.5 Time per operation in the MLC, with three different levels of AMR and for optimal uniform grid case. Timings are on a Cray Y-MP with the cft77 compiler. 60

6.1 Parameters used in vortex ring calculations. 113

Acknowledgements

I am deeply indebted to my thesis advisor, Phil Colella. His teaching and guidance, encouragement and support throughout the time I have worked with him have been invaluable. I thank him for being my mentor.

I would like to thank Alexandre Chorin and Phil Marcus for their timely and thorough reading of this thesis, and for their many constructive comments.

Thanks to all in the CFD group who have made the CFD experience fun; thanks especially to Mindy Fruchtman for her support, and to Scott Dudek for his sense of humor. Thanks to Margo Seltzer and Marie des Jardins for all the lunches and good conversation and support. I don't know how I would have gotten through graduate school without the daily e-mail from Ann Trenk.

Chris Anderson, Omar Knio, and the members of the Applied Mathematics Group at Lawrence Livermore National Laboratory have each taken time to answer questions and work with me, for which I am very appreciative. Thanks to Tom Buttke, on whose initial implementation of the uniform grid MLC in three dimensions this work is based. Thanks also to John Bell and Michael Welcome of LLNL for their help in implementing the details of AMR.

I gratefully acknowledge the financial support over the past six years from: a Graduate Opportunity Fellowship from U.C. Berkeley; a National Science Foundation Graduate Fellowship; Army Research Office grant DAALO3-88-K-0197; DARPA and National Science Foundation grant DMS-8919074; and summer support from the Applied Mathematics Group at LLNL.

Computational support for the calculations presented in this thesis was provided

by the San Diego Supercomputer Center and the Center for Computational Sciences at the NASA Goddard Space Flight Center. I gratefully acknowledge the computing time made available by both centers.

I thank all of my (extended) family, who have never once stopped believing in me, often more than I believed in myself.

And finally, many thanks to Shawn, whose love has helped me through.

Introduction

Vortex methods are used to approximate time-dependent incompressible flows. They are particle methods based on the Lagrangian formulation of the flow equations, in which the quantity carried by the flow is vorticity. The configuration of vortex elements at a given time determines the velocity field via the Biot-Savart law; the velocity is then used to update the positions of the vortices. In three dimensions, the vorticity itself must be updated as well. Vortex methods are especially useful for flows which are dominated by the presence of vorticity, e.g., shear flows, wakes and jets. In these flows most of the vorticity is confined to a relatively small portion of the flow, and then a method based on the vorticity can be very economical.

Point vortex methods were first introduced by Rosenhead in 1935 [50]. A general vortex method for simulating high Reynolds number flow was developed for two dimensions by Chorin [22], and for three dimensions by Chorin [23] and Leonard [40]. Convergence of these methods has been extensively studied, and proven for a variety of cases [5, 8, 9, 10, 28, 29, 31, 34, 35, 47].

The usefulness of vortex methods has been seriously limited in the past by their cost. The accuracy of the methods and their ability to resolve small scales increase with the number of particles, N, as does the time and expense. The cost of standard vortex methods is $O(N^2)$, making them prohibitively expensive for relatively few vortices (on the order of thousands). Fast vortex methods have been developed to try to maintain the accuracy and adaptivity of the standard vortex method while increasing their speed. These fast methods approximate the $O(N^2)$ velocity calculation with a fast calculation whose cost is $O(N \log N)$ for large N.

This thesis presents an extension of a fast vortex method known as the Method of Local Corrections (MLC). In the MLC, developed by Anderson [1] in two dimensions and extended to three dimensions by Buttke and Colella, one introduces a uniform grid that covers the computational domain enclosing the vortices, and calculates the velocity field due to the vortices on that grid. A corrected form of this velocity is then interpolated onto the vortices, and local interactions are calculated directly. While the MLC is faster than the standard vortex methods for N greater than several thousand, there is further efficiency to be gained by introducing adaptive mesh refinement (AMR) [11, 12] to the grid. This thesis implements the addition of AMR to the MLC, and shows that, especially in three dimensions, there is significant reduction of cost with no loss of accuracy. For example, for a vortex ring in three dimensions with $N \approx 64,000$, the speedup of the MLC with AMR over the uniform MLC is approximately three, the speedup of MLC with AMR over the standard vortex method is approximately twelve.

The thesis is divided into six chapters. In Chapter 1 we present the equations governing incompressible fluid flow and their formulation for use in vortex methods. We then introduce the standard vortex method, in two and three space dimensions. Chapter 2 contains a review of past work. The first section presents the history of the standard vortex method, the second section introduces several different fast vortex methods.

Chapter 3 introduces the method of local corrections in two and three dimensions, and presents an analysis of the parameters affecting the accuracy of the method. A comparison is made with the fast vortex methods introduced in Chapter 2. In Chapter 4 we introduce the concept of adaptive mesh refinement for

finite difference calculations, and then discuss the application of this technique to the method of local corrections. Error and timing results for MLC with AMR are presented here.

Chapter 5 describes the grid-related details of the final algorithm, including the stencils for the discrete Laplacian and for interpolation in two and three dimensions, standard multigrid, and multigrid with mesh refinement. Boundary conditions are discussed here.

In Chapter 6 we present the results from calculations of a three dimensional vortex ring. First we validate the new algorithm by comparison with calculations done using the direct method, next we investigate the stability of the ring using different discretizations and numbers of vortices. A discussion of the findings and suggestions for future work conclude this chapter.

CHAPTER 1

Standard Vortex Method

1.1 The Navier-Stokes Equations

Incompressible fluid flow is governed by the equations

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} &= -\frac{1}{\rho}\nabla p + \nu \Delta \mathbf{u}, \\ \nabla \cdot \mathbf{u} &= 0. \end{aligned}$$

These equations describe the velocity $\mathbf{u}$ and pressure p of a general incompressible fluid with kinematic viscosity $\nu = \mu/\rho$, where μ is the viscosity and ρ the density. We nondimensionalize these equations by setting

$$\begin{aligned} u' &= u/U \\ t' &= (U/L)\, t \\ x' &= x/L \\ p' &= p/\rho U^2 \end{aligned}$$

where L is a length scale of the problem to be considered, and U is a typical velocity scale. We then get the non-dimensional equations (now removing the primes):

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \frac{1}{\mathrm{Re}} \Delta \mathbf{u} \tag{1.1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{1.2}$$

where $\mathrm{Re} = UL/\nu$ is the Reynolds number, the ratio of the inertial to viscous forces in the flow. The Reynolds number measures the significance of the viscous term in the equation; equivalently, the importance of viscosity in the physical flow.

The first term in the Navier-Stokes equations is the time derivative of velocity, measuring the rate of change of velocity at a fixed position $\mathbf{x}$. The second term is the *advection* term, and describes the change of $\mathbf{u}$ at $\mathbf{x}$ resulting from the fact that the particles of the fluid are moving past $\mathbf{x}$. The first two terms together define the material derivative $D\mathbf{u}/Dt$, and represent the change in $\mathbf{u}$ at a fixed *particle* moving in the flow.

1.2 Vorticity - Stream Function Formulation

In the rest of this work we consider only inviscid flow, defined by $\mathrm{Re} = \infty$, so that the viscous term $\frac{1}{\mathrm{Re}}\Delta\mathbf{u}$ makes no contribution. Equations (1.1) and (1.2) are four equations for four unknowns, the pressure and three velocity components. These equations can be rewritten in terms of vorticity $\omega = \nabla \times \mathbf{u}$, by taking the curl of both sides of the equation (1.1):

$$\frac{\partial \omega}{\partial \mathbf{t}} + (\mathbf{u} \cdot \nabla)\omega - (\omega \cdot \nabla)\mathbf{u} = \mathbf{0} \tag{1.3}$$

Note that the pressure term is removed because the curl of a gradient is identically

zero. To complete the set of equations we include the definition of ω,

$$\omega = \nabla \times \mathbf{u}. \tag{1.4}$$

This is now a system of six equations and six unknowns, $\mathbf{u}$ and ω. Recalling $D/Dt = \partial/\partial t + (\mathbf{u} \cdot \nabla)$, we can write (1.3) as

$$\frac{D\omega}{Dt} - (\omega \cdot \nabla)\mathbf{u} = \mathbf{0} \tag{1.5}$$

The first term here is the material derivative of the vorticity; the second is known as the *stretching* term. This formulation is the basis of the three-dimensional vortex method, as will be seen in Section 1.3.

The incompressibility of $\mathbf{u}$ allows us to write $\mathbf{u} = \nabla \times \mathbf{\Psi}$, where $\mathbf{\Psi}$ is uniquely determined by $\mathbf{u}$ to within the gradient of a scalar ζ. The curl of a gradient is identically zero, so ζ never enters into the equation for $\mathbf{u}$; thus we are free to choose ζ so that $\nabla \cdot \mathbf{\Psi} = \mathbf{0}$.

Equations (1.4) and (1.5) define ω in terms of $\mathbf{u}$ and describe the evolution of ω in time; we also need an expression for $\mathbf{u}$ in terms of ω . The inversion of $\omega = \nabla \times \mathbf{u}$ can be achieved by expanding ω in terms of $\mathbf{\Psi}$,

$$\omega = \nabla \times \mathbf{u} = \nabla \times (\nabla \times \mathbf{\Psi}) = \nabla(\nabla \cdot \mathbf{\Psi}) - \mathbf{\Delta \Psi}.$$

Since we are free to set $\nabla \cdot \mathbf{\Psi} = \mathbf{0}$, we see that $-\Delta \mathbf{\Psi} = \omega$. Thus, knowing ω, one can solve this Poisson equation for $\mathbf{\Psi}$ (with appropriate boundary conditions on $\mathbf{\Psi}$), and find $\mathbf{u}$ as the curl of $\mathbf{\Psi}$. The existence of $\mathbf{\Psi}$ and the equations relating $\mathbf{\Psi}$, ω, and $\mathbf{u}$, allow us to work with velocity or vorticity fields interchangeably.

Now consider a velocity field lying in the x-y plane and dependent only on x and y, i.e., $\mathbf{u}(\mathbf{x}) = \mathbf{u}(x, y)$ and $\mathbf{u} \cdot \mathbf{e}_z = 0$. The curl of this field is everywhere parallel to the z-axis, thus we write $\omega = \omega \mathbf{e_z}$, and we may in this case think of vorticity as the scalar value ω. We can immediately see that the term $\omega \cdot \nabla \mathbf{u}$ must be zero, since $\omega \cdot \nabla \mathbf{u} = \omega \mathbf{e_z} \cdot \nabla \mathbf{u} = \omega \frac{\partial \mathbf{u}}{\partial \mathbf{z}}$, and $\frac{\partial \mathbf{u}}{\partial z} = 0$ by construction. The equations for two dimensional flows reduce to

$$\frac{D\omega}{Dt} = \frac{\partial \omega}{\partial \mathrm{t}} + (\mathbf{u} \cdot \nabla)\omega \quad = \quad 0 \tag{1.6}$$

$$\omega \quad = \quad \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \tag{1.7}$$

In two dimensions the vector stream function Ψ reduces to a scalar stream function ψ. The equations

$$u = \frac{\partial \psi}{\partial y}, v = -\frac{\partial \psi}{\partial x},$$

$$\frac{\partial \omega}{\partial t} + u\frac{\partial \omega}{\partial x} + v\frac{\partial \omega}{\partial y} = 0,$$

$$-\Delta \psi = \omega$$

define the vorticity-stream function formulation in two dimensions. Note that the boundary conditions for this problem will most likely be given as the normal velocity, or the tangential derivative of ψ, hence ψ itself, at the boundary. This vorticity-stream function formulation is the basis for several different fast vortex methods, as will be discussed in chapter 2.

1.3 The Standard Vortex Method

1.3.1 Two Dimensions

We first consider the standard vortex method in two dimensions. Recall equation (1.6),

$$\frac{D\omega}{Dt} = 0,$$

i.e., the material derivative of the vorticity is zero. Thus vorticity is a scalar quantity carried along particle paths, which suggests a computational method based on tracking particles and thereby evolving the vorticity field associated with those particles.

Define the flow map $\mathbf{x} : \Re^2 \times [0,T] \rightarrow \Re^2$, so that $\mathbf{x}(\alpha,t)$ is the position at time t of the fluid particle which at time $t = 0$ was at position α. Then $\mathbf{x}$ satisfies

$$\frac{d\mathbf{x}}{dt}(\alpha,t) = \mathbf{u}(x(\alpha,t),t). \tag{1.8}$$

To find the velocity given the vorticity, recall that in two dimensions

$$u = \frac{\partial\psi}{\partial y}, v = -\frac{\partial\psi}{\partial x},$$

$$-\Delta\psi = \omega.$$

The infinite-domain Green's function of the two-dimensional Laplacian is

$$G(\mathbf{x}) = -\frac{1}{2\pi}\log\sqrt{x^2+y^2},$$

so the stream function at $\mathbf{x} = (x,y)$ due to a point vortex of strength ω_0 at $\mathbf{x}_0 =$

(x_0, y_0) is

$$\psi(\mathbf{x}) = -\frac{\omega_0}{2\pi} \log \sqrt{(x - x_0)^2 + (y - y_0)^2},$$

and the velocity field is

$$u(x, y) = \frac{\partial \psi}{\partial y} = -\frac{\omega_0}{2\pi} \frac{(y - y_0)}{(x - x_0)^2 + (y - y_0)^2}$$

$$v(x, y) = -\frac{\partial \psi}{\partial x} = \frac{\omega_0}{2\pi} \frac{(x - x_0)}{(x - x_0)^2 + (y - y_0)^2}.$$

More generally, the solution to $-\Delta\psi = \omega$ for any $\omega(\mathbf{x})$ is $\psi = G * \omega$, where $*$ is the convolution operator, i.e.,

$$\psi(\mathbf{x}) = \int_{\Re^2} G(\mathbf{x} - \mathbf{x}')\, \omega(\mathbf{x}') d\mathbf{x}'.$$

We define the kernel $\mathbf{K}(\mathbf{x})$ as the vector with components $(\frac{\partial G}{\partial y}, -\frac{\partial G}{\partial x})$ so that we can write $\mathbf{u} = \mathbf{K} * \omega$, or

$$\mathbf{u}(\mathbf{x}) = \int_{\Re^2} \mathbf{K}(\mathbf{x} - \mathbf{x}')\, \omega(\mathbf{x}') d\mathbf{x}'.$$

Consider now an initial vorticity field $\omega(\alpha, 0)$. Since the vorticity moves with the fluid, we can write

$$\frac{d\omega}{dt}(\mathbf{x}(\alpha, t), t) = 0, \tag{1.9}$$

The original vortex method as developed in 1935 [50] discretized the vorticity field as:

$$\omega(\mathbf{x}, t) = \sum_{i=1}^{N} \omega_i(t)\, \delta(\mathbf{x} - \mathbf{x}_i(t)), \tag{1.10}$$

where $\mathbf{x}_i(t)$ and ω_i are the position at time t and circulation, respectively, of the i^{th} vortex, and $\delta(\mathbf{x})$ is the Dirac delta function. Substitution of (1.10) into equations (1.8) and (1.9) yields equations describing the evolution of ω_i and $\mathbf{x}_i$:

$$\frac{d\mathbf{x}_i}{dt}(t) = \mathbf{u}(\mathbf{x}_i, t) = \sum_{j=1, j\neq i}^{N} \mathbf{K}(\mathbf{x}_i(t) - \mathbf{x}_j(t))\, \omega_j(t)$$

and

$$\frac{d\omega_i}{dt}(t) = 0.$$

Both $G(\mathbf{x})$ and $\mathbf{K}(\mathbf{x})$ as presented above are singular at $|\mathbf{x}| = 0$, which leads to an ill-conditioned numerical method [38]. Various techniques have been used to desingularize the kernel; we describe here the "vortex blob" introduced by Chorin [22]. The smoothing is achieved by replacing the Dirac delta function in the initial discretization by $f_\delta(r)$, where $f_\delta(r)$ is the core shape function. Thus we write

$$\omega(\mathbf{x}, t) = \sum_{i=1}^{N} \omega_i(t)\, f_\delta(|\mathbf{x} - \mathbf{x}_i(t)|).$$

The equation governing $\mathbf{x}_i(t)$ then becomes

$$\frac{d\mathbf{x}_i}{dt}(t) = \sum_{j=1, j\neq i}^{N} (K * f_\delta)(\mathbf{x}_i(t) - \mathbf{x}_j(t))\, \omega_j(t) = \sum_{j=1, j\neq i}^{N} K_\delta(\mathbf{x}_i(t) - \mathbf{x}_j(t))\, \omega_j(t),$$

where $K_\delta = K * f_\delta$. The evolution of ω_i is unaffected.

In Chorin's method

$$f_\delta(r) = \begin{cases} 1/2\pi\delta r & \text{for } r < \delta \\ 0 & \text{for } r \geq \delta \end{cases},$$

where $r = \sqrt{x^2 + y^2}$. We require that the integral of the core function over the region of its support be unity; this explains the $2\pi\delta$ above. With this core function the velocity field at $\mathbf{x}$ due to a single vortex blob of circulation ω_0 at $\mathbf{x}_0$ is

$$u(x,y) = -\frac{\omega_0}{2\pi}(y - y_0)h(r), \tag{1.11}$$

$$v(x,y) = \frac{\omega_0}{2\pi}(x - x_0)h(r), \tag{1.12}$$

where $r = \sqrt{(x - x_0)^2 + (y - y_0)^2}$ and

$$h(r) = \begin{cases} 1/r^2 & \text{for } r \geq \delta \\ 1/\delta r & \text{for } r < \delta \end{cases}.$$

Thus in two dimensions the vortex method is composed simply of an initial discretization of the vorticity field, and a discrete time-stepping procedure to solve the evolution equation for $\mathbf{x}_i$.

There are several choices to make in discretizing the initial vorticity field. The first is how many vortices N to use, i.e., what the intervortex spacing h_v should be. The second is what core radius δ to choose; some calculations define $\delta = h_v^q, 0 < q < 1$, others let $\delta = Ah_v$ for some constant $A, 1 < A$. It has been well-established in two dimensions that the core radius should be larger than the intervortex spacing, but there are no definitive guidelines for choosing an optimal core radius.

There are several choices for how to determine the values of ω_i given $\omega(\mathbf{x}, 0)$, once one has chosen N and δ; the two most common are:

1) $\omega_i = \omega(\mathbf{x}_i(0))dA_i$, where dA_i is the area represented by the i^{th} vortex element;

2) $\omega_i = \int_{dA_i} \omega(\mathbf{x}, 0)d\mathbf{x}$.

Knio and Ghoniem [39] use a third approach, which is discussed further in Chapter 6.

Note that the ω_i carry the *circulation* of a patch of vorticity, not the vorticity itself.

To solve for the positions $\mathbf{x}_i(t)$ of the vortices, in our computations we use the following second-order time-stepping scheme:

$$\mathbf{x}_i^* = \mathbf{x}_i^n + \mathbf{u}(\mathbf{x}_i^n)\Delta t \tag{1.13}$$

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + .5(\mathbf{u}(\mathbf{x}_i^n) + \mathbf{u}(\mathbf{x}_i^*))\Delta t,$$

where $\mathbf{x}_i^n \approx \mathbf{x}_i(n\Delta t)$, $\mathbf{x}_i^{n+1} \approx \mathbf{x}_i((n+1)\Delta t)$, and $\mathbf{x}_i^*$ is an intermediate value.

1.0.1 Three Dimensions

Recall from section 1.2 the three dimensional evolution equation for vorticity:

$$\frac{D\omega}{Dt} = (\omega \cdot \nabla)\mathbf{u}. \tag{1.14}$$

In three dimensions vorticity is a vector, and is not preserved along particle trajectories. Thus we can no longer track constant-strength vortex blobs; there must be some mechanism for correctly evolving the vortex strengths.

As in two dimensions we discretize the original vorticity field into N nonsingular computational elements:

$$\omega(\mathbf{x}, t) = \sum_{i=1}^{N} \omega_i(t)\, f_\delta(|\mathbf{x} - \mathbf{x}_i(t)|). \tag{1.15}$$

Here ω_i and $\mathbf{x}_i(t)$ are the vector strength and location, respectively, of the i^{th} vortex

element, $f_\delta(r)$ is the core function used in [4],

$$f_\delta(r) = \begin{cases} \frac{3}{4\pi\delta^3} & \text{for } r < \delta \\ 0 & \text{for } r \geq \delta \end{cases}.$$

We again require that the integral of the core function over the region of its support be unity; this accounts for the $\frac{3}{4\pi\delta^3}$ seen in the expression.

Several different forms of this discretization have been developed to represent vorticity in three dimensions. One is the most obvious extension from two dimensions; again let the vortex element be a blob, but now with a vector strength. We can advect the positions of these blobs according to

$$\frac{d\mathbf{x}_i}{dt}(t) = \mathbf{u}(\mathbf{x}_i, t) = \sum_{j=1}^{N} \mathbf{K}_\delta(\mathbf{x}_i(t) - \mathbf{x}_j(t))\, \omega_j(t),$$

and we then need an expression for the evolution of $\omega_i(t)$. Anderson and Greengard [5] present two alternatives for the evolution of the vortex strengths:

$$\frac{d\omega_i}{dt} = \omega_i(t) \cdot \nabla \mathbf{u}(\mathbf{x}_i)$$

and

$$\omega_i(t) = (\nabla_\alpha \mathbf{x})\, \omega_i(0),$$

where $(\nabla_\alpha \mathbf{x})$ is the deformation matrix of the fluid, the derivatives of the present position with respect to the original position. The convergence of both methods has been shown as long as the flow is regular. The first of these is simply a result of the substitution of (1.15) into (1.14); for a derivation of the second see [26].

We choose a third approach: the representation of vorticity along line segments

rather than in spherically symmetric blobs:

$$\omega(\mathbf{x},t) = \sum_{i=1}^{N} \Gamma_i(t)\, \ell_i(t)\, f_\delta(|\mathbf{x} - \mathbf{x}_i^C(t)|),$$

where Γ_i, ℓ_i and $\mathbf{x}_i^C$ are now the circulation, vector length, and location of the center, respectively, of the i^{th} segment, and $\omega_i = \Gamma_i \ell_i$. This representation was developed by Chorin [23, 24, 25], and is similar to the vortex filament method developed by Leonard [40, 41]. In [40, 41], the vorticity is defined along space curves which are referred to as vortex filaments, and the geometry of these curves is evolved in time. In [23, 24, 25], the filaments are discretized into short, connected segments. This greatly simplifies the computational element, since each segment is assumed to stay uncurved (although it can certainly be reoriented), and the core undergoes no deformation. The effect of curvature of filaments is handled implicitly by allowing vortex segments to change orientation relative to each other while staying connected. This segment method was used in calculations in the works cited, and has been shown to converge in [31].

Initially, segment midpoints $\mathbf{x}_i^C$ are placed on a Lagrangian mesh on the support of the vorticity; each segment is then aligned along the direction of the vorticity at that point. The length of the segment is determined by the desired resolution of the calculation; the top and bottom of the segment, $\mathbf{x}_i^T$ and $\mathbf{x}_i^B$, are determined by

$$\mathbf{x}_i^T(0) - \mathbf{x}_i^B(0) = \ell_i(0),$$

$$\frac{1}{2}(\mathbf{x}_i^T(0) + \mathbf{x}_i^B(0)) = \mathbf{x}_i^C(0).$$

The circulation Γ_i of each segment is found as in two dimensions, as a product

of pointwise vorticity and cross-sectional area represented by the segment, or the integral of initial vorticity across that area. Thus these segments are pieces of vortex lines in the flow (vortex lines are simply defined as curves tangent to the vorticity). By the Kelvin Circulation Theorem we know that circulation around vortex lines is constant in time, and so the circulation of these computational elements can be held constant in time,

$$\frac{D\Gamma_i(t)}{Dt} = 0.$$

However, the lengths ℓ_i of the segments change as the endpoints move, and so the strength of each vortex evolves as well. This method differs from the two methods mentioned above in that the stretching is incorporated implicitly by the relative motion of the endpoints of each segment.

Since the divergence of the curl of a flow field is identically zero, we know that vortex lines cannot end in a flow, they must extend to infinity or end on a boundary in inviscid flow. In order to approximate this numerically, for our computations without boundaries (using infinite domain or periodic boundary conditions) we initialize the vorticity into segments connected in closed loops; for segments $i-1, i$ located on the same filament, $\mathbf{x}_i^B = \mathbf{x}_{i-1}^T$. Vortex segments, once connected, remain connected for all time, thus for N vortex segments there are only N rather than $2N$ independent endpoints.

One consequence of the variable lengths of the segments is that as the vorticity in a region of flow increases, the lengths of the segments may become disproportionately large. Thus, as a part of the algorithm, one must check at each time step whether the segment lengths have exceeded a preset critical length. If they have, we divide the long segments in half, giving each new vortex the same circulation as the original

vortex. This can result in the number of vortices growing rapidly as the calculation proceeds, and underscores the need for adaptive (in time) algorithms.

We find the velocity field induced by a single vortex element in the same manner as in the previous section. We know $\omega = \nabla \times \mathbf{u}$, and we can invert this expression using the stream function Ψ, to find $\mathbf{u}$ from ω. The infinite domain Green's function for the three dimensional Laplacian is

$$G(\mathbf{x}) = -\frac{1}{4\pi r}$$

where $r = \sqrt{x^2 + y^2 + z^2}$.

The kernel $\mathbf{K}(\mathbf{x}) = \nabla \times G$ satisfying $\nabla \times (G * \omega) = \mathbf{K} * \omega$ is now the matrix

$$\frac{1}{4\pi r^3} \begin{bmatrix} 0 & z & -y \\ -z & 0 & x \\ y & -x & 0 \end{bmatrix}$$

Again we desingularize the kernel, replacing $\mathbf{K}$ by $\mathbf{K}_\delta = \mathbf{K} * f_\delta(r)$, Then the velocity $\mathbf{u}$ is defined by $\mathbf{u} = \mathbf{K}_\delta * \omega$. The velocity field at $\mathbf{x}$ due to a single vortex segment with center at $\mathbf{x}_0$, circulation Γ and length $\ell = (\ell_x, \ell_y, \ell_z)$ is

$$\begin{aligned} u &= \frac{\Gamma}{4\pi}(\ell_y(z - z_0) - \ell_z(y - y_0))\, h(r) \\ v &= \frac{\Gamma}{4\pi}(\ell_z(x - x_0) - \ell_x(z - z_0))\, h(r) \\ w &= \frac{\Gamma}{4\pi}(\ell_x(y - y_0) - \ell_y(x - x_0))\, h(r) \end{aligned}$$

where

$$h(r) = \begin{cases} 1/r^3 & \text{for } r > \delta \\ (4 - 3\frac{r}{\delta})/\delta^3 & \text{for } r \leq \delta \end{cases}$$

In these calculations we evaluate the velocity at the endpoints, but the vorticity is assumed to be centered at the segment midpoints. The positions of the endpoints are evolved according to

$$\frac{d\mathbf{x}_i^T}{dt}(t) = \mathbf{u}(\mathbf{x}_i^T(t), t), \quad \frac{d\mathbf{x}_i^B}{dt}(t) = \mathbf{u}(\mathbf{x}_i^B(t), t),$$

where

$$\mathbf{u}(\mathbf{x}, t) = \sum_{j=1}^{N} \Gamma_j(t) \, (\mathbf{K}_\delta(\mathbf{x} - \mathbf{x}_j^C(t))\ell_j(t).$$

The locations of the center points are updated by

$$\mathbf{x}_i^C = \frac{1}{2}(\mathbf{x}_i^T + \mathbf{x}_i^B).$$

We use the same time-stepping procedure (1.13) as in two dimensions.

We should note here that this vortex segment method is not really so different from the first blob method presented above. Both change the strength of the vortex using the gradient of the velocity; the segment method defines the endpoints of the segments as material points in the flow, and thus the difference in $\mathbf{u}$ between the endpoints of a given vortex segment is just $\frac{1}{\Gamma}(\omega \cdot \nabla)\mathbf{u}$. In the blob method there is actually more freedom in how to evaluate the velocity gradient, and one can use a higher-order finite difference operator if one chooses. However, the blob method

requires computation of additional information about the velocity field.

CHAPTER 2

Review of Previous Work

2.1 History of the Standard Vortex Method

The standard vortex method in two dimensions was first developed in 1935 by Rosenhead [50], who approximated the motion of a two-dimensional vortex sheet by evolving in time the positions of point vortices. Point vortices have since been replaced as the computational element, since the system of point vortices is too singular to accuarately represent the physical vorticity field.

In the past twenty years, more sophisticated two-dimensional core functions have been substituted for the original point vortex. Chorin [22] was the first to introduce the two dimensional vortex blob method; present calculations all use vortex elements with nonsingular core functions.

Different algorithms have been suggested for handling the stretching term in the vorticity formulation of the three-dimensional Euler equations. Chorin first proposed the vortex segment method [23, 24, 25]; Leonard [40, 41] introduced an alternate method; Beale and Majda [8, 9] and Anderson and Greengard [5] present further formulations and comparisons with the previously suggested methods.

The convergence of vortex methods has been rigorously established in two and three dimensions for inviscid flow without boundaries. Proofs show higher order convergence depending on the smoothness of the core function. The first conver-

gence proof was given by Hald [34] who showed convergence to second order in the intervortex spacing for two-dimensional flow. Numerical experiments by Hald and del Prete [35] verified second-order convergence. Beale and Majda [8, 9] improved Hald's results by showing that the vortex method can converge with arbitrarily high-order accuracy, provided that the initial vorticity distribution is sufficiently smooth and that the core function satisfies certain moment conditions. They also extended a proof to three dimensions. Later, Beale and Majda [10] suggested a class of infinitely differentiable core functions which in theory provide higher order accuracy. Greengard [31] proved convergence of the three-dimensional filament method we use. Further contributions to the theory have been made by Cottet and Raviart [29], Cottet [28], and Anderson and Greengard [5]. A useful discussion of vortex methods is given in Leonard [40, 41], and in Anderson and Greengard [5].

Perlman [47] presents a study of the observed (as opposed to theoretical) accuracy of the two dimensional vortex method with different core functions. She concludes that, for radially symmetric vorticity with compact support, high-order core functions improve the accuracy of the vortex method as long as the flow remains smooth. See [47] for further details of the dependence of accuracy on the core function.

In choosing the core function we use, we keep in mind the comments of Anderson and Greengard [5], "... the improvement in accuracy in using a high order method over a lower order one becomes extremely small after the initial configuration of the particles has been sufficiently distorted."

Perlman also investigates the difference in methods of assigning vorticity to the blob, and concludes that Hald's choice [34] of assigning to each vortex element the

vorticity contained in the blob around it leads to second order accuracy for any core function; the approach chosen by Beale and Majda [8, 9] of assigning the value of the vorticity at the location of the blob times the area of the blob can provide higher order accuracy.

2.2 Fast Vortex Methods

Several fast vortex methods have been developed, relying on different formulations of the governing equations. These types of methods are not restricted to incompressible flow; they are in fact much more general techniques, used, for example, for galaxy simulations and in plasma physics. The goal of the fast vortex methods is to reduce the cost of the velocity calculation in the vortex method; this is done by exploiting local regularity in the elliptic differential equation solved to find the velocity from the vorticity. This local regularity can be used in two ways: first, it enables one to use fast methods to solve the Poisson equation; second, it allows one to represent the field away from the support of the right-hand side with a small number of computational degrees of freedom.

2.2.1 Cloud-in-Cell Method

One of the earliest fast techniques is known as *cloud-in-cell*, or *vortex-in-cell* when applied to vortex methods. We describe it here in terms of the vorticity equations.

Recall the vorticity-stream function formulation from Chapter 1:

$$\mathbf{u} = \nabla \times \Psi, \quad \omega = \nabla \times \mathbf{u},$$

and thus

$$\omega = -\Delta \Psi.$$

With the vortex-in-cell method, one begins by discretizing the vorticity into Lagrangian elements, as is standard in vortex methods. Then, however, the vorticity itself is averaged onto an Eulerian grid placed in the domain. The equation $\Delta \Psi = -\omega$ is solved on the grid, $\mathbf{u} = \nabla \times \Psi$ is calculated at the grid points, and then $\mathbf{u}$ is interpolated from the grid points onto the vortices. Boundary conditions for the Poisson equation are most likely given in the form of the normal velocity, hence the stream function at the boundary.

The accuracy of vortex methods has been shown to depend on the order of the core function used in the discretization of the vorticity, but with the vortex-in-cell technique there is no obvious way to incorporate the effects of the core function. The inability to separate the near-field and far-field effects is a serious limitation for the vortex-in-cell technique.

2.2.2 PPPM Method

PPPM (Particle-Particle Particle-Mesh) methods address the limitations of cloud-in-cell techniques by incorporating near-field effects directly. The basic idea behind PPPM methods is that there are two types of interactions between particles: near-

field and far-field forces. The far-field forces are smooth and can be accurately represented on a grid, and these comprise the Particle-Mesh part of the calculation, which is essentially cloud-in-cell. The near-field forces, as described in the original method, affect only nearby particles, and so can be calculated directly (Particle-Particle). The idea of having part of the velocity field represented on the grid and part calculated directly between particles is fundamental to the Method of Local Corrections. However, in the MLC it is not a separate force which is calculated between particles, rather it is the *correction* to the velocity field which is local, and is therefore computed directly. See [36] for a more complete description of Cloud-in-Cell and PPPM.

2.2.3 Method of Multipoles

More recently a fast summation method based on multipole expansion was developed by Greengard and Rokhlin [21, 32]. This method again uses the vorticity-stream function formulation, representing the velocity field in terms of Ψ originally, then taking derivatives to find the velocity components.

In the multipole method in two dimensions, use is made of a theorem in complex variable theory that allows us to expand the stream function due to a collection of point vortices with strengths ω_i at locations $z_i = x_i + iy_i$, as

$$\psi(z) = \mathrm{Re}(\Omega \log z + \sum_{k=1}^{\infty} \frac{a_k}{z^k})$$

if the vortices lie within a disk of radius r around the origin and $|z| > r$. This

expansion can be truncated at p terms with analytically known error. Here

$$\Omega = \sum_{i=1}^{m} \omega_i, \quad a_k = \sum_{i=1}^{m} -\frac{\omega_i z_i^k}{k}.$$

Consider the calculation of the potential at a set of n points $\mathbf{y}_j, |\mathbf{y}_j| > r$, due to a collection of m vortices at locations $\mathbf{x}_i, |\mathbf{x}_i| < r$. The cost of the direct calculation is $O(n \cdot m)$. Alternatively, the cost of evaluating the coefficients of a p-term multipole expansion of the potentials due to m vortices, and then evaluating the expansion at the n points $\mathbf{y}_j$, is $O(n \cdot p + m \cdot p)$. For p fixed by choice of accuracy, the cost is $O(n + m)$, a substantial reduction from $O(n \cdot m)$ for n and m sufficiently larger than p.

Note that the two collections of points must be "well-separated", so a sorting of the vortices is necessary, and the vortices must be grouped into collections which can each generate a single p-term multipole expansion. Theorems describing how to shift and combine expansions are used to reduce the number of computations. The interactions of vortices which are not "well-separated" from each other are calculated directly.

The original version of this method [32] was non-adaptive (i.e., constant "bin" size) and for two dimensions. Since the original work, the method has been made fully adaptive, so that the grouping of the vortices for expansions is dependent on the local vortex spacing, and has been extended to three dimensions [21]. In two dimensions the expansion is a complex power series; in three dimensions spherical harmonic expansions are used.

The ideas developed in the method of multipoles have been extended recently. Anderson [2] has developed an "implementation of the fast multipole method with-

out multipoles," based on the same principles as the fast adaptive multipole method, but using a representation by Poisson integrals rather than multipoles. The two common ingredients, as noted by Anderson, are that a large number of particles are combined into a single computational element and that particles are sorted and combined in an efficient manner.

In Anderson's variant in two dimensions, the computational element is a ring with p points rather than a p-term multipole expansion. The stream function induced by the vortices inside the ring is first calculated at these p points on the ring. Poisson integrals are then used to find the field outside the ring, using only the values at the points on the ring. Again, for m vortices inside the ring and n evaluation points outside the ring, the work is of $O(m \cdot p + n \cdot p)$, rather than $O(n \cdot m)$ for the direct method.

In three dimensions, harmonic functions have a more complicated form, containing Legendre polynomials in the cosine of the azimuthal angle. However, once Poisson's formula is written in terms of these functions, the rest of the method is very similar to the two dimensional case.

One last fast adaptive summation method is presented by van Dommelen and Rundensteiner [54]. This method is most similar to the adaptive multipole method; the differences are that Laurent series rather than Taylor series are used, and that the sorting and collecting of vortices, followed by their correct combination into multipole expansions, is done by a numbering of the groups based on a binary representation of the bin locations, gaining efficiency in the adaptivity. The authors note that their contribution is "primarily a programming technique which allows an easily addressable adaptive description of irregular distributions of points."

CHAPTER 3

Method of Local Corrections (MLC)

3.1 Description of the Algorithm

The Method of Local Corrections, developed by Anderson [1] in two dimensions, and extended to three dimensions by Buttke and Colella, is a method which reduces the cost of calculating the velocity at the vortices. We describe below how this velocity evaluation is done; the initial discretization and time-stepping are as described in Section 1.3.

The goal of the MLC is to replace the full $O(N^2)$ velocity calculation with a fast calculation whose cost varies as $O(N \log N)$ for large N. This is achieved by separating the velocity calculation into two parts: calculation of the far-field velocity on a grid and interpolation of a corrected velocity from the grid onto the vortices, and local interactions calculated between nearby vortices. The algorithm can be expressed as follows.

(I) Find at every grid point $\mathbf{i}$ a field $\mathbf{g}^{\Omega^0}$ which satisfies

$$\mathbf{g}_{\mathbf{i}}^{\Omega^0} \approx (\Delta^h \mathbf{u}^{e,h})_{\mathbf{i}}.$$

Here Δ^h is the discrete Laplacian operator with mesh spacing h; $\mathbf{u}_{\mathbf{i}}^{e,h}$ is defined as the exact velocity field induced by the vortices at the grid points, calculated without

core function effects,

$$\mathbf{u}_\mathbf{i}^{e,h} = \mathbf{u}^{exact}(\mathbf{i}h) = \sum_{n=1}^{N} \mathbf{K}((\mathbf{i}h) - \mathbf{x}_n)\,\omega_n.$$

(II) Solve

$$\Delta^h \tilde{\mathbf{u}}^h = \mathbf{g}^{\Omega^0}$$

for the velocity $\tilde{\mathbf{u}}$ on the grid with appropriate boundary conditions.

(III) For each vortex p, define

$$\mathbf{u}(\mathbf{x}_p) = I(\hat{\mathbf{u}}; \mathbf{x}_p) + \sum_{k\ near} \mathbf{K}_\delta(\mathbf{x}_p - \mathbf{x}_k)\omega_k,$$

where

$$\hat{\mathbf{u}}_\mathbf{i} = \tilde{\mathbf{u}}_\mathbf{i} - \sum_{k\ near} \mathbf{K}(\mathbf{x}_p - \mathbf{x}_k)\omega_k,$$

and I is the interpolation function.

Note that if $\mathbf{g}^{\Omega^0}$ were defined exactly as the discrete Laplacian of the velocity due to every vortex at every grid point, and the boundary conditions were specified exactly, then $\tilde{\mathbf{u}}_\mathbf{i} = \mathbf{u}_\mathbf{i}^{e,h}$ at every grid point. However, this is greater accuracy than is needed (since other errors in the method would swamp this error), and so in the method of local corrections we approximate the discrete Laplacian rather than computing it at every point. We define the contribution of each vortex to $\mathbf{g}^{\Omega^0}$ as exactly $\Delta^h \mathbf{u}^{e,h}$ on grid points *near* the vortex, but set $\mathbf{g}^{\Omega^0} = 0$ at grid points *far* from the vortex, thereby approximating the value of the discrete Laplacian with the value of the exact Laplacian, which is zero since the velocity due to a point vortex is harmonic. The error of the approximation is just the error of the discrete

Laplacian for a harmonic function, which is proportional to the higher derivatives of $\mathbf{u}^{exact}$. Near the vortex these derivatives are large, but it is only away from the vortex, where the derivatives are small and hence the error is small, that we make the approximation.

If in step (III) we were to interpolate the full velocity field from the grid onto the vortices, the dominant error in the method would come not from the error in approximating the discrete Laplacian of a harmonic field, but from the interpolation. The velocity field due to vortices near to the evaluation point is singular. Putting the velocity on a grid and then interpolating it back to the vortex locations is not sufficiently accurate, since the interpolation error is large where the derivatives of the velocity are large, i.e near the singularity. Also, in calculating the velocity on the grid we represent each vortex as a point vortex, and we now want to incorporate higher-order core function effects for a more accurate calculation. Therefore, the velocity which is interpolated from the grid onto the vortices is only the *corrected* velocity, i.e., that due to vortices sufficiently far away. So, in the interpolation step, we subtract the velocity due to *near* vortices from the velocity on the grid, and interpolate this corrected velocity onto the vortices. The corrected velocity only represents the influence of *far* vortices, and is a discrete harmonic function. The local interactions are then added directly, incorporating the core function effects.

In the above algorithm we need a mechanism for distinguishing between near and far vortices. Since the cost of calculating the distance between each pair of vortices is $O(N^2)$, all vortices are sorted into *bins* at the beginning of each time step; this sorting is based on the locations of the vortex centers. The centers of the bins are placed at the grid points, and each bin is defined as the box of width h

around its center. All sorting of near and far vortices is done using the bin indices.

Let $\Omega^0 = [0,1]^d$ be the physical domain of the problem in d space dimensions; place a uniform $[0,M]^d$ mesh with mesh spacing $h = \frac{1}{M}$ over the domain. Define $B^{\mathbf{i}}$ as the bin centered on the point $\mathbf{i}h$, and define the $|\cdot|_B$ norm such that $|\mathbf{i}-\mathbf{m}|_B$ is the minimum distance (in units of the mesh spacing) between any point in $B^{\mathbf{i}}$ and any point in $B^{\mathbf{m}}$. Around every grid point now define $R^{\mathbf{i}}$ and $R_0^{\mathbf{i}}$ as:

$$R^{\mathbf{i}} = \bigcup_{\mathbf{m}:|\mathbf{m}-\mathbf{i}|_B \le (D+1)} B^{\mathbf{m}},$$

$$R_0^{\mathbf{i}} = \bigcup_{\mathbf{m}:|\mathbf{m}-\mathbf{i}|_B \le D} B^{\mathbf{m}},$$

where D is called the spreading distance, and typically is in the range $1 \le D \le 4$. The right-hand side for the Poisson equation, $\mathbf{g}^{\Omega^0}$, is found as follows. See Figure 3.1 for a picture of the two dimensional case.

(1) For each $\mathbf{i}$ in the interior of Ω^0

(a) compute by direct interaction the exact velocity at every grid point $\mathbf{m}$ in $R^{\mathbf{i}}$ due to every vortex n in $B^{\mathbf{i}}$ such that $|\mathbf{x}_n - \mathbf{m}h| > \varepsilon$, ε slightly larger than machine precision, with no core function effects:

$$\mathbf{u}_{\mathbf{m}}^{\varepsilon,h} = \sum_{n:\mathbf{x}_n \in B^{\mathbf{i}}} \mathbf{K}(\mathbf{m}h - \mathbf{x}_n)\,\omega_n.$$

(b) Calculate the Laplacian of this velocity field at every point in $R_0^{\mathbf{i}}$. The stencils for the discrete Laplacian are presented in Chapter 5. Define

$$\mathbf{g}^{\mathbf{i}} = \begin{cases} \Delta^h \mathbf{u}^{\varepsilon,h} & \text{inside } R_0^{\mathbf{i}} \\ \mathbf{0} & \text{in interior}(\Omega^0) - R_0^{\mathbf{i}}. \end{cases}$$

Note that $\mathbf{g}^{\mathbf{i}}$ is defined at every point in the interior of Ω^0, but carries information only about the vortices in $B^{\mathbf{i}}$.

(2) Superimpose these fields $\mathbf{g}^{\mathbf{i}}$ to form

$$\mathbf{g}^{\Omega^0} = \sum_{\mathbf{i} \in \text{ interior}(\Omega^0)} \mathbf{g}^{\mathbf{i}}.$$

(3) The velocity is then found by

$$\tilde{\mathbf{u}} = (\Delta^h)^{-1} \mathbf{g}^{\Omega^0}.$$

The work to represent the velocity field on the entire grid due to all the vortices is broken down as follows: for each vortex, evaluate the exact velocity in a subset of the grid; for each bin of vortices, evaluate the Laplacian in an even smaller subset of the grid; for the whole domain (i.e., only once) solve the equation $\Delta^h \tilde{\mathbf{u}} = \mathbf{g}^{\Omega^0}$. The details of how we solve the Poisson equation are described in Chapter 5.

For the local corrections part of the algorithm, around each grid point $\mathbf{i}$ define

$$S^{\mathbf{i}} = \bigcup_{\mathbf{m}:|\mathbf{m}-\mathbf{i}|_B \leq C} B^{\mathbf{m}},$$

where C is called the correction radius, and typically falls in the range $1 \leq C \leq 4$. If the correction radius C and spreading radius D are the same then $S^{\mathbf{i}} = R^{\mathbf{i}}$. A vortex p in bin $B^{\mathbf{i}}$ is defined as *near* to a vortex k in bin $B^{\mathbf{k}}$ if $B^{\mathbf{i}}$ is in $S^{\mathbf{k}}$. Note that p is near to k implies k is near to p (this is not necessarily true when AMR is added to the MLC). The local corrections step is performed one bin at a time; see Figure 3.2 for a picture of the two dimensional case. For each $\mathbf{i}$ such that $B^{\mathbf{i}}$ contains vortices:

(4) Define the interpolation stencil $\{\mathbf{X}_\alpha\}, \alpha = 1 \ldots, S$ where S is the number of interpolation points. Compute by direct interaction the exact velocity at each point $\mathbf{X}_\alpha$ due to every vortex n in $S_\mathbf{i}$ such that $|\mathbf{x}_n - \mathbf{X}_\alpha| > \varepsilon$, without core function effects, and subtract this field from the existing velocity at these grid points:

$$\hat{\mathbf{u}}(\mathbf{X}_\alpha) := \tilde{\mathbf{u}}(\mathbf{X}_\alpha) - \sum_{n:\mathbf{x}_n \in S_\mathbf{i}} \mathbf{K}(\mathbf{X}_\alpha - \mathbf{x}_n)\, \omega_n.$$

Note that if $C = D$ then this corrected field $\hat{\mathbf{u}}$ satisfies $\Delta^h \hat{\mathbf{u}} = 0$, i.e., the field to be interpolated is discretely harmonic.

(5) Interpolate this corrected field $\hat{\mathbf{u}}$ from the interpolation points $\mathbf{X}_\alpha$ onto each vortex p in $B_\mathbf{i}$:

$$\mathbf{u}(\mathbf{x}_p) := I(\hat{\mathbf{u}}(\mathbf{X}_1), ..., \hat{\mathbf{u}}(\mathbf{X}_S); \mathbf{x}_p)$$

After this interpolation, the velocity of every vortex in $B_\mathbf{i}$ is due only to the vortices outside $S_\mathbf{i}$.

(6) Now add the velocity due to every vortex n in $S_\mathbf{i}$ to the existing velocity of every vortex p in $B_\mathbf{i}$ using $\mathbf{K}_\delta$ rather than $\mathbf{K}$:

$$\mathbf{u}(\mathbf{x}_p) := \mathbf{u}(\mathbf{x}_p) + \sum_{n:\mathbf{x}_n \in S_\mathbf{i}} \mathbf{K}_\delta(\mathbf{x}_p - \mathbf{x}_n)\, \omega_n.$$

In two dimensions velocities are evaluated at the locations $\mathbf{x}_i$ of the vortices; in three dimensions recall that the vorticity is centered at the segment midpoints, but the velocity is evaluated at the segment endpoints. Thus in three dimensions the final

two steps would be written:

$$\mathbf{u}(\mathbf{x}_p^{T/B}) := I(\hat{\mathbf{u}}(\mathbf{X}_1), ..., \hat{\mathbf{u}}(\mathbf{X}_S); \mathbf{x}_p^{T/B}) + \sum_{n:\mathbf{x}_n \in S_i} \mathbf{K}_\delta(\mathbf{x}_p^{T/B} - \mathbf{x}_n^C)\, \omega_n,$$

where

$$\hat{\mathbf{u}}(\mathbf{X}_\alpha) := \tilde{\mathbf{u}}(\mathbf{X}_\alpha) - \sum_{n:\mathbf{x}_n \in S_i} \mathbf{K}(\mathbf{X}_\alpha - \mathbf{x}_n^C)\, \omega_n.$$

Although conceptually the algorithm is similar in two and three dimensions, numerically it is more complicated in three dimensions. Rather than a 9-point Laplacian we use a 27-point Laplacian; rather than a 5-point interpolation scheme in two dimensions we require a 19-point interpolation stencil in three dimensions. The stencils of the discrete Laplacian and the interpolation functions are presented in Chapter 5.

3.2 Parameter Analysis

There are a number of parameters which affect the accuracy and cost of the method of local corrections. We distinguish here between the error inherent in using a vortex method to approximate the solution to the Euler equations and the error which results from approximating the standard vortex method with the method of local corrections.

The first type of error, that of the vortex method itself, depends on three parameters: 1) h_v, the intervortex spacing (in two dimensions this is only one quantity; in three dimensions, one may differentiate between the intervortex spacing in the plane normal to the direction of vorticity, and the length of the vortex segments), 2) f_δ,

the core function, and 3) Δt, the time step. The standard approach to choosing the parameters tends to be that one first chooses as many vortex elements as computationally possible. One then selects a core function to give the desired theoretical order of accuracy.

Since there is no grid in the standard vortex method, there is no CFL (Courant-Friedrichs-Levy) condition, hence no limit on the time step due to stability considerations. There is certainly a limit due to accuracy considerations; in practice the procedure is to justify the choice of a time step by repeating the calculation at a smaller time step and showing that the results do not change. There are other ways to choose, such as requiring that a vortex not move farther than some fraction of the intervortex spacing (in two dimensions), or that a vortex not rotate through an arc more than some fraction of its length (in three dimensions).

The second type of error is that of approximating the standard vortex method with the method of local corrections. This error can be separated into two parts: (a) the error in representing the velocity on the grid, and (b) the error in interpolating the corrected velocity from the grid onto the vortices. The first error results from approximating the value of the discrete Laplacian of the velocity due to a vortex element by zero away from that element; this error depends on the spreading distance D. As D increases for constant grid spacing h, we make this approximation on fewer points farther away from the vortex element, and thus in the limit as D for each vortex covers the entire domain this error goes to zero. The second error results from interpolation. For a constant grid spacing, h, as we increase the correction radius C we are interpolating not only a smaller fraction of the total velocity field (since the velocity we are interpolating is due to fewer vortices), but also a smoother

function, since the corrected velocity is due only to vortices outside the correction radius. Thus as C increases the interpolation error goes to zero, and in the limit C = M, for a $[0, M]^d$ grid, the method of local corrections effectively reduces to the standard vortex method.

We present several parameter studies in two dimensions. Consider first the error in finding the velocity field on the grid. In Tables 3.1 and 3.2 we see the error in the x-component of velocity on the grid for varying spreading distance D and mesh spacing h. Table 3.1 shows the relative error in the x-component of velocity at the point $(.75, .75)$ (in a domain $[0, 1]^2$) due to a single vortex of unit strength at $(.505, .505)$. Table 3.2 shows the discrete relative L_2 norm of the error in the velocity.

There are several conclusions we can draw from these tables. First, we see that $D = 1$ does not give sufficient accuracy, but D need be no larger than 2 or 3 for good accuracy. Recall that we need not seek accuracy in the MLC greater than that of the vortex method itself.

Second, note that the error in both tables varies inversely with the mesh spacing, so that the greatest accuracy in velocity is found on the coarsest grid. In the cases of $h = \frac{1}{8}$, $D = 3$ and $D = 4$, we see that the error is especially small; this we would expect since the exact velocity is put on all points of the grid, the discrete Laplacian is applied and then inverted, and the exact velocity field is regained to the specified precision of the solver, which is 10^{-10} in this calculation.

In Tables 3.3 and 3.4 are the relative point and L_2 norm of the error in velocity due to a smoother distribution of vorticity, a circular patch of vorticity of radius .2

	$D = 1$	$D = 2$	D=3	D=4
$h = \frac{1}{8}$	3.02×10^{-3}	4.51×10^{-5}	9.88×10^{-10}	9.88×10^{-10}
$h = \frac{1}{16}$	3.54×10^{-3}	6.01×10^{-5}	4.05×10^{-6}	6.30×10^{-6}
$h = \frac{1}{32}$	2.47×10^{-2}	1.02×10^{-4}	5.82×10^{-6}	2.39×10^{-6}
$h = \frac{1}{64}$	2.93×10^{-1}	6.51×10^{-4}	8.20×10^{-6}	3.70×10^{-6}

Table 3.1: Relative point error in u at $(.75, .75)$ due to a single vortex of unit strength at $(.505, .505)$ as a function of spreading distance D and mesh spacing h.

	$D = 1$	$D = 2$	D=3	D=4
$h = \frac{1}{8}$	9.94×10^{-4}	2.43×10^{-5}	1.16×10^{-10}	1.16×10^{-10}
$h = \frac{1}{16}$	2.41×10^{-3}	5.01×10^{-5}	5.31×10^{-6}	1.09×10^{-6}
$h = \frac{1}{32}$	1.40×10^{-2}	9.34×10^{-5}	8.80×10^{-6}	1.94×10^{-6}
$h = \frac{1}{64}$	2.08×10^{-1}	4.41×10^{-4}	1.30×10^{-6}	2.89×10^{-6}

Table 3.2: Relative L_2 norm of error in u due to a single vortex of unit strength at $(.505, .505)$ as a function of spreading distance D and mesh spacing h.

centered at (.5, .5). The original vorticity was given by

$$\omega(x,y,0) = \begin{cases} 4\pi(1-4R^2)^7 & \text{for } R \leq 0.5 \\ 0 & \text{for } R \geq 0.5 \end{cases},$$

where $R^2 = (x-.5)^2 + (y-.5)^2$. We see here a slightly weaker dependence of the error on h than for the single vortex. We do not include data for $D = 3$ or $D = 4$ when $h = \frac{1}{8}$ because the grid is too small to accommodate that large a spreading distance around a patch of radius .2.

Note also from Tables 3.1 through 3.4 that for $D \geq 2$ the relative error is in places an order of magnitude or more smaller for the smooth distribution than for the point vortex.

Consider now the error due to interpolating the velocity field from the grid onto the vortex locations. In the calculations for Table 3.5 we put the exact velocity onto the grid points, and then measure the L_2 norm of the error in velocity at the vortices, thus capturing the error due entirely to interpolation. The parameters for the results in Table 3.5 are the same as for Tables 3.3 and 3.4, except that the velocity is exact now on the grid, so D is effectively infinite.

In comparing Tables 3.4 and 3.5, both for smooth distributions of vorticity, we draw two conclusions: (1) the error due to approximating the Laplacian of the velocity on the grid is of the same order of magnitude as the error due to interpolation; (2) the error depends more strongly on the correction radius C and the spreading distance D than it does on the mesh spacing h.

In Table 3.6 we present results from [1], which show the relative error in vortex locations after a finite time two-dimensional calculation with the method of local corrections using different core functions, due to Chorin [22] and to Beale and Ma-

	$D = 1$	$D = 2$	D=3	D=4
$h = \frac{1}{8}$	2.53×10^{-3}	5.59×10^{-5}		
$h = \frac{1}{16}$	2.57×10^{-4}	4.82×10^{-5}	7.58×10^{-6}	1.05×10^{-6}
$h = \frac{1}{32}$	7.47×10^{-4}	3.24×10^{-5}	4.68×10^{-6}	1.53×10^{-6}
$h = \frac{1}{64}$	1.12×10^{-3}	3.22×10^{-5}	4.21×10^{-6}	1.57×10^{-6}

Table 3.3: Relative point error in u at $(.75, .75)$ due to a patch of vorticity centered at $(.5, .5)$, radius .2, as a function of spreading distance D and mesh spacing h. The intervortex spacing is .01, the core radius is $(.01)^{.75}$, and there are 1257 vortices.

	$D = 1$	$D = 2$	D=3	D=4
$h = \frac{1}{8}$	8.76×10^{-4}	2.47×10^{-5}		
$h = \frac{1}{16}$	2.67×10^{-4}	2.10×10^{-5}	2.36×10^{-6}	1.12×10^{-6}
$h = \frac{1}{32}$	7.04×10^{-4}	3.08×10^{-5}	3.59×10^{-6}	1.55×10^{-6}
$h = \frac{1}{64}$	1.39×10^{-3}	3.57×10^{-5}	4.40×10^{-6}	1.71×10^{-6}

Table 3.4: Relative L_2 norm of error in u due to a patch of vorticity centered at $(.5, .5)$, radius .2, as a function of spreading distance D and mesh spacing h. The intervortex spacing is .01, the core radius is $(.01)^{.75}$, and there are 1257 vortices.

	$C = 1$	$C = 2$	C=3
$h = \frac{1}{8}$	9.14×10^{-4}	4.45×10^{-5}	
$h = \frac{1}{16}$	2.68×10^{-4}	5.50×10^{-5}	1.44×10^{-5}
$h = \frac{1}{32}$	6.79×10^{-5}	1.19×10^{-5}	5.04×10^{-6}
$h = \frac{1}{64}$		3.99×10^{-6}	1.43×10^{-6}

Table 3.5: Relative L_2 norm of error in velocity at the vortex locations. Parameters are as in Table 3.3.

	DIRECT	$C = D = 1$	$C = D = 2$	$C = D = 3$
Chorin core	8.36×10^{-3}	8.59×10^{-3}	8.35×10^{-3}	8.36×10^{-3}
Beale-Majda core	4.15×10^{-3}	4.32×10^{-3}	4.14×10^{-3}	4.14×10^{-3}

Table 3.6: L_2 norm of error in vortex locations at time $t = 1$ using two different core functions. Anderson [1].

jda [9]. The calculations are done first using the direct method, then with the MLC, varying $C = D$. The core functions are:

Chorin:

$$f_\delta(r) = \begin{cases} 1/(2\pi r\delta) & \text{for } r < \delta \\ 0 & \text{for } r \geq \delta \end{cases};$$

Beale-Majda:

$$f_\delta(r) = \frac{e^{-(r/\delta)^2}}{\pi\delta^2}(2 - \frac{r^2}{\delta^2}),$$

where r is the distance from the vortex. Here the original vorticity was as for Tables 3.3 and 3.4, $\delta = h^{.95}$, the time-stepping procedure was 4^{th} order Runge-Kutta, and the time $t = 1$ corresponded to a maximal rotation of one revolution.

We present this here to verify that the higher-order accuracy of the second core function is in fact preserved by the method of local corrections for a finite time calculation. In addition we see from Table 3.6 that the error of the solution as calculated using the MLC does not vary with $C = D$ here, and is essentially the error of the solution as calculated using the direct method. We conclude from this that the error inherent in the vortex method in fact swamps the error in approximating the standard vortex method by the MLC for finite time calculations.

Baden [6] performed parameter studies of the method of local corrections in two dimensions, varying the number of vortices, the mesh spacing, and the time

N	h	$\Delta t = 0.1$	$\Delta t = 0.05$	$\Delta t = 0.025$	$\Delta t = 0.0125$	$\Delta t = 0.00625$
1005	1/30	9.09×10^{-3}	4.57×10^{-3}	3.50×10^{-3}	3.24×10^{-3}	3.18×10^{-3}
	1/60	9.08×10^{-3}	4.56×10^{-3}	3.49×10^{-3}	3.23×10^{-3}	3.17×10^{-3}
4020	1/30	7.66×10^{-3}	3.65×10^{-3}	1.48×10^{-3}	1.22×10^{-3}	-
	1/60	7.66×10^{-3}	3.65×10^{-3}	1.48×10^{-3}	1.22×10^{-3}	-
	direct	7.66×10^{-3}	3.65×10^{-3}	1.47×10^{-3}	1.21×10^{-3}	-
16043	1/30	7.17×10^{-3}	1.97×10^{-3}	7.48×10^{-4}	4.83×10^{-4}	4.22×10^{-4}
	1/60	7.17×10^{-3}	1.97×10^{-3}	7.50×10^{-4}	4.85×10^{-4}	4.24×10^{-4}
	1/120	7.17×10^{-3}	1.97×10^{-3}	7.51×10^{-4}	4.86×10^{-4}	4.25×10^{-4}

Table 3.7: L_2 norm of error in vortex locations after finite time as a function of number of vortices N, mesh spacing h, and time step Δt .Baden [6].

step. The core radius was $\delta = h^{.75}$, and the correction and spreading distances were $C = D = 2$. The initial distribution of vorticity was the same as used by Anderson in the above calculation, and Chorin's core function was used. The results are displayed in Table 3.7. Baden's results show:

(1) The time step Δt should vary proportionally with the intervortex spacing. We define the optimal time step as the largest time step for which reducing the time step does not significantly lower the accuracy. The choice of appropriate time step can be inferred from the table by moving across a row and noting when decreasing the time step does not appreciably decrease the error.

(2) Accuracy improves with decreasing intervortex spacing. Note that the improvement here is not large, because the original vorticity distribution is so smooth.

(3) Given a fixed number of vortices, the error is independent of mesh spacing.

Again we see by comparing the direct and MLC calculations with $N = 4020$ that the error inherent in the vortex method swamps the error due to the MLC.

3.3 Comparison of Fast Vortex Methods

In this and the previous chapter we have discussed several different types of fast vortex methods: cloud-in-cell, PPPM, multipole method, and the method of local corrections (MLC). These methods have several fundamental similarities and differences.

The first distinction we make is between grid-based and non-grid-based methods. Cloud-in-cell, PPPM, and MLC use a grid to represent the velocity field before it is interpolated onto the vortices. The different multipole methods have no grid representation of data; they only use a grid for sorting purposes.

The essential similarity between all the above methods is their reliance on the regularity of harmonic functions. This regularity ensures that the solution to Poisson's equation is very smooth away from the support of the right-hand side, and therefore one can approximate the far-field effects with a small number of computational degrees of freedom with little loss of accuracy. The solution to Poisson's equation with a rapidly varying right-hand side may be rapidly varying near the support of the right-hand side, but far away the solution is smooth regardless of the singularities in the right-hand side or its derivatives.

All of these methods rely on this regularity to represent the field due to a collection of grouped "charges" by a smaller number of computational degrees of freedom with good accuracy. In cloud-in-cell, PPPM, and MLC, the representation of the field, whether stream–function Ψ as in cloud-in-cell, or $\mathbf{u}$ as in MLC, is on a grid. In the multipole algorithm, it is the coefficients in the multipole expansions that

determine the field; in Anderson's Poisson integral method, it is the values of the stream function at points on a circular ring.

It is interesting to consider a comparison with Fourier transforms and multigrid as well. In using Fourier transforms, one is also representing physical data in another space, here the spectral representation. If the data is sufficiently smooth, then one need only keep the lower-frequency coefficients, and one has the same conceptual representation as with the multipole expansion. To find the field due to a collection of charges, one could find the first p terms of the Fourier series of the field due to these charges, then evaluate that series at points in physical space. Since the interaction of close vortices would need to be represented with higher frequencies, one would need a way of correctly separating the near-field and far-field effects, just as with all the fast methods. See Strain [52, 53] for a discussion of how to implement a method based on Ewald summation and fast transforms of Gaussians and Fourier series.

Multigrid also uses the concept that a smooth field can be represented accurately in a coarser way than a rapidly varying field. The coarse grid transmits information across the grid much more rapidly than a fine grid, since in this iterative method information is only transmitted one grid point per relaxation. Multigrid can be used to solve a Poisson equation with a rapidly varying right-hand side, but the work is apportioned to make the solution as efficient as possible—relaxation on the coarser grids captures the smoother components of the field, and the finer detail of the solution is found on the fine grid.

The programming structure of multigrid and of the adaptive multipole-type expansion methods is virtually the same [2, 37]. In the adaptive multipole-type meth-

ods one sweeps from finer to coarser levels, creating nested boxes to use as computational elements. In multigrid, one descends from fine grid to coarse grid and relaxes on the grid at every step. Forming an element at one level by combining elements from a finer level is equivalent to the restriction operator in multigrid. Creating an inner approximation from potential values induced by an inner ring approximation associated with a parent box in the multipole method is similar to the prolongation operator in multigrid. There are two sweeps necessary in the multipole-type method; first the creation of the outer ring approximations, then the evaluation of the inner ring approximations; these we liken to the two sides of the multigrid V-cycle, first going from fine to coarse grid, then returning from coarse to fine grid.

We will also see in later chapters that the MLC with adaptive mesh refinement (AMR) fits very easily into the multigrid structure. The original MLC algorithm represents all the information on a single grid, but with AMR we introduce multiple levels, and keep the least smooth information on the finest grids, the smoother components on the coarser grids.

The differences between multipole-type methods were mentioned in Chapter 2; here we contrast a grid-based and a non-grid based method, i.e. the method of local corrections and the multipole method. Note that the appropriate comparisons are between the original multipole method and the original MLC, or between the adaptive multipole method and MLC with AMR. Both of the latter have the same degree of adaptivity, in that the data structure used to represent the field values varies in fineness with the vortex spacing. In both cases the goal is to represent as much of the field as possible in the "fast" way (by multipole expansion, or on the MLC grid) within certain limits of accuracy, and then to capture the remaining

fine-detail structure of the field by direct interaction. The direct interactions are done identically, the difference lies in how the smooth far field is represented.

Boundary conditions can be handled much more easily in a grid-based method. We will discuss in Chapter 5 how to calculate the different types of boundary conditions; here we want mainly to point out that with a grid representation it is fairly straightforward to satisfy various boundary conditions (infinite domain, periodic, or no-flow wall). To impose no-flow wall boundary conditions for a non-grid based method requires conformal mapping, or solving Laplace's equation on a grid and interpolating the solution from the grid to the vortices. The elliptic solver and the interpolation are already incorporated in the MLC, so no-flow wall conditions are in fact trivial to impose. The same applies to periodic boundary conditions; on a grid it is simple to "wrap" the values around the periodic direction, but without a grid one must create image vortices in the periodic direction, requiring additional computational effort.

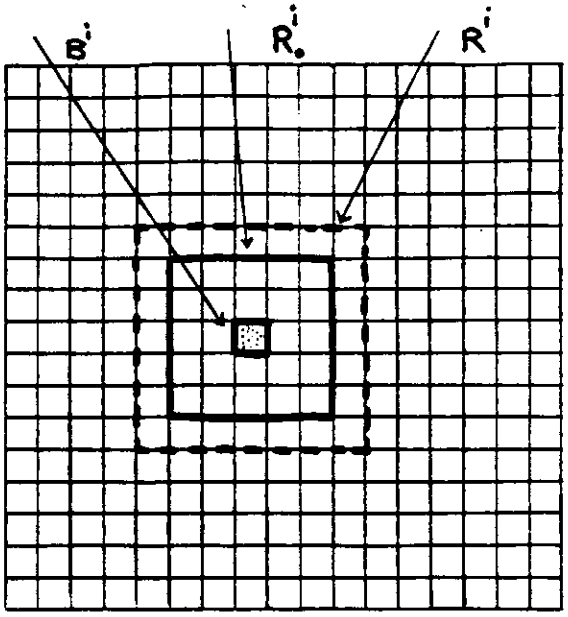


Figure 3.1: Grid showing $B^{\mathbf{i}}, R^{\mathbf{i}}, R_0^{\mathbf{i}}$ for creation of $\mathbf{g}^{\mathbf{i}}$ in two dimensions. Small dots are vortices.

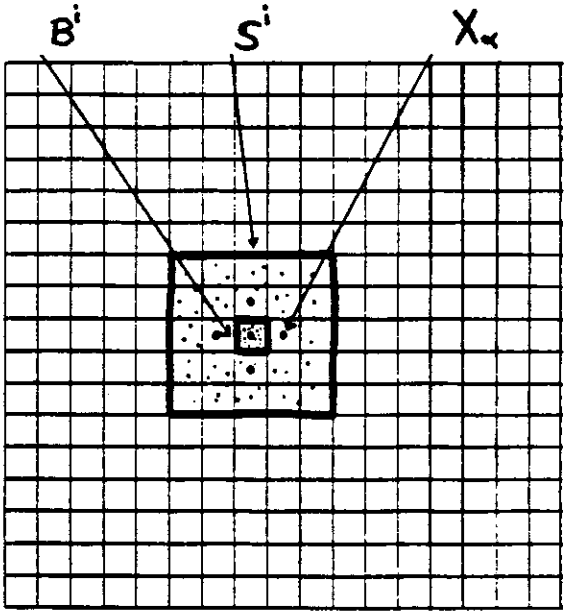


Figure 3.2: Grid showing $B^{\mathbf{i}}, S^{\mathbf{i}}$ for local corrections in two dimensions. Small dots are vortices, large dots are the grid points in the interpolation stencil.

CHAPTER 4

MLC with Adaptive Mesh Refinement

4.1 Adaptive Mesh Refinement

Adaptive mesh refinement (AMR) is a technique for selectively refining the mesh in a grid-based calculation. Using AMR, a base grid is maintained for all time and new finer grids are added adaptively in time, overlaying regions of the base grid. The motivation for adaptive mesh refinement is that for most problems the level of refinement needed to capture important features of the solution can vary widely throughout the problem domain. Certain areas in the domain where there is little activity need only a coarse grid; other areas may need a very fine grid. In finite difference calculations the penalty for using a single coarse grid is loss of accuracy; the penalty for using a single fine grid is time—the cost of a single time step of a three-dimensional finite-difference calculation will grow by a factor of eight for every factor of two refinement.

AMR has been implemented in finite-difference codes [12], but never in the method of local corrections. The automatic, adaptive mesh refinement strategy that we use in this work was developed by Berger and Oliger [11] for hyperbolic equations on rectangular grids. Earlier work was done by Bolstad [17] in one dimension, and Gropp [33] for scalar problems in two dimensions. The algorithm has been extended to three dimensions [15] and mapped grids [14, 16]. Here we consider

only rectangular grids in two and three dimensions. For our algorithm, we use a modified version of the code discussed in [12, 15], a robust and efficient version of the algorithm for time-dependent shock hydrodynamics. There has also been considerable effort to use adaptive mesh ideas for solving elliptic equations using finite difference or finite element approaches; for a review, see [43].

Adaptive mesh refinement is simple in concept, but nontrivial to implement. In the simplest form of AMR, the *base* grid is defined as the original uniform rectangular mesh with mesh spacing h, and one first determines in what regions of the base grid a finer mesh is needed. A grid with mesh spacing $\frac{h}{m}$, m typically 2, is created in these regions. One then determines what regions of the refined grid need refining, and a new finer grid is created if needed. This process of refinement can be repeated many times in an iterative fashion. The questions whether to refine by more than a factor of two at each level, how to properly nest the grids, and whether grids must be rectangular, depend on the application.

Note that AMR does not adapt the grid by moving grid points into one region, thereby leaving a coarser mesh elsewhere. With AMR, a refined grid is placed on top of a region of the coarse grid, but the resolution in all regions never becomes coarser than that of the original base grid.

The refinement criterion in finite difference calculations is an error measure based on Richardson extrapolation. For these calculations one computes the solution on a coarse grid and on a fine grid. If the difference between the solutions is above a certain tolerance, ϵ_{max}, the region is refined, and the solution is computed on a new finer level. If once again the difference between the fine and finer solutions is greater than ϵ_{max}, the region is again refined. Regions are refined until the difference

between the solution at that level and at the next level of refinement is below the tolerance. For AMR with the MLC, the refinement criterion is based on the number of vortices per bin. We discuss this further in Section 4.4.

For AMR as used with the MLC, grid regions at each level must be rectangular. We allow multiple rectangles per level, and in general, we allow these rectangles to overlap. We define level ℓ points or grids as those with mesh spacing $h_\ell = h_0/2^\ell$. The set of all rectangles generated by the grid creation procedure at level ℓ defines G^ℓ, and we call

$$G^{0:\ell} = \bigcup_{i=0}^{\ell} G^i$$

the *composite grid* at level ℓ. See Figure 4.1 for a sample composite grid at level 2. Certain computations are performed on each rectangle separately; the Poisson equation is solved on the composite grid for each level above and at the base grid level. Note that "above" refers to a finer level, "below" denotes a coarser level.

Initially, the regions of a level ℓ mesh needing refinement are represented by a set of flagged points at level $\ell + 1$; this set contains the points which must be included in the grids generated at level $\ell + 1$. The creation of the individual rectangles given a list of flagged points is achieved using a modified bisection algorithm presented in [13]. This algorithm aims to create as few rectangles as possible at each level, to reduce the computational overhead and allow the longest possible vector lengths for the Poisson solver, but each with at least a minimum efficiency, measured as number of flagged points divided by number of total points within the rectangle. The required efficiency is an input parameter, and in our cases has been in the range 50% to 80%.

The process of grid creation is described below in more detail. Assume that we

have a list of flagged points for each level ℓ up to a maximum level of refinement. Starting with the finest level,

(1) *Ensure proper nesting of the grids* by flagging at the present level all points lying under a grid at a level finer than the present level. (This does not apply at the finest level.)

(2) *Add buffer points* around the currently flagged points at this level. The purpose of the buffer zone in finite difference calculations of hyperbolic problems is to ensure that discontinuities or other regions of high error do not propagate out from a fine grid into coarser regions before the next regridding. In the MLC with AMR, buffer points are necessary so that the full representation of the right-hand side due to a vortex in a flagged bin can fit on the interior of the grid at that level, and that local corrections can be done properly. This buffering step is actually done in a slightly different manner for AMR with MLC than for AMR in a finite difference calculation; this is discussed further in the next section.

(3) *Define a cluster* as the minimum rectangular box containing all the flagged points (including the buffer points) at this level.

(4) *Check the efficiency of each cluster at this level.* If it is greater than the minimum efficiency, check the next cluster. If this is the only cluster and it satisfies the efficiency requirement, or if all of the clusters have at least minimum efficiency, go to step (6).

(5) *Create new clusters.* For each cluster which does not satisfy the efficiency requirement, calculate the number of points in a cross-section of the cluster for each coordinate direction. Then calculate an approximation of the second derivative of this data, and locate the lowest value of the derivative, indicating a local minimum in width of the rectangle. Repeat in each coordinate direction, and choose the lowest value of this derivative over all directions. Bisect the rectangle along the cross-section with the lowest value of the derivative. Two new clusters are now created by taking the minimum rectangle containing all flagged points on each side of the bisection. Return to step (4) to check these new clusters.

(6) *Store each cluster as a part of the level ℓ grid* in the data structure. Note that these rectangles do not overlap but may share common boundaries. If currently at level $\ell > 1$, return to step (1) for the points at the next coarsest level.

The bisection procedure in step (5) above is constrained not to create new clusters with less than a minimum prescribed thickness. Also, if the procedure cannot decide where to bisect, the cluster is bisected at the midpoint of its longest direction.

The data structures for AMR adapt in time with the evolution of the grids. They allocate space for the solution vector in each rectangle independently, using only the necessary amount of space. A pointer vector is also maintained, indexed by the number of the rectangle it describes, containing the grid indices defining the rectangle, and the pointers to the locations in memory where the solution and other necessary values are stored.

4.2 MLC with AMR

The accuracy of the method of local corrections has very weak dependence on the mesh spacing of the grid used to calculate the far-field velocity, as long as the mesh spacing is sufficiently larger than the interparticle spacing. Thus, we can choose the mesh spacing within limits by timing considerations; just how to do this is discussed further in Section 4.4. The goal of AMR with MLC is to reduce the cost of the local corrections by creating smaller bins in regions where the vortices are concentrated while increasing as little as possible the cost of solving of the Poisson equation.

Adaptive mesh refinement introduces a hierarchy of grids, $G^\ell, \ell = 0, ..., \ell_{max}$, with mesh spacing $h^\ell = h^0/2^\ell$. G^0 is the base grid and $G^{\ell_{max}}$ is the finest grid. Only G^0 covers all of Ω^0, the full computational domain.

Initially, the regions at level ℓ to be refined are defined as the bins at level ℓ (those centered on level ℓ grid points and with width h_ℓ) containing more than N_{max} vortices; if a bin on a level ℓ grid contains more than N_{max} vortices, then all level $\ell + 1$ grid points which lie within or on the boundary of that bin are flagged. In two dimensions, a single level ℓ bin needing refinement results in 9 flagged points at level $\ell + 1$; in three dimensions a level ℓ bin needing refinement requires 27 level $\ell + 1$ points to be flagged. This ensures that the bin at level ℓ will be completely covered by the bins that are created at level $\ell + 1$. since the boundaries of bins at level ℓ and at level $\ell + 1$ do not coincide. How to determine N_{max} is discussed in Section 4.4.

Once the grids G^ℓ are created for every level ℓ, $0 < \ell \leq \ell_{max}$, using the procedure described in Section 4.1, we sort all vortices into sets V^ℓ. A vortex p is a level ℓ

vortex, and hence a member of V^ℓ, if p is contained in a level ℓ bin which is in the interior of G^ℓ and centered on a grid point more than $b+1$ level ℓ grid points from all boundaries of G^ℓ, where $b = \max(C, D)$. Note that not all level ℓ bins contain level ℓ vortices. The velocity due to a level ℓ vortex is represented on $G^{0:\ell}$; this containment criterion ensures that the method of local corrections can properly represent the right-hand side of the Poisson equation and perform the local corrections correctly for the level ℓ vortex on the level ℓ grid.

The method of local corrections is performed once on $G^{0:\ell}$ for each level ℓ. For the level ℓ calculation, the right-hand side for the Poisson equation is defined using vortices in V^ℓ only, and boundary conditions due to vortices in V^ℓ only are defined on the base level grid. The Poisson equation is then solved on $G^{0:\ell}$, generating a velocity field $\tilde{\mathbf{u}}^{\ell:m}$ on each G^m, $0 \le m \le \ell$, of the composite grid $G^{0:\ell}$. Note that because the Poisson equation is solved separately for each group of vortices V^ℓ, the right-hand side is only nonzero on G^ℓ; elsewhere on $G^{0:\ell}$ it is set to zero. The velocity is then interpolated onto all vortices, with local corrections required only on G^ℓ. This procedure is repeated for all levels, ℓ, $0 \le \ell \le \ell_{max}$, adding the contributions from each set of vortices V^ℓ until the velocity of every vortex p due to vortices at all levels has been calculated. We can express the full algorithm as

$$\mathbf{u}(\mathbf{x}_p) = \sum_{\ell=0}^{\ell_{max}} \mathbf{u}^\ell(\mathbf{x}_p),$$

where

$$\mathbf{u}^\ell(\mathbf{x}) = \begin{cases} I^m(\tilde{\mathbf{u}}^{\ell:m}; \mathbf{x}) & \text{if } m < \ell \\ I^\ell(\hat{\mathbf{u}}^{\ell:\ell}; \mathbf{x}) + \sum_{k \in V^\ell, k\ near} \mathbf{K}_\delta(\mathbf{x} - \mathbf{x}_k)\omega_k & \text{if } m = \ell, \end{cases}$$

and

$$\hat{\mathbf{u}}^{\ell:\ell} = \tilde{\mathbf{u}}^{\ell:\ell} - \sum_{k \in V^\ell, k \; near} K(\mathbf{x} - \mathbf{x}_k)\omega_k.$$

The level m at which the interpolation is done is the finest level such that $m \leq \ell$ and $\mathbf{x}_p$ is in the interior of G^m. The interpolation stencil I^m is composed of points in G^m with spacing h_m. Note that p need not be in V^ℓ for the interpolation stencil for $\mathbf{x}_p$ to be I^ℓ; vortices in $V^{\ell-1}$ which lie interior to and near (but not on) the boundary of G^ℓ will use I^ℓ. Note also that the vortices in V^ℓ correct only the velocity $\tilde{\mathbf{u}}^{\ell:\ell}$ on G^ℓ; grid points in G^ℓ are the only points within distance Ch_ℓ of the vortices in V^ℓ.

We discuss here the difference in the buffering step of the grid creation algorithm between AMR for the MLC and AMR for finite difference calculations. For AMR with the MLC, rather than adding buffer points around individual flagged points, we add buffer regions around rectangles already created at each level. There are two approaches to this buffering depending on whether we want overlapping rectangles.

In the first approach, which will generate nonoverlapping rectangles, at each level ℓ we begin by performing steps (1) and (3)-(5) once, generating feasible rectangles around the originally flagged bins. Next we add a buffer of $b+1$ level ℓ points in each coordinate direction from the boundary of the rectangle, and flag all the points now contained in all the rectangles, including the new points. We perform steps (1) and (3)-(6) now with this new list of points. The new rectangles will not overlap, but may share boundaries. We repeat the process for sequentially coarser levels until grids at all levels $\ell > 1$ are defined. The advantage of nonoverlapping rectangles is that in the multigrid relaxation, we need not worry about transmission of information between rectangles at the finest level of the composite grid. The disadvantage is that the bisection algorithm may create a division in the middle of an originally

flagged region, and thus vortices which would for overlapping rectangles be level ℓ vortices are instead level $\ell - 1$ vortices. This will not worsen the accuracy, but will reduce the savings of the mesh refinement.

The second approach is: starting at the top level $\ell = \ell_{max}$, perform steps (1) and (3)-(5) once at level ℓ, then add a buffer of $b+1$ level ℓ points in each coordinate direction from the boundary of the rectangle once it is created by the algorithm. After the buffer is added, define each newly enlarged rectangle as a grid, and repeat at level $\ell - 1$, until $\ell = 0$. This will potentially create overlapping grids. The disadvantage is that solving the Poisson equation takes more iterations; the advantage is that every vortex is represented at the highest level appropriate.

There is one exception to the buffering procedure: we allow the fine grid boundaries to exist at physical boundaries, without being buffered by intermediate grids. This is necessary for the problems in which the vorticity extends to the boundaries.

The MLC with AMR is outlined in greater detail below, using the same numbering of steps as in Chapter 3. Once the grid hierarchy is created and the vortices are sorted, for each level ℓ, $\ell = 0, ..., \ell_{max}$:

(1) For each level ℓ bin $B^{\mathbf{i}:\ell}$ containing vortices in V^ℓ, define

$$R^{\mathbf{i}:\ell} = \bigcup_{\mathbf{m}:\mathbf{m}\in G^\ell, |\mathbf{m}-\mathbf{i}|_B \le (D+1)} B^{\mathbf{m}:\ell},$$

$$R_0^{\mathbf{i}:\ell} = \bigcup_{\mathbf{m}:\mathbf{m}\in G^\ell, |\mathbf{m}-\mathbf{i}|_B \le D} B^{\mathbf{m}:\ell}.$$

where the $|\cdot|_B$ norm is defined such that $|\mathbf{i}-\mathbf{m}|_B$ is the minimum distance (in units of the mesh spacing h_ℓ) between any point in $B^{\mathbf{i}:\ell}$ and any point in $B^{\mathbf{m}:\ell}$.

(a) Compute by direct interaction the exact velocity at every level ℓ grid point $\mathbf{m}$ in $R^{\mathbf{i}:\ell}$ due to every level ℓ vortex n in $B^{\mathbf{i}:\ell}$ with no core function effects:

$$\mathbf{u}_{\mathbf{m}}^{e,h_\ell} = \sum_{n:n\in V^\ell, \mathbf{x}_n \in B^{\mathbf{i}:\ell}} \mathbf{K}(\mathbf{m}h_\ell - \mathbf{x}_n)\,\omega_n.$$

(b) Calculate the discrete Laplacian on the level ℓ grid of this velocity field at every point in $R_0^{\mathbf{i}:\ell}$. Define

$$\mathbf{g}^{\mathbf{i}:\ell} = \begin{cases} \Delta^{h_\ell}\mathbf{u}^{e,h_\ell} & \text{inside } R_0^{\mathbf{i}:\ell} \\ \mathbf{0} & \text{in interior}(G^\ell) - R_0^{\mathbf{i}:\ell}. \end{cases}$$

(2) Superimpose these fields $\mathbf{g}^{\mathbf{i}:\ell}$ to form

$$\mathbf{g}^{G^\ell} = \sum_{\mathbf{i}} \mathbf{g}^{\mathbf{i}:\ell}.$$

(3) Solve the Poisson equation on the composite grid $G^{0:\ell}$, with $\mathbf{g}^{G^\ell}$ as the right-hand side, for the velocity $\tilde{\mathbf{u}}^{\ell:m}$ on G^m in $G^{0:\ell}$. Multigrid relaxation on a hierarchical mesh is described in Chapter 5.

Define for every level ℓ bin $B^{\mathbf{i}:\ell}$,

$$S^{\mathbf{i}:\ell} = \bigcup_{\mathbf{m}:\mathbf{m}\in G^\ell, |\mathbf{m}-\mathbf{i}|_B \le C} B^{\mathbf{m}:\ell}.$$

Then for each level j bin $B^{\mathbf{i}:j}$ at each level j, $j = \ell_{max}, ..., 0$, define m as the finest level at which $B^{\mathbf{i}:j}$ is fully contained in the interior of G^m. Note that $m = j$ only if $B^{\mathbf{i}:j}$ is in the interior of G^j and not contained in the interior of any finer grid. If $B^{\mathbf{i}:j}$ is on the boundary of G^j and not contained in any finer grid then $m = j - 1$.

(4) Compute

$$\hat{\mathbf{u}}^{\ell:m}(\mathbf{X}_\alpha) := \begin{cases} \tilde{\mathbf{u}}^{\ell:m}(\mathbf{X}_\alpha) & \text{if } j < \ell \\ \tilde{\mathbf{u}}^{\ell:\ell}(\mathbf{X}_\alpha) - \sum_{n:n\in V^\ell, \mathbf{x}_n \in S^{\mathbf{i}:\ell}} \mathbf{K}(\mathbf{X}_\alpha - \mathbf{x}_n)\,\omega_n & \text{if } j = \ell \end{cases}.$$

at the interpolation points $\{\mathbf{X}_\alpha\}$ on G^m.

(5) Interpolate $\hat{\mathbf{u}}^{\ell:m}$ from the interpolation points $\mathbf{X}_\alpha$ onto each vortex p in $B_{\mathbf{i}}$ which does not lie in a finer level bin in the interior of a finer level grid:

$$\mathbf{u}^\ell(\mathbf{x}_p) := I^m(\hat{\mathbf{u}}^{\ell:m}(\mathbf{X}_\alpha); \mathbf{x}_p)$$

Note that it is possible for two vortices in the same level j bin not to have the same interpolation stencil, since one may lie in a level $j+1$ bin in the interior of G^{j+1} while the other lies in a level $j+1$ bin on the boundary of G^{j+1}. The first would use the interpolation stencil I^{j+1}, while the second would use I^j.

(6) For only those vortices p in $B^{\mathbf{i}:\ell}$ using the interpolation stencil I^ℓ, add the velocity due to every level ℓ vortex n in $S^{\mathbf{i}:\ell}$ to the existing velocity using $\mathbf{K}_\delta$ rather than $\mathbf{K}$:

$$\mathbf{u}^\ell(\mathbf{x}_p) := \mathbf{u}^\ell(\mathbf{x}_p) + \sum_{n:n\in V^\ell, \mathbf{x}_n \in S^{\mathbf{i}:\ell}} \mathbf{K}_\delta(\mathbf{x}_p - \mathbf{x}_n)\omega_n.$$

There is a certain asymmetry in the local corrections calculation, because the statement that vortex k is *near* to vortex p no longer implies that vortex p is *near* to vortex k, since the definition of near depends on the level at which the vortex is located. Thus it is possible that vortex k can be used to locally correct the velocity

of p, but vortex p is not used to correct the velocity k. This is in fact the correct way to define the local corrections, since this asymmetry is present in the original far-field calculation, and an asymmetric correction is appropriate. The correction must be consistent with the construction of the initial velocity field: if the right-hand side in the first step above is computed out to D level ℓ grid points, then the local corrections should extend out C level ℓ bins.

The algorithm as described above applies in two or three space dimensions. The savings gained by adding AMR to MLC is greater in three than two dimensions, however, because the vorticity typically is concentrated in a smaller fraction of the domain.

4.3 Error and Timing Results

When AMR is added to the method of local corrections, we see substantial savings at large N. For few vortices, the direct method is more cost-effective than MLC or MLC with AMR because of the overhead due to the presence of a grid, but for many vortices MLC with AMR requires considerably less computational effort than the direct method.

Table 4.1 shows the relative L_2 norm of the error in the velocity field at the vortices in a three-dimensional vortex ring calculation as compared to the direct method. The core function used here is that presented in Chapter 1, and $\delta = h_v^{.75}$. We see very little variation in the error as the base grid and the maximum level of refinement vary. The exceptions are: (a) on a uniform $8 \times 8 \times 8$ grid the correction radius encloses all the vortices, thus the error of the MLC relative to the direct

LEVEL	Relative L_2 Error	
	16232 vortices	8011 vortices
8	1.6e-5	1.6e-4
8-16	1.9e-3	2.2e-3
8-32	1.8e-3	2.0e-3
8-64	7.9e-4	8.8e-4
16	7.1e-4	7.8e-4
16-32	6.9e-4	7.7e-4
16-64	7.5e-4	8.2e-4
16-128	7.6e-4	
32	7.1e-4	7.8e-4
32-64	7.5e-4	8.2e-4
32-128	7.6e-4	

Table 4.1: Relative L_2 norm of the error for different levels of mesh refinement. Calculations for a three-dimensional vortex ring of radius .1 around the z-axis and cross-sectional radius .02.

method is very low, and (b) For a base grid of $8 \times 8 \times 8$ with $\ell_{max} > 0$, the error is higher than for a base grid of $16 \times 16 \times 16$ or $32 \times 32 \times 32$ because there is some loss of accuracy in the boundary conditions for the coarser grid, and this error is transmitted to the velocity at interior points through the solution of the Poisson equation.

The first column in this table specifies the level of refinement; "8" refers to a uniform $8 \times 8 \times 8$ grid, "8-32" refers to a $8 \times 8 \times 8$ base grid with two levels of refinement above that (so that the finest level has $h = 1/32$). The second column is the relative L_2 norm of the error for a ring with 8011 vortices, the third column is the relative L_2 norm of the error for a ring with 16232 vortices. The ring has a radius of .1 around the z-axis and cross-sectional radius .02.

Table 4.2 displays results of timing comparisons for a single velocity evaluation at N endpoints between the direct method, the MLC with a uniform grid, and MLC with AMR. We see that at $N \approx 3000$ the MLC becomes faster than the

N	Time (CPU seconds) (Grid)		
	Direct	MLC	MLC with AMR
1023	.34	.69 (8)	.69 (8)
2016	1.48	1.83 (8)	1.83 (8)
3940	4.92	4.61 (8)	4.39 (8-16)
8766	24.2	15.8 (16)	11.1 (8-16)
17641	98.0	49.5 (32)	24.5 (8-64)
31988	322.6	105.7 (32)	49.3 (8-64)
63759	1281.	314.5 (64)	104.3 (8-128)

Table 4.2: Timings for a single velocity evaluation using the direct method, MLC and MLC with AMR, for calculations of a three-dimensional vortex ring. Timings are on a Cray Y-MP with the cft77 compiler.

direct method, and at $N \approx 64000$ the MLC with AMR is approximately three times faster than the MLC on a uniform grid, and over twelve times faster than the direct method. All MLC with AMR calculations have a base grid of $8 \times 8 \times 8$. The uniform grid calculations use grids ranging from $8 \times 8 \times 8$ for $N \leq 4000$ to $64 \times 64 \times 64$ for $N \approx 64000$; the MLC with AMR has refinement varying from $\ell_{max} = 0$ for $N < 4000$ to $\ell_{max} = 4$ for $N \approx 64000$; these timings are for the optimal grid for each method. Again these results are from vortex ring calculations; here the ring has a radius of .1 around the z-axis and cross-sectional radius .0275. All the calculations in Tables 4.1 and 4.2 have $C = D = 1.5$, a value found to give sufficient accuracy with this core function in a variety of calculations.

All timings were done on a Cray Y-MP with the cft77 compiler. Note that we would expect a vector machine to improve the performance of the direct method relative to the MLC, since the direct calculation is simple and can be completely vectorized. Thus, the savings from using MLC or MLC with AMR would be expected to be even greater on a scalar machine. See Baden [7] for a discussion of how vectorization affects the timings of the MLC in two dimensions.

4.4 Optimal Refinement Criteria

The issue of where and by how many levels to refine the mesh in MLC with AMR is an important one, because it is the correct choice of refinement which allows such dramatic savings in cost and time. We saw in Table 4.1 above that the error in the velocity evaluation is independent of mesh spacing within certain limits. As we will see in Chapter 6, for a fixed $C = D$, using too fine a mesh relative to the effective radius of the core function results in a loss of accuracy. The accuracy of the MLC algorithm relies on the radius for local corrections being sufficiently larger than the effective radius of the core function.

Ideally, we would like to have a very simple criterion for refinement, e.g., refine whenever the number of vortices per bin is above N_{max}, where N_{max} is the same for different levels and different distributions. Unfortunately, this is not a sophisticated enough criterion when adaptive refinement is allowed. As can be seen in Table 4.3, which presents the timings on a Cray Y-MP for different levels of possible refinement in calculations of a three dimensional ring with $N = 17641$, refining naively can make the algorithm much more costly.

The first step in developing an algorithm for choosing the optimal mesh refinement is determining which operations in the algorithm vary in cost as the mesh spacing varies. Table 4.4 shows the dependencies for a velocity evaluation in the MLC with a uniform mesh in d dimensions, assuming a uniform distribution of vortices in a fixed region of the domain, as a function of number of vortices N, total number of grid points, M, and number of bins containing vortices, M_v.

Table 4.5 shows the CPU time for the different stages of the velocity evaluation

LEVEL	Time (CPU secs)
8	71.4
8-16	52.7
8-32	29.1
8-64	24.5
8-128	31.8
16	49.5
16-32	31.9
16-64	27.7
16-128	33.0
32	53.0
32-64	49.4
32-128	40.7

Table 4.3: Timings for a single velocity evaluation with different possible refinements of the grid for calculations of a three-dimensional vortex ring with $N = 17641$. Note that 8-64 is the optimal refinement here; this was the timing seen in Table 4.2. The time for the direct evaluation is 98.02 seconds. Timings are on a Cray Y-MP with the cft77 compiler.

Operation	Cost is proportional to
Calculation of $\mathbf{g}^{\Omega^0}$	N,M_v
Direct calculation of boundary conditions	$NM^{\frac{d-1}{d}}$
Solution of Poisson equation	$M \log M$
Correction and interpolation of velocity field	N
Direct local interactions	$\frac{N^2}{M_v}$

Table 4.4: Operation count for the MLC on a uniform grid with M total grid points, M_v bins containing vortices, and N vortices.

Operation	Time (CPU secs)			
	8-32	8-64	8-128	32
Calculation of $\mathbf{g}^{\Omega^{l_{max}}}$	1.4	1.8	3.9	1.4
Direct calculation of boundary conditions	3.2	3.2	3.2	10.3
Solution of Poisson equation	1.4	3.8	10.4	11.7
Correction and interpolation of velocity field	6.0	8.6	10.7	6.8
Direct local interactions	17.6	7.2	3.3	19.5
Total time for full velocity evaluation	29.7	24.6	31.7	49.5

Table 4.5: Time per operation in the MLC, with three different levels of AMR and for optimal uniform grid case. Timings are on a Cray Y-MP with the cft77 compiler.

for three calculations using MLC with AMR and the fastest uniform grid calculation. The initial data is a vortex ring with $N = 17641$, as for Table 4.3. In each case the base grid is $8 \times 8 \times 8$ (hence the cost of boundary conditions does not change), and ℓ_{max} varies from 2 to 4.

The "8-64" calculation is the fastest of the four, and we see that the savings relative to the uniform grid calculation are mostly in solving the Poisson equation (even though the top level of the refined mesh is finer than the uniform mesh), in the local corrections, and in the boundary conditions. We expect the cost of the boundary conditions to be proportional to the number of points on the boundary of the base grid, but here the cost of the boundary conditions increases by less than a factor of 4 for a 16-fold increase in number of boundary points. This is due to the fact that the vector lengths in this calculation are longer for the larger calculation. Thus the time for the calculation is not a simple linear function of the total number of operations.

The cost of calculating boundary conditions for the MLC can be reduced in two ways: (1) by adding AMR so that the base grid, on which the boundary conditions are calculated, can be kept relatively coarse, and (2) by implementing fast boundary methods as discussed in Chapter 5. We defer further discussion of boundary conditions to Chapter 5, except to note that the cost of boundary conditions is not a strong consideration in the choice of ℓ_{max} because the boundary conditions are only computed on the base level grid. The boundary conditions must be computed for every level containing vortices, but each vortex enters the boundary calculation only once.

We now consider the cost of solving the Poisson equation and the cost of the local

corrections. Assume a uniform distribution of vortices on a fixed region of a single uniform mesh, with N, M and $M_v = fM$ as defined above, with f the fraction of bins containing vortices. We show below that the optimal mesh spacing is that for which the time for solving the Poisson equation and the time for computing the direct interactions are equal. Alternately, maintaining the optimal level of refinement requires keeping N/M constant.

The claim follows directly from writing the time to solve the Poisson equation as $T_1 = c_1 M \log M$, and the time for the local interactions as $T_2 = c_2 N^2/(fM)$, where c_1 and c_2 are constants. We approximate $M \log M$ by M, and then the combined time for both operations is $T = T_1 + T_2 = c_1 M + c_2 N^2/(fM)$. The minimum of T with respect to M occurs at $M = N\sqrt{c_2/fc_1}$. The values of T_1 and T_2 at this minimum is are $T_1 = T_2 = N\sqrt{(c_1 c_2)/f}$. Thus the minimum combined time occurs when $T_1 = T_2$, and the value of M such that T is a minimum is proportional to N.

In most cases it is not possible to find a mesh spacing such that the times to solve the Poisson equation and to compute the local interactions are approximately equal, but this analysis can serve as a general guide in choosing the optimal level of refinement. Unfortunately, it is not in a very useful form, since we seek a value for N_{max} to use in refining individual bins.

The cost of solving Poisson's equation using multigrid is proportional to the total number of points P at all the levels of the multigrid V-cycle: $P = \sum_{\ell=0}^{\ell_{min}} (M_{side}/2^\ell)^d$ where $\ell_{min} = \log_2 M_{side}$ for a grid in d dimensions with M_{side} points per side. This can be approximated by the number of points on the finest level grid for uniform grid relaxation, since the sum is dominated by the M_{side}^d term in that case. However, when we evaluate the cost of using multigrid on a hierarchical refined grid, the

approximation may no longer be valid, since the finer levels will not necessarily have many more grid points than the base level. For example, the cost of multigrid on a uniform $16 \times 16 \times 16$ grid is 8 times the cost of multigrid on a uniform $8 \times 8 \times 8$ grid; however, if only half of the grid in each coordinate direction is refined above a base grid of $8 \times 8 \times 8$, then the cost of multigrid with the partial refinement is approximately twice the cost of multigrid on the uniform $8 \times 8 \times 8$ grid.

It is possible to derive a formula expressing the time needed to solve the Poisson equation as a function of number of levels and sizes of regions of refinement at each level, using the fact that the cost of multigrid is proportional to the total number of points at which relaxation is being done. To evaluate the actual CPU time on a specific machine, we must first perform sample calculations in a test problem to find the values of the constants of proportionality which relate actual CPU time to number of levels and number of points per level.

The earlier discussion of timings assumed a uniform distribution of vortices in a region of the domain, and used this assumption to show that the time for local corrections is proportional to N^2/M_v. For problems of interest, however, the distribution is often very nonuniform, and not easily classified geometrically. For these distributions we cannot express the time for local corrections before we know the exact geometry of the distribution. However, given the locations of the vortices at each time step, we can count the number of local interactions which would be calculated, and thus at the time of execution get an estimate of the time needed for local corrections at any given level of refinement.

The algorithm we propose is: at the beginning of a time step, for each possible level of refinement evaluate the time needed to solve the Poisson equation and the

time to do the local corrections, using the operation counts and timings from the machine on which we are computing. Find the lowest of these times, and choose the associated level of refinement for the actual calculation. This process need not be repeated every time step, since the geometry does not change significantly in a few time steps, and the choice of optimal refinement is not sensitive to small changes in geometry. See [2] for further discussion of this type of algorithm.

The difficulty with this method is figuring out which combinations of rectangles to evaluate. The easiest option is to first count the maximum number of vortices per bin at each level of refinement up to the maximum level allowed. Divide this maximum into, say, four increments, and let N_{max} sequentially take the value of each of these four numbers. For example, if we find for a given mesh spacing that there are at most 120 vortices per bin, we would evaluate the cost of the algorithm with $N_{max} = 0, N_{max} = 30, N_{max} = 60, N_{max} = 90$ at that level. For each case generate a grid hierarchy, and find the operation count associated with the composite grid. Then evaluate the times for these operations using the proportionality constants established earlier, and compare the total times to find N_{max} which yields the fastest calculation. In practice, we would not refine more than four or five levels above the base grid, so the number of cases to be evaluated is limited.

The reason we choose to compare timings as described in the above paragraph is that the alternative, trying to decide whether to refine each bin on a bin-by-bin basis, is too complicated. Since the regions of refinement must be rectangles, refining any one bin may have as a consequence the refinement of many other bins, or alternatively it may have no consequence at all, since this bin would have been refined anyway as one of the "unflagged" bins in a rectangle. Thus it is not practical

to use a bin-by-bin refinement criterion, and the method we use should be sufficient.

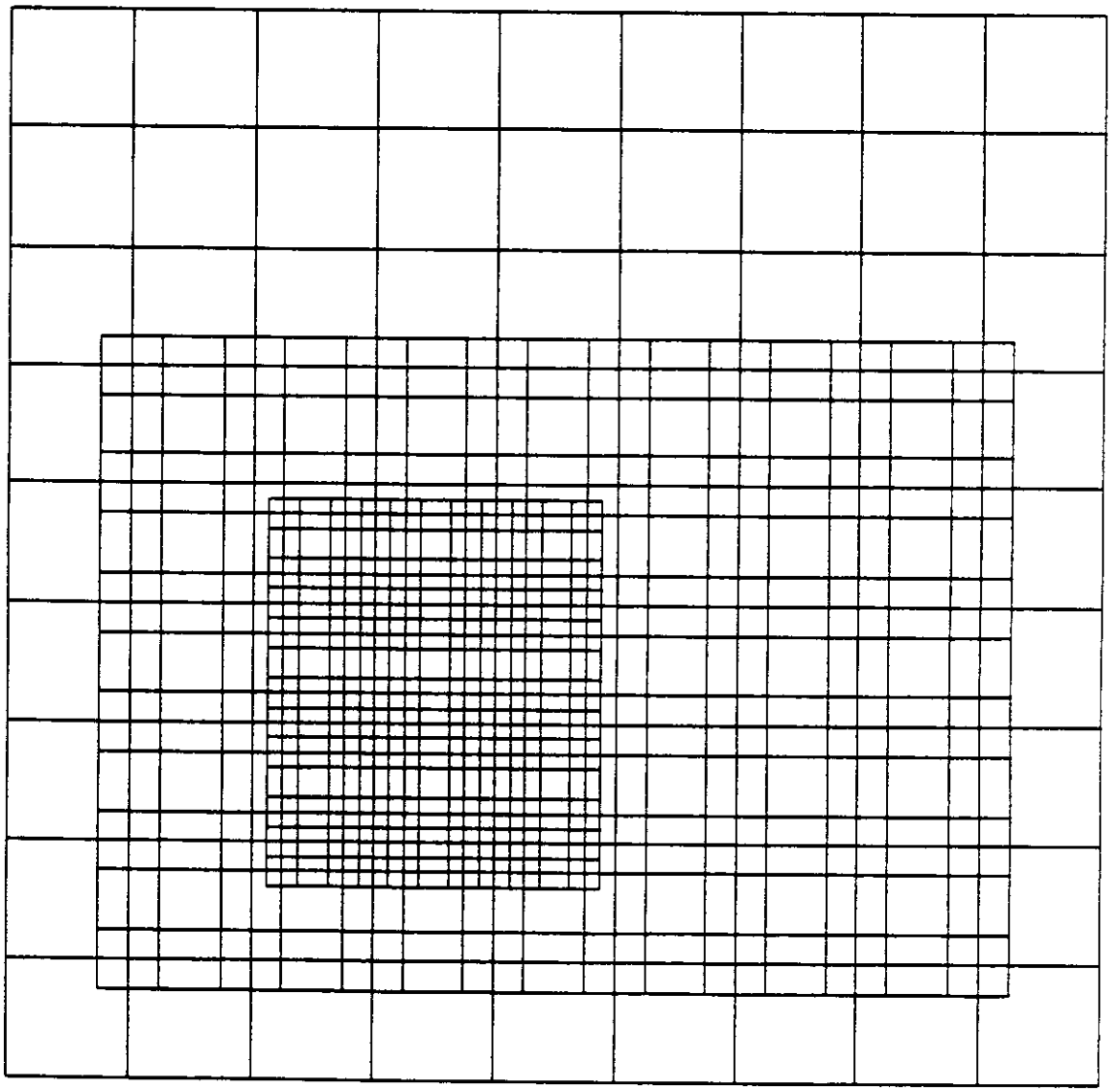


Figure 4.1: Sample composite grid at level 2.

CHAPTER 5

Grid-Related Topics

The standard vortex method is a grid-free method; the method of local corrections introduces a grid overlaying the vortices. Adaptive mesh refinement introduces new complexity to the grid-related parts of the algorithm. In this chapter we describe in detail the parts of the algorithm which are defined on the grid: the two- and three-dimensional discrete Laplacians; the two- and three-dimensional interpolation stencils; the solution of the Poisson equation using multigrid, with and without AMR; and a variety of boundary conditions.

5.1 Finite Difference Stencils

5.1.1 Discrete Laplacian

In two dimensions, we use a nine-point stencil for the discrete Laplacian, which is $O(h^4)$ for general u, $O(h^6)$ for harmonic functions:

$$\begin{aligned}(\Delta^h u)_{i,j} &= \frac{1}{6h^2}(-20u_{i,j} + 4(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}) + \\ &\quad u_{i-1,j-1} + u_{i+1,j-1} + u_{i-1,j+1} + u_{i+1,j+1})\end{aligned}$$

In three dimensions, consider the cube of points $(\mathbf{i}+\mathbf{s}) = (i+s_1, j+s_2, k+s_3)$, $|s_i| \leq 1$, immediately surrounding the point $\mathbf{i}$. Define *face points* on the cube such that $|s_1| + |s_2| + |s_3| = 1$. Similarly define *edge points* by $|s_1| + |s_2| + |s_3| = 2$, and let *corner points* be those satisfying $|s_1| + |s_2| + |s_3| = 3$.

Then the 27-point Laplacian can be written

$$(\Delta^h u)_{\mathbf{i}} = \frac{1}{h^2}(-\frac{128}{60}u_{\mathbf{i}} + \frac{1}{60}\sum_{corner\ points} u_{\mathbf{i+s}} + \frac{1}{20}\sum_{edge\ points} u_{\mathbf{i+s}} + \frac{7}{30}\sum_{face\ points} u_{\mathbf{i+s}}).$$

These formulae for the discrete Laplacians can be found in [27].

5.1.2 Interpolation Stencils

In the two dimensional MLC, Anderson [1] exploits the fact that the velocity induced by a point vortex is a potential flow field away from the vortex to construct an interpolation function with a highly compact stencil relative to its accuracy. Since the flow field is potential, the velocity components u and v are the real and imaginary parts, respectively, of a complex analytic function, and thus we can use a complex version of Lagrange's formula for polynomial interpolation [30].

The complex interpolation stencil around the point (k, ℓ) that we use for two dimensional calculations has five points, $z_m = (k + i\ell) + s_m$, where $s_1 = 0 + 0i$, $s_2 = 1 + 0i$, $s_3 = -1 + 0i$, $s_4 = 0 + 1i$, and $s_5 = 0 - 1i$. The interpolation algorithm is

$$(u - iv)(z) = \sum_{m=1}^{5} a_m \mathbf{u}_m,$$

where $\mathbf{u}_m = (u - iv)(z_m)$ and

$$a_m = \frac{\prod_{i=1, i\neq m}^{5}(z - z_i)}{\prod_{i=1, i\neq m}^{5}(z_m - z_i)}.$$

In three dimensions, the velocity field induced by a single vortex element is no longer a potential flow field away from the support of the vorticity. However, the

velocity is divergence-free, and the Laplacian of each of its components vanishes. Buttke and Colella take advantage of these features to construct an accurate interpolation function with a compact stencil, and we present it below.

Consider that we want to interpolate a scalar function u onto position (x_0, y_0, z_0) from an interpolation stencil centered at grid point (i, j, k) of a uniform grid with mesh spacing h. Assume that (x_0, y_0, z_0) lies closer to (ih, jh, kh) than to any other grid point. Define $x = x_0 - ih, y = y_0 - jh, z = z_0 - kh$. We see that $|x| \leq \frac{h}{2}, |y| \leq \frac{h}{2}, |z| \leq \frac{h}{2}$. Using a Taylor expansion, we can write

$$\begin{aligned} u(x_0, y_0, z_0) &= u(ih, jh, kh) \\ &+ xu_x + yu_y + zu_z \\ &+ \tfrac{1}{2}(x^2 u_{xx} + y^2 u_{yy} + z^2 u_{zz}) + xyu_{xy} + yzu_{yz} + xzu_{xz} \\ &+ \tfrac{1}{6}(x^3 u_{xxx} + y^3 u_{yyy} + z^3 u_{zzz}) \\ &+ \tfrac{1}{2}(x^2 y u_{xxy} + xy^2 u_{xyy} + x^2 z u_{xxz} + xz^2 u_{xzz} + y^2 z u_{yyz} + yz^2 u_{yzz}) \\ &+ xyzu_{xyz} + O(h^4) \end{aligned} ,$$

where $u_x = \frac{\partial u}{\partial x}$, $u_{xy} = \frac{\partial^2 u}{\partial x \partial y}$, $u_{xyz} = \frac{\partial^3 u}{\partial x \partial y \partial z}$, and so on. All derivatives here are evaluated at (ih, jh, kh).

To create a fourth-order interpolation scheme, we must approximate the first derivatives to $O(h^3)$, the second derivatives to $O(h^2)$, and the third derivatives to $O(h)$, since x, y, and z are of $O(h)$.

Define

$$\begin{aligned} s_x^+ &= u_{i+1,j,k+1} + u_{i+1,j,k-1} + u_{i+1,j-1,k} + u_{i+1,j+1,k} \\ s_x^- &= u_{i-1,j,k+1} + u_{i-1,j,k-1} + u_{i-1,j-1,k} + u_{i-1,j+1,k} \\ f_x &= (s_x^+ - s_x^- + 2(u_{i+1,j,k} - u_{i-1,j,k}))/(12h), \end{aligned}$$

$$s_y^+ = u_{i,j+1,k+1} + u_{i,j+1,k-1} + u_{i+1,j+1,k} + u_{i-1,j+1,k}$$

$$s_y^- = u_{i,j-1,k+1} + u_{i,j-1,k-1} + u_{i+1,j-1,k} + u_{i-1,j-1,k}$$

$$f_y = (s_y^+ - s_y^- + 2(u_{i,j+1,k} - u_{i,j-1,k}))/(12h),$$

and

$$s_z^+ = u_{i,j+1,k+1} + u_{i,j-1,k+1} + u_{i+1,j,k+1} + u_{i-1,j,k+1}$$

$$s_z^- = u_{i,j+1,k-1} + u_{i,j-1,k-1} + u_{i+1,j,k-1} + u_{i-1,j,k-1}$$

$$f_z = (s_z^+ - s_z^- + 2(u_{i,j,k+1} - u_{i,j,k-1}))/(12h).$$

If we Taylor expand each term in the above expressions for f_x, f_y, f_z about (ih, jh, kh), we see by cancellation of the zeroth and all the first and second order terms that these are third-order approximations to the first derivatives.

Now define

$$f_{xx} = (u_{i+1,j,k} + u_{i-1,j,k} - 2u_{i,j,k})/h^2$$

$$f_{yy} = (u_{i,j+1,k} + u_{i,j-1,k} - 2u_{i,j,k})/h^2$$

$$f_{zz} = (u_{i,j,k+1} + u_{i,j,k-1} - 2u_{i,j,k})/h^2$$

$$f_{xy} = ((u_{i+1,j+1,k} - u_{i-1,j+1,k}) - (u_{i+1,j-1,k} - u_{i-1,j-1,k}))/(4h^2)$$

$$f_{xz} = ((u_{i+1,j,k+1} - u_{i-1,j,k+1}) - (u_{i+1,j,k-1} - u_{i-1,j,k-1}))/(4h^2)$$

$$f_{yz} = ((u_{i,j+1,k+1} - u_{i,j-1,k+1}) - (u_{i,j+1,k-1} - u_{i,j-1,k-1}))/(4h^2)$$

These are the standard second-order centered-difference expressions for the second derivatives of a function.

Now, since we only need the third derivatives to be first order in h, we approximate u_{xxy} by a centered difference in y of the above expression for f_{xx}. The above expression was second-order; by approximating the first derivative of that we lose an order of accuracy, but first order is all we need. Similarly for the other third derivatives. Thus

$$f_{xxy} = (((u_{i+1,j+1,k} - 2u_{i,j+1,k} + u_{i-1,j+1,k}) - (u_{i+1,j-1,k} - 2u_{i,j-1,k} + u_{i-1,j-1,k}))/(2h^3)$$

$$f_{xxz} = ((u_{i+1,j,k+1} - 2u_{i,j,k+1} + u_{i-1,j,k+1}) - (u_{i+1,j,k-1} - 2u_{i,j,k-1} + u_{i-1,j,k-1}))/(2h^3)$$

$$f_{zzy} = ((u_{i,j+1,k+1} - 2u_{i,j+1,k} + u_{i,j+1,k-1}) - (u_{i,j-1,k+1} - 2u_{i,j-1,k} + u_{i,j-1,k-1}))/(2h^3)$$

$$f_{zzx} = ((u_{i+1,j,k+1} - 2u_{i+1,j,k} + u_{i+1,j,k-1}) - (u_{i-1,j,k+1} - 2u_{i-1,j,k} + u_{i-1,j,k-1}))/(2h^3)$$

$$f_{yyx} = ((u_{i+1,j+1,k} - 2u_{i+1,j,k} + u_{i+1,j-1,k}) - (u_{i-1,j+1,k} - 2u_{i-1,j,k} + u_{i-1,j-1,k}))/(2h^3)$$

$$f_{yyz} = ((u_{i,j+1,k+1} - 2u_{i,j,k+1} + u_{i,j-1,k+1}) - (u_{i,j+1,k-1} - 2u_{i,j,k-1} + u_{i,j-1,k-1}))/(2h^3)$$

$$\begin{aligned} f_{xyz} = &(((u_{i+1,j+1,k+1} - u_{i-1,j+1,k+1}) - (u_{i+1,j-1,k+1} - u_{i-1,j-1,k+1})) \\ &-((u_{i+1,j+1,k-1} - u_{i-1,j+1,k-1}) - (u_{i+1,j-1,k-1} - u_{i-1,j-1,k-1})))/(8h^3) \end{aligned}$$

In order to maintain a compact stencil (i.e., one contained in the $3 \times 3 \times 3$ cube surrounding the center point), instead of approximating u_{xxx} by the first derivative in x of u_{xx}, we use the fact that u is harmonic to write

$$\begin{aligned} u_{xxx} &= (u_{xx})_x = (-u_{yy} - u_{zz})_x = -u_{xyy} - u_{xzz}, \\ u_{yyy} &= (u_{yy})_y = (-u_{xx} - u_{zz})_y = -u_{xxy} - u_{yzz}, \\ u_{zzz} &= (u_{zz})_z = (-u_{xx} - u_{yy})_z = -u_{xxz} - u_{yyz}. \end{aligned}$$

Then the Taylor expansion becomes:

$$\begin{aligned} u(x_0, y_0, z_0) &= u(ih, jh, kh) \\ &+ xu_x + yu_y + zu_z \\ &+ \tfrac{1}{2}(x^2 u_{xx} + y^2 u_{yy} + z^2 u_{zz}) + xyu_{xy} + yzu_{yz} + xzu_{xz} \end{aligned}$$

$$+\tfrac{1}{6}((3x^2-y^2)yu_{xxy}+(3x^2-z^2)zu_{xxz}+(3y^2-x^2)xu_{yyx}+(3y^2-z^2)zu_{yyz}$$
$$+(3z^2-x^2)xu_{zzx}+(3z^2-y^2)yu_{zzy}).+xyzu_{xyz}+O(h^4),$$

and the interpolation scheme, in terms of the above defined terms, can be written

$$u(x_0,y_0,z_0)=u(ih,jh,kh)$$
$$+xf_x+yf_y+zf_z$$
$$+\tfrac{1}{2}(x^2f_{xx}+y^2f_{yy}+z^2f_{zz})+xyf_{xy}+yzf_{yz}+xzf_{xz}$$
$$+\tfrac{1}{6}((3x^2-y^2)yf_{xxy}+(3x^2-z^2)zf_{xxz}+(3y^2-x^2)xf_{yyx}+(3y^2-z^2)zf_{yyz}$$
$$+(3z^2-x^2)xf_{zzx}+(3z^2-y^2)yf_{zzy})+xyzf_{xyz}+O(h^4).$$

5.2 Multigrid

The method of local corrections requires the solution of the Poisson equation, $\Delta^h\tilde{\mathbf{u}}=\mathbf{g}^{\Omega^0}$ on the domain Ω^0. We use multigrid [18] with Gauss-Seidel relaxation, red-black ordering, and V-cycles to do this inversion. See [19] for an introduction to and further description of multigrid techniques.

Consider the equation, $Au=\rho$, where A is a linear operator. In our problem, $A=\Delta^h$, $\rho=\mathbf{g}^{\Omega^0}$. Define u as the exact solution to $Au=\rho$, and v as the numerical approximation to u. Multigrid is a multilevel relaxation method, i.e., it solves this equation by iterating on the equation $v^{m+1}=v^m+\lambda(Au^m-\rho)$, where λ is the relaxation parameter and m is the relaxation counter, until v is sufficiently close to the exact solution u. We define λ so that the term $\bar{v}_{\mathbf{i}}^m$ does not appear in the

right-hand side of the equation for $v_{\mathbf{i}}^{m+1}$, i.e. $\lambda = -C_0$, where C_0 is the coefficient of the $v_{\mathbf{i}}$ term in the definition of $(\Delta^h v)_{\mathbf{i}}$.

Define $e = u - v$. Since u is an unknown, e must also be unknown; however, $e = 0$ implies $v = u$, i.e., convergence of the numerical solution to the exact solution of the original equation. Define the residual $R = \rho - Av$. Substituting $u = v + e$ into $Au = \rho$ gives $A(v + e) = \rho$, and using the definition of the residual gives $Ae = R$; this is the residual equation. Solving the residual equation is equivalent to solving the original equation $Au = \rho$. We will relax on the residual equation rather than the original equation.

Two common relaxation methods are Gauss-Seidel and Jacobi relaxation; Gauss-Seidel relaxation differs from Jacobi relaxation in that new values of v^{m+1} are used as soon as they are created. In the Jacobi method, all values of v^{m+1} depend only on values of v^m; in the Gauss-Seidel method, values of v^{m+1} depend on values of v^{m+1} as well as values of v^m. Red-black ordering refers to the order in which the $u_{\mathbf{i}}^m$ are updated. Thinking of the grid in two dimensions as a checkerboard, we first update all the red squares, then using the new values we update all the black squares. For the standard 5-point Laplacian in two dimensions, using λ as defined above, the "red" points and "black" points decouple from each other, in that the values at the red points depend only on the values at the black points, and reciprocally. For the 9-point stencil of the discrete Laplacian in two dimensions and the 27-point stencil in three dimensions, the alternate grid points do not decouple.

The problem with single-level relaxation is that while high frequencies are damped efficiently, low frequencies persist for many iterations. Multigrid was developed to speed up the iteration process, and relies on the fact that what is a low-frequency

mode on a fine grid is a higher-frequency mode on a coarser grid. Thus in a single V-cycle of multigrid, residual equations are successively coarsened onto lower levels, until $Ae = R$ is solved exactly at the lowest level. Corrections e are then interpolated back up to the finest level, where they are added to the solution v. Boundary conditions for the relaxation at coarse levels are homogeneous Dirichlet if the original boundary conditions are Dirichlet; they are periodic if the original boundary conditions are periodic.

In two dimensions, we define the coarsening operator I^C, such that $R^{coarse} = I^C R^{fine}$, for i, j even:

$$R^{coarse}_{i/2,j/2} = \frac{1}{16}(4R^{fine}_{i,j} + 2(R^{fine}_{i+1,j} + R^{fine}_{i-1,j} + R^{fine}_{i,j+1} + R^{fine}_{i,j-1})$$

$$+(R^{fine}_{i+1,j+1} + R^{fine}_{i+1,j-1} + R^{fine}_{i-1,j+1} + R^{fine}_{i-1,j-1}))$$

In three dimensions, recall the definition of *face* points, *edge* points, and *corner* points from Section 5.1. Then we define the coarsening operator, I^C, such that $R^{coarse} = I^C R^{fine}$, for i all even, by

$$R^{coarse}_{i/2} = \frac{1}{64}(8R^{fine}_{i} + \sum_{corner\ points} R^{fine} + 2 \sum_{edge\ points} R^{fine} + 4 \sum_{face\ points} R^{fine})$$

The coarsening operator, I^C, is used for the residuals, but to coarsen the correction or velocity we use a point-wise coarsening, P, such that

$$(Pe^n)_i = e^n_{2i}.$$

We define the interpolation operator I^F in the interior of a grid as simple bilinear

interpolation in two dimensions, trilinear interpolation in three dimensions.

Consider now a computational grid in d dimensions, $[0, N]^d$; define $\ell_{min} = \log_2 N$, and let $h_0 = 1/N$. In keeping with the notation of Chapter 3, we define Ω^ℓ, $0 \geq \ell \geq \ell_{min}$ as the ℓ^{th} level grid. Initialize $v^0 = 0$ in the interior of Ω^0; on $\partial\Omega^0$ it carries the boundary conditions. The solution of $\Delta^{h_0} v^0 = \rho$ is then found as follows.

$R^0 := \rho - \Delta^{h_0} v^0$

While $(|R^0| < \varepsilon)$

  $e^0 := \mathbf{MGRelax}(0, R^0, h_0)$

  $v^0 := v^0 + e^0$

  $R^0 := \rho - \Delta^{h_0} v^0$

EndWhile

Procedure $\mathbf{MGRelax}(\ell, R^\ell, h_\ell)$

$e^\ell := 0$

$e^\ell := \mathbf{Relax}(R^\ell, e^\ell, h_\ell)$

If $(\ell > \ell_{min})$ then

  $h_{\ell-1} := 2h_\ell$

  $R^{\ell-1} := I^C(R^\ell - \Delta^{h_\ell} e^\ell)$.

  $e^{\ell-1} := \mathbf{MGRelax}(\ell - 1, R^{\ell-1}, h_{\ell-1})$

  $e^\ell := e^\ell + I^F e^{\ell-1}$.

  $e^\ell := \mathbf{Relax}(R^\ell, e^\ell, h_\ell)$

Endif

Return e^ℓ

Procedure **Relax**(R^ℓ, e^ℓ, h_ℓ)

Repeat ν times:

If $((\sum_{n=1}^{d} i_n) \bmod 2 = 0)$ then

$e^\ell_{\mathbf{i}} := e^\ell_{\mathbf{i}} + \lambda((\Delta^{h_\ell} e^\ell)_{\mathbf{i}} - R^\ell_{\mathbf{i}})$

Else

$e^\ell_{\mathbf{i}} := e^\ell_{\mathbf{i}}$

Endif

If $((\sum_{n=1}^{d} i_n) \bmod 2 = 1)$ then

$e^\ell_{\mathbf{i}} := e^\ell_{\mathbf{i}} + \lambda((\Delta^{h_\ell} e^\ell)_{\mathbf{i}} - R^\ell_{\mathbf{i}})$

Else

$e^\ell_{\mathbf{i}} := e^\ell_{\mathbf{i}}$

Endif

5.3 Multigrid with AMR

When AMR is added to the MLC, we need to solve the Poisson equation on a composite grid $G^{0:\ell_{top}}$, $0 \le \ell_{top} \le \ell_{max}$. The correct equations to satisfy on $G^{0:\ell_{top}}$ are (recalling the grid notation from Chapter 4):

$$\begin{aligned}
\Delta^{h_{\ell_{top}}} u^{\ell_{top}} &= \rho^{\ell_{top}} && \text{in interior}(G^{\ell_{top}}), \\
\Delta^{h_\ell} u^\ell &= \rho^\ell = 0, \ 0 \le \ell < \ell_{top} && \text{in interior}(G^\ell) - \text{interior}(G^{\ell+1}), \\
u^\ell &= P u^{\ell+1}, 0 \le \ell < \ell_{top} && \text{in } G^\ell.
\end{aligned}$$

Note that we do not attempt to satisfy any equation with Δ^{h_ℓ} in the interior of $G^{\ell+1}$; only the finest level equation possible is satisfied at any given point in the domain. The equation containing Δ^{h_ℓ} is satisfied on the boundary of $\Omega^{\ell+1}, \ell \geq 0$, however, and this gives the appropriate matching condition.

Since the G^ℓ do not cover the entire domain Ω^0, we need to interpolate values onto the boundaries of the G^ℓ, $\ell > 0$ during each multigrid V-cycle. We use a fourth-order interpolation function, I^∂, on these boundaries.

A single V-cycle of the modified multigrid procedure is presented below, starting from the finest level $\ell = \ell_{top}$ of the composite grid $G^{0:\ell_{top}}$ on which we are solving. This procedure is initialized by setting $v^\ell, \rho^\ell = 0$ for $0 \leq \ell \leq \ell_{top}$, except that v^0 is set equal to the Dirichlet boundary conditions on ∂G^0, and $\rho^{\ell_{top}}$ is the right-hand side induced by the vortices on the finest grid.

We note here that the multigrid procedure with AMR is very similar to that starting with a uniform grid, with the exception that the solution v^ℓ is updated at each relaxation on the boundaries ∂G^ℓ as well as in the interior of G^ℓ for $\ell > 0$, and the correction e^ℓ is nonzero on ∂G^ℓ. The corrections themselves carry the boundary conditions for relaxation at the finer levels. This is in contrast to the levels $\ell < 0$ in multigrid, for which the boundary conditions are homogeneous Dirichlet.

$R^{\ell_{top}} := \rho^{\ell_{top}} - \Delta^{h_{\ell_{top}}} v^{\ell_{top}}$

While $(|R^{\ell_{top}}| < \varepsilon)$

  $\ell = \ell_{top}$

  $R^\ell := \rho^\ell - \Delta^{h_\ell} v^\ell$

  While $(\ell > 0)$

$e^\ell := 0$

$e^\ell := \mathbf{Relax}(R^\ell, e^\ell, h_\ell)$

$v^{\ell-1} := P(v^\ell + e^\ell)$

$$R^{\ell-1} := \begin{cases} I^C(R^\ell - \Delta^{h_\ell} e^\ell) & \text{in interior}(G^\ell) \\ -\Delta^{h_{\ell-1}} v^{\ell-1} & \text{in interior}(G^{\ell-1}) - \text{interior}(G^\ell) \end{cases}$$

$\ell := \ell - 1$

EndWhile

$e^0 := \mathbf{MGRelax}(0, R^0, h_0)$

$v^0 := v^0 + e^0$

$\ell = 1$

While $(\ell \leq \ell_{top})$

$e^\ell := e^\ell + I^F e^{\ell-1}$ in interior(G^ℓ)

$e^\ell := e^\ell + I^\partial e^{\ell-1}$ on ∂G^ℓ

$e^\ell := \mathbf{Relax}(R^\ell, e^\ell, h_\ell)$

$v^\ell := v^\ell + e^\ell$

$\ell := \ell + 1$

EndWhile

EndWhile

Although this is written above in nonrecursive form for clarity, in the actual implementation relaxation at successively finer grids is performed recursively.

5.4 Boundary Conditions

For this section on boundary conditions, unless stated otherwise we assume that all vortices in the domain are $D + 1$ or more bins from the boundaries, where D is

the spreading distance. It is possible to compute boundary conditions for vortices near boundaries, but there are additional complications with the method of local corrections for such vortices, since there are not enough grid points to adequately represent the right-hand side of the Poisson equation. For a nonperiodic calculation requiring vortices near boundaries we could calculate the velocities due to those vortices using the direct method, and use the MLC for the remaining vortices.

5.4.1 Infinite Domain

The first boundary conditions we consider are infinite domain conditions, i.e. the boundary conditions corresponding to flow with no physical boundary. We present here two types of methods for computing these: *direct* and *fast* methods.

Direct Method

The direct method of computing the infinite domain boundary conditions is the most straightforward. Simply evaluate the exact velocity at every boundary point due to every vortex, using the velocity expression for the desingularized vortex. If there are N vortices and the numerical domain has edges of M points, then the cost of this method is $O(NM^{d-1})$ for a problem in d space dimensions.

A simple way to speed up the velocity calculation at the boundaries is to evaluate the velocity directly at every other grid point, and interpolate the velocity onto the remaining points using a sufficiently accurate interpolation scheme. However, this calculation is still proportional to the number of vortices; below we present methods which do not rely on the direct $O(NM^{d-1})$ calculation.

Fast Methods

With the direct method we use the infinite domain Green's function for each vortex to define the infinite domain boundary values; for the fast method, we use a different potential theory approach. Consider the computational domain Ω with boundary $\partial\Omega$, on which we want to find infinite domain boundary conditions. The fast method can be summarized as follows: (a) solve the original Poisson equation on a domain Ω^η, $\Omega^\eta = \Omega$ or Ω^η contained in Ω, with homogeneous Dirichlet boundary conditions on $\partial\Omega^\eta$; (b) define a surface S equal to or interior to $\partial\Omega^\eta$ and interpolate the values of the solution onto S if $S \neq \Omega^\eta$; (c) extend the solution harmonically exterior to S; (d) compute the jump in the normal derivative of this extended solution on S, and define this jump as the surface "charge"; and (e) evaluate the field exterior to the surface due to the surface charge.

Note that the boundary conditions for the initial solution of the Poisson equation need not be homogeneous; they may take any value. The key to this method is that the jump in the normal derivative of the solution to the original Poisson equation, extended harmonically, is equal to the surface potential induced by the imposition of Dirichlet boundary conditions which were not the infinite domain boundary conditions. If the original Dirichlet boundary conditions were in fact the correct infinite domain boundary conditions, then the jump in the normal derivative for the harmonic extension of the solution would be zero. The field due to this single-layer potential exterior to S is exactly the field due to the original "charges" (the vortices) in the interior. Thus, knowing just the surface potential we can compute values of the field due to the vortices at any point exterior to this surface.

We present two ways of implementing this method: first, for $\Omega^\eta \neq \Omega$ and $S =$

$\partial\Omega^\eta$, second for $\Omega^\eta = \Omega$ and $S \neq \Omega^\eta$. Let G be the grid in Ω with spacing h. In the first approach:

(1) Create a grid G^η centered in Ω with mesh spacing $h_\eta = \eta h$, η in the range .9 to .95.

(2) Create the right-hand side $\mathbf{g}^{\Omega^\eta} \approx \mathbf{u}^{h_\eta,\epsilon}$ in G^η using the procedure described in Chapter 3.

(3) Solve $\Delta\tilde{\mathbf{u}} = \mathbf{g}^{\Omega^\eta}$ on G^η with $\tilde{\mathbf{u}} = 0$ on ∂G^η.

(4) Extend $\tilde{\mathbf{u}}$ to be zero outside Ω^η, and compute $\rho = \partial\tilde{\mathbf{u}}/\partial n|_{outer} - \partial\tilde{\mathbf{u}}/\partial n|_{inner}$, the jump in the normal derivative, at all points on ∂G^η. Note that the constant function with value zero is the harmonic extension of the solution, and since $\tilde{\mathbf{u}} = 0$ on and exterior to $\partial\Omega$, $\partial\tilde{\mathbf{u}}/\partial n|_{outer} = 0$.

(5) Approximate numerically the integral

$$\mathbf{u}^\infty(\mathbf{x}) = \frac{1}{2\pi}\int_{\partial\Omega^\eta} \rho(\mathbf{x}')G(\mathbf{x}-\mathbf{x}')d\mathbf{x}',$$

where G is the infinite domain Green's function of the Laplacian, $\mathbf{x}$ is on $\partial\Omega$, $\mathbf{x}'$ is on $\partial\Omega^\eta$. The values of $\tilde{\mathbf{u}}^\infty$ are the correct infinite domain boundary conditions on Ω.

This method requires creating an additional right-hand side, solving the Poisson equation on G^η, and numerical integration of $O(4M^{2(d-1)})$ work, where M is the number of grid points per edge. The second implementation does not require creation of an additional right-hand side, but does require additional interpolation. In this approach, which uses only the original grid G, we create $\mathbf{g}^\Omega$ on the original grid once for both the boundary calculation and the final velocity calculation, then

(1) Solve $\Delta\tilde{\mathbf{u}} = \mathbf{g}^\Omega$ on G with $\tilde{\mathbf{u}} = 0$ on ∂G.

(2) Interpolate values of $\tilde{\mathbf{u}}$ from interior points of G onto a surface S (circle in two dimensions, sphere in three dimensions) defined in the interior of Ω.

(3) Use Poisson integrals in two dimensions, spherical harmonics in three dimensions, to extend the field harmonically outside S. Define $\mathbf{F}$ as $\tilde{\mathbf{u}}$ interior to and on S, and as the harmonic extension of $\tilde{\mathbf{u}}$ exterior to S. Define ρ as the jump in the normal derivative of $\mathbf{F}$: $\rho = \partial\mathbf{F}/\partial n|_{outer} - \partial\mathbf{F}/\partial n|_{inner}$ on S.

(4) Approximate numerically the integral

$$\mathbf{u}^{\infty}(\mathbf{x}) = \frac{1}{2\pi}\int_S \rho(\mathbf{x}')G(\mathbf{x}-\mathbf{x}')d\mathbf{x}',$$

where G is the infinite domain Green's function of the Laplacian, $\mathbf{x}$ is on $\partial\Omega$, $\mathbf{x}'$ is on S.

What makes the fast method faster than the direct method is the fact that one substitutes a calculation whose cost is proportional to the number of vortices for a calculation whose cost is proportional to the number of grid points on the boundary. For N much larger than M the savings can be substantial.

5.4.2 Solid Wall (No Flow) on Regular Domain

Solid wall boundary conditions are straightforward to implement; the procedure is as follows.

(1) Compute the infinite domain boundary conditions, $\mathbf{u}^{\infty}$, by any of the methods above.

(2) Solve $\Delta\phi = 0$ with Neumann boundary conditions,

$$\frac{\partial\phi}{\partial n} = -\mathbf{u}^{\infty} \cdot \mathbf{n}$$

where $\partial/\partial n$ is the normal derivative, $\mathbf{n}$ is the unit normal facing in the positive coordinate direction.

(3) Compute $\mathbf{u}^{NF} = \nabla\phi$ on the boundaries, using one-sided derivatives since ϕ is defined only on the interior and the boundary, then define

$$\mathbf{u}^{wall} = \mathbf{u}^{\infty} + \mathbf{u}^{NF}.$$

Note that $\mathbf{u}^{wall}$ satisfies $\mathbf{u}^{wall} \cdot \mathbf{n} = 0$ at each boundary, and the tangential components correctly represent the sum of the vortex-induced velocity and the potential flow.

5.4.3 Fully Periodic

Fully periodic boundary conditions are perhaps the simplest to implement, because we effectively need no boundary conditions, just a wrap-around rule. On a $[0, N]^d$ grid, we keep only N independent values in each direction, and require the values at $i = N$ to equal the values at $i = 0$, and similarly in the other coordinate direction(s). Thus, for the relaxation at every level of the multigrid V-cycle, we use the above substitutions when the boundary values are needed. We also modify the relaxation procedure to relax on the residual equation on the boundary points as well as in the interior, since for periodic boundary conditions we allow vortices in

all bins, including boundary bins.

Local corrections are done with wrap-around as well; for example, a vortex in the (N, j) bin (in two dimensions) is within the correction distance of vortices in bins close to the $(0, j)$ bin as well as vortices in bins close to the (N, j) bin, since bins centered at $(0, j)$ and (N, j) are physically equivalent. Finally, we allow vortices to leave the domain through a boundary and reenter it through the opposing boundary.

5.4.4 Walls with Periodic

Consider now mixed no-flow and periodic boundary conditions. Let there be walls at $x = 0$ and $x = 1$, and let the domain be periodic in the other coordinate direction(s). We consider the three-dimensional case for most generality. Again, the walls induce a potential flow, as does the periodicity, but in this case we need not solve for the potential directly, rather we are able to alter the boundary conditions to incorporate it implicitly. The steps are as follows, for $\mathrm{u} = (u, v, w)$.

(1) Solve $\Delta u = g_u$, where g_u is the component of $\mathbf{g}^\Omega$ in the x-direction, with periodic wrap-around in the y- and z- directions, and $u = 0$ on the x=constant boundaries. This satisfies the no-flow condition at the wall.

(2) The vorticity at the wall must be zero since the flow is inviscid, so substituting $u(0, y, z) = u(1, y, z) = 0$ into the definitions $\omega_x = \partial w/\partial x - \partial u/\partial z$ and $\omega_z = \partial u/\partial y - \partial v/\partial x$, we get $\partial v/\partial x = 0$ and $\partial w/\partial x = 0$ at $x = 0$ and $x = 1$ since $\partial u/\partial y = \partial u/\partial z = 0$ there. Thus, simply solve $\Delta v = g_v$ and $\Delta w = g_w$, with periodic wrap-around in the y- and z- directions and Neumann conditions $\partial v/\partial x = \partial w/\partial x = 0$ at $x = 0$ and $x = 1$.

We treat vortices near the periodic boundaries as described in the previous section; we do not allow vortices near the walls.

5.4.5 Mixed Periodic - Infinite Domain Conditions

While it is straightforward to implement mixed wall-periodic boundary conditions, it is much more difficult to find the correct wall-infinite domain and periodic-infinite domain conditions.

The approach most commonly used to generate fully periodic or semi-periodic boundary conditions is to actually place particles, in this case vortices, outside the original computational domain and include the effect of these particles directly. Often three or fewer additional domains are added. This approach does model the local part of the periodicity, but does not capture the full far-field effect. The physical problem effectively has an infinite number of images, and the sum is not rapidly convergent. Methods using Ewald summation and Fourier series can accurately represent the field from a true periodic distribution of vorticity, but the method as developed by Strain [52, 53] works only for distributions periodic in all space dimensions. Below we present methods for deriving the correct boundary conditions, in two then three spatial dimensions, for a vorticity distribution which is periodic in only one dimension.

Two Dimensions

For mixed periodic-infinite domain boundary conditions in two dimensions we use a slightly different approach than that used for the fast method in Section 5.4.1. In the fast method for infinite domain boundary conditions, we found the single-

layer potential on a surface surrounding the vortices which was used to represent the field exterior to the surface due to the vortices. We used the harmonic extension of the original solution to calculate this surface potential. Here, we also extend the original solution harmonically, but rather than computing a line charge as a jump in the normal derivative of the extended solution, we use matching conditions to find the correct harmonic functions so that there is no jump in the normal derivative. The sum of the original function and these harmonic functions is then the correct solution of the original equation with infinite domain boundary conditions. This is a variation of the method presented in [3].

Consider the domain $\Omega = [0,1]^2$; we seek the solution to $\Delta u = f$ with periodic boundary conditions in the y-direction and infinite domain boundary conditions in the x-direction. For the MLC, u and f would be $\tilde{\mathrm{u}}$ and $\mathbf{g}^{\Omega}$, respectively.

In a two dimensional domain, given values of a function Φ on a single line $x = a$ which is periodic in y, we can define harmonic functions for $x > a$ and $x < a$ such that the values of the harmonic functions on $x = a$ are equal to the values of Φ on $x = a$:

$$\phi^R(x,y) = \sum_{k=-\infty}^{\infty} \hat{\Phi}_k e^{2\pi i k y} e^{-2\pi|k|(x-a)}$$

for $x \geq a$ and

$$\phi^L(x,y) = \sum_{k=-\infty}^{\infty} \hat{\Phi}_k e^{2\pi i k y} e^{2\pi|k|(x-a)}$$

for $x \leq a$, where

$$\hat{\Phi}_k = \int_0^1 \Phi(y) e^{-2\pi i k y} dy.$$

Computationally we work with finite sums, so we redefine the limits in the above

sums as $k = -M$ to $k = M$. To capture M Fourier modes we need $J = 2M + 1$ physical points.

The procedure for finding the boundary conditions is as follows. Given a right-hand side f, we first solve $\Delta u = f$ with homogeneous Dirichlet boundary conditions in the x-direction and periodic boundary conditions in the y-direction. We then interpolate u to define $g_j^{0+} = u(\Delta x, j\Delta y)$, $g_j^{1-} = u(1 - \Delta x, j\Delta y)$, Δx a small parameter, $\Delta y = 1/J$.

Next, we construct four functions as follows:

$$\phi_0^R(x,y) = \sum_{k=-M}^{M} \hat{\phi}_k^{R0} e^{2\pi iky} e^{-2\pi|k|x} \quad \text{for } x \geq 0$$

$$\phi_0^L(x,y) = \sum_{k=-M}^{M} \hat{\phi}_k^{L0} e^{2\pi iky} e^{2\pi|k|x} \quad \text{for } x \leq 0;$$

$$\phi_1^R(x,y) = \sum_{k=-M}^{M} \hat{\phi}_k^{R1} e^{2\pi iky} e^{-2\pi|k|(x-1)} \quad \text{for } x \geq 1;$$

$$\phi_1^L(x,y) = \sum_{k=-M}^{M} \hat{\phi}_k^{L1} e^{2\pi iky} e^{2\pi|k|(x-1)} \quad \text{for } x \leq 1,$$

with $\hat{\phi}_k^{R0}, \hat{\phi}_k^{L0}, \hat{\phi}_k^{R1}$ and $\hat{\phi}_k^{L1}$ to be defined by the matching conditions. Define

$$\hat{g}_k^{0+} = \sum_{j=1}^{J} g_j^{0+} e^{2\pi ikj\Delta y},$$

$$\hat{g}_k^{1-} = \sum_{j=1}^{J} g_j^{1-} e^{2\pi ikj\Delta y},$$

$$\hat{u}_k(x) = \sum_{j=1}^{J} u(x, j\Delta y) e^{2\pi ikj\Delta y},$$

Now define $F(x,y) =$

$$\sum_{k=-M}^{M} F_k(x)e^{2\pi iky} = e^{2\pi iky}\begin{cases} \phi_k^{L0}e^{2\pi|k|x} + \phi_k^{L1}e^{2\pi|k|(x-1)} & \text{for } x \le 0 \\ \phi_k^{R0}e^{-2\pi|k|x} + \phi_k^{L1}e^{2\pi|k|(x-1)} + \hat{u}_k(x) & \text{for } 0 < x < 1 \\ \phi_k^{R0}e^{-2\pi|k|x} + \phi_k^{R1}e^{-2\pi|k|(x-1)} & \text{for } x \ge 1 \end{cases}.$$

The matching conditions for infinite domain boundary conditions require that the function F and its normal derivative be continuous at the boundaries. We can impose these matching conditions on the Fourier coefficients $F_k(x)$ independently since all functions have the same y-dependence. The conditions are as follows (recalling that u and therefore $\hat{u}_k$ are zero at $x = 0$ and at $x = 1$):

i) continuity at $x = 0$

$$\hat{\phi}_k^{L0} = \hat{\phi}_k^{R0};$$

ii) continuity at $x = 1$

$$\hat{\phi}_k^{R1} = \hat{\phi}_k^{L1};$$

iii) continuity in normal derivative at $x = 0$

$$2\pi|k|\hat{\phi}_k^{L0} = \frac{\partial u_k}{\partial x}|_{x=0} - 2\pi|k|\hat{\phi}_k^{R0};$$

iv) continuity in normal derivative at $x = 1$

$$-2\pi|k|\hat{\phi}_k^{R0} = \frac{\partial u_k}{\partial x}|_{x=1} + 2\pi|k|\hat{\phi}_k^{L1}.$$

Solving this system of four equations in four unknowns and defining the derivatives

as

$$\frac{\partial u_k}{\partial x}|_{x=0} = \frac{g_k^{0+} - 0}{\Delta x}, \quad \frac{\partial u_k}{\partial x}|_{x=1} = \frac{0 - g_k^{1-}}{\Delta x},$$

we find

$$\hat{\phi}_k^{L0} = \hat{\phi}_k^{R0} = \frac{1}{4\pi|k|} \frac{\hat{g}_k^{0+}}{\Delta x},$$

$$\hat{\phi}_k^{L1} = \hat{\phi}_k^{R1} = \frac{1}{4\pi|k|} \frac{\hat{g}_k^{1-}}{\Delta x},$$

At $x = 0$ and at $x = 1$, the correct boundary conditions are

$$u^{mixed}(0, y) = \sum_{k=-M}^{M} (\hat{\phi}_k^{R0} + \hat{\phi}_k^{L1} e^{-2\pi|k|}) e^{2\pi i k y},$$

$$u^{mixed}(1, y) = \sum_{k=-M}^{M} (\hat{\phi}_k^{R0} e^{-2\pi|k|} + \hat{\phi}_k^{L1}) e^{2\pi i k y},$$

and the final solution is

$$u^{mixed}(x, y) = \sum_{k=-M}^{M} (\hat{\phi}_k^{R0} e^{-2\pi|k|x} + \hat{\phi}_k^{L1} e^{-2\pi|k|(1-x)} + \hat{u}_k(x)) e^{2\pi i k y},$$

for $0 \leq x \leq 1$.

Three Dimensions

Consider the equation $\Delta^h u = g$ in three dimensions on a domain Ω with boundary $\partial\Omega$. We use an extension of the fast methods presented in Section 5.4.1 to find boundary conditions on $\partial\Omega$ which are periodic in the z-direction, infinite domain in the x- and y- directions. Here, we perform the same initial steps (a)-(d) as in that method, to the point of finding the charges on a surface S in the interior of Ω as the jump in the normal derivative, with the exception that we solve the Poisson equation on Ω^n using boundary conditions which are periodic in the z-direction and

homogeneous Dirichlet on the other boundaries, rather than homogeneous Dirichlet on all boundaries. Again we can use either of two approaches: define Ω^η interior to Ω and let $S = \partial\Omega^\eta$, or define $\Omega^\eta = \Omega$ and let S be a right circular cylinder interior to Ω.

Once we know the surface potential ρ on S we want to find the field due to a periodic extension of ρ in the z-direction. For simplicity, first consider a charge ρ at $\mathbf{x}_0$ only, where $\mathbf{x}_0 = (x_0, y_0, z_0)$ lies on S. The field due to this charge and all of its periodic images in the z-direction is the infinite sum,

$$u(x, y, z) = \frac{\rho}{4\pi} \sum_{j=1}^{\infty} 1/r_{3_j},$$

where $r_{3_j} = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0 - j)^2}$.

This function $u(x, y, z)$ is periodic in z; its Fourier transform in z is

$$\hat{u}_k(x, y) = \frac{\rho}{4\pi} \int_0^1 u(x, y, z) e^{-ikz} dz = K_0(r_2 k),$$

where $K_0(r_2 k)$ is the modified Bessel function of order 0 and parameter r_2, $r_2 = \sqrt{(x - x_0)^2 + (y - y_0)^2}$. Then the field due to the periodic charges can be approximated by the finite sum

$$u(x, y, z) = \sum_{k=-J}^{J} \hat{u}(x, y) e^{ikz} = \sum_{k=-J}^{J} K_0(r_2 k) e^{ikz},$$

where J is the number of Fourier modes, $M_z = 2J + 1$ is the number of grid points on $\partial\Omega$ in the z-direction. Thus, for a line of M_z sources of strength ρ_j at $\mathbf{x}_j = (x_0, y_0, z_0 + j\Delta z)$, $j = 1, ..., M_z$, on S, the velocity induced by those sources and all of their periodic images can be expressed in terms of M_z sums of modified

Bessel functions:

$$u(x,y,z) = \frac{1}{4\pi} \sum_{n=1}^{M_z} \rho_n \sum_{k=-J}^{J} K_0(r_{2_n} k) e^{ikz},$$

with $r_{2_n} = \sqrt{(x-x_n)^2 + (y-y_n)^2}$. Thus, rather than numerically calculating the convolution of the charges on S with the infinite domain Green's function, we numerically calculate the convolution of the charges on S with this periodic Green's function, the sum of modified Bessel functions.

Because the location of all of the points on $\partial\Omega^n$ and S are constant for all time, these Bessel functions need only be evaluated once at the beginning of a calculation and then stored, rather than evaluated at every time step. Also note that by reflection symmetry in the $x-y$ plane, we need only do $(\frac{1}{4}M^2)(4M^2)$ evaluations rather than the full $(4M^2)^2$ evaluations, where $M = M_x = M_y = M_z$, i.e. M is the number of grid points in each coordinate direction.

5.4.6 Mixed Wall-Infinite Domain Conditions

Mixed wall-infinite domain boundary conditions are a special case of periodic-infinite domain boundary conditions. Consider a physical domain $[0,1] \times [0,1]$ in two dimensions; we want to find boundary conditions which are no-flow in the y-direction and infinite domain in the x-direction. Let the domain contain vortices ω_i, $i = 1, ..., N$ located at $\mathbf{x}_i$. Define new vortices ω_j in the domain $[0,1] \times [1,2]$, such that $\omega_j = -\omega_{j-N}$, $(x_j, y_j) = (x_{j-N}, 2 - y_{j-N})$, $j = N+1, ...2N$ Define $\Omega = [0,1] \times [0,2]$, and compute periodic-infinite domain boundary conditions on Ω using techniques from the previous section. The boundary conditions on $[0,1] \times [0,1]$ are now infinite domain in the x-direction and no-flow in the y-direction. Once the

boundary conditions are found, the rest of the velocity calculation may be completed considering only the original vortices. The extension to three dimensions is straightforward.

CHAPTER 6

Three Dimensional Results : Stability of a Vortex Ring

6.1 Vortex Rings

The vortex ring has been studied analytically, experimentally, and numerically. Analytically, linear stability theories indicate that a thin vortex ring is unstable to certain azimuthal bending waves around its perimeter [56, 57, 58, 51]. Experimental observations support this prediction [44, 45, 46, 42, 55]. The most complete numerical study of this problem using vortex methods is by Knio and Ghoniem [39], and we directed our initial studies for comparison with theirs. Knio and Ghoniem study a ring using the direct $O(N^2)$ vortex method for vortex segments with third-order exponential core functions. They do a study of the thin tube approximation, representing the vortex ring as a single filament of finite radius, as well as discretize the ring with multiple filaments.

Consider a vortex ring of radius R_0, symmetric about the z-axis. The intersection of this torus with the y-z plane is two disks of radius σ, centered at $x = \pm R_0$ (see Figure 6.1a). The ring is discretized by first placing points at different radial stations within the cross-section centered at $x = R_0$; the positions of these points for the different discretizations are shown in Figure 6.1b. (The meshes are labeled in Figure 6.1b according to the labeling in [39]; Mesh KG:II and KG:III are, respectively,

Meshes II and III in [39].) Then a circular filament whose center lies on the z-axis is passed through each of these points, and each filament is discretized using a finite number of segments. We define our computational domain as $[0,1]^3$; in all our calculations $R_0 = 0.1$ and $\sigma/R_0 = 0.275$. The base grid in all calculations using MLC is $8 \times 8 \times 8$, and $C = D = 1.5$, unless otherwise noted. The other parameters for all the calculations are shown in Table 6.1.

We examine the time evolution of perturbed vortex rings. The unperturbed core has an exponential distribution of vorticity, $\Omega(r)$. In all calculations the perturbation is imposed as a displacement of the vortex segment endpoints in the radial direction (radial from the z-axis, not the center of the cross-section), sinusoidal in the azimuthal angle with wavenumber n. The amplitude of the perturbation is $.02R_0$, and is coplanar with the original unperturbed filament.

Rather than the core function presented in Chapter 1 we use the core shape function proposed by Leonard [41] and shown to be second-order by Beale and Majda [10], which was used in [39]:

$$f_\delta(r) = \frac{3}{4\pi\delta^3} e^{-(r/\delta)^3},$$

where δ is the core radius of the segment. We again require that the integral of the core function over the region of its support be unity; this accounts for the $\frac{3}{4\pi\delta^3}$ seen in the expression. The velocity field $\mathbf{u} = \mathbf{K}_\delta * \omega$ at $\mathbf{x}$ due to a single vortex segment with center at $\mathbf{x}_0$, circulation Γ and length $\ell = (\ell_x, \ell_y, \ell_z)$ is then

$$u = \Gamma \frac{1 - e^{-(r/\delta)^3}}{4\pi r^3} (\ell_y(z - z_0) - \ell_z(y - y_0))$$
$$v = \Gamma \frac{1 - e^{-(r/\delta)^3}}{4\pi r^3} (\ell_z(x - x_0) - \ell_x(z - z_0))$$

$$w = \Gamma \frac{1 - e^{-(r/\delta)^3}}{4\pi r^3}(\ell_x(y - y_0) - \ell_y(x - x_0))$$

In order to compare our results directly with [39], we define the remaining parameters appropriately, so that the dimensionless quantities σ/R_0, $\Delta r/R_0$ (for the first calculations presented), ε/R_0, and $\Omega(r)\Delta t$ are the same in our calculations as in [39]. In fact, we use the same values of Δt ($\Delta t = 0.1$ for $N \approx 2000$ calculations) and $\Omega(r)$, so that the results can be more easily compared. The initial analytic vorticity is

$$\Omega(r) = \frac{e^{-(r/\sigma)^3}}{\sigma^2 a},$$

where $a = (1600\pi/3)\gamma(2/3) \approx 2268.85$; the total circulation in [39] is 2 and in our calculations is 2/1600, since the circulation is proportional to the cross-sectional area, and $R_0 = 4, \sigma = 1.1$ in [39] while in our calculations $R_0 = .1, \sigma = 0.0275$.

In setting the circulation of the individual filaments, we first sought to compare directly with [39], so we followed their approach. The circulation of each filament was found by solving the linear system of equations,

$$\omega(\mathbf{x}, 0) = \sum_{i=1}^{N} \Gamma_i \ell_i f_\delta(\mathbf{x} - \mathbf{x}_i(0)) \tag{6.1}$$

at each vortex midpoint $\mathbf{x} = \mathbf{x}_j, j = 1, ..., N$. This discretization will be discussed further in the next section.

The dynamics of a steady unperturbed vortex ring are relatively simple. The ring translates along the z-axis (in the positive z-direction for $\Omega(r) > 0$) with a self-induced velocity proportional to the circulation of the ring. In addition, the core rotates about its center; a filament initially at $\phi = 0$ with total length $2\pi(R_0 + r)$,

is at some later time at $\phi = \pi$ with length $2\pi(R_0 - r)$; still later it will return to its original position and length. Similarly, all filaments move about the center of the core, but maintain their relative radial positions. A vortex ring translating at constant velocity and with a constant shape is a steady solution of the Euler equations.

To accommodate the translation of the ring numerically, at every time step in our calculations we calculated the average value of the z-coordinates of all vortex segment midpoints, then subtracted this value from the z-coordinate of every endpoint and midpoint. Thus the vortex ring stayed centered in the $[0,1]^3$ domain. This meant that the ring did not translate relative to the computational mesh; in future we will instead recenter the ring once it has moved a full mesh spacing.

When the ring is perturbed in the radial direction with a sinusoidal perturbation of a stable wave number, the perturbation moves around the ring but does not grow in amplitude. The frequency of the rotation of the perturbation around the z-axis is low at small wavenumber n, grows to a maximum, then decreases again as n approaches the neutrally stable wavenumber, n_n. These modes $n \leq n_n$ are linearly stable. At $n = n_n$ the perturbation neither grows nor rotates. When a perturbation of an unstable wave number is imposed, the perturbation is a standing wave and grows in amplitude.

6.2 Presentation of Data

In this chapter we present the data in four different formats. The first is three-dimensional pictures of the vortex filaments. While these give a good sense of the

overall dynamics, they are not quantitatively useful.

The second format is histogram plots of the amplitudes of the Fourier modes in θ of the radial displacements of the centers of the vortex segments. The histograms represent the unweighted averages of the amplitudes of the modes over all filaments. Initially the vortex segments were evenly spaced in the θ-direction; at later times the segments have moved and some have split. After vortex splitting had occurred, we measured only the radial displacement of the centers of the original segments, trying to maintain data points as evenly spaced as possible in the θ-direction. The amplitude of the k^{th} mode was defined as:

$$a_k = \frac{1}{N_{fil}} \sum_{i_{fil}=1}^{N_{fil}} \mathrm{Re}\,(\frac{2}{N_{seg}} \sum_{j=firstseg(i_{fil})}^{lastseg(i_{fil})} \frac{\rho_j}{R_0} e^{2\pi ijk/N_{seg}}),$$

where ρ_j is the distance of the center of segment j from the z-axis, N_{fil} is the number of filaments, and N_{seg} is the original number of segments per filament. We see in the long time calculations, using MLC or the direct method, that this method of representing the data generates misleading plots after repeated vortex splitting has occurred. The late-time histograms show all other modes being introduced at close to the same amplitude as the $n = 12$ mode, yet in the three-dimensional views a 12-fold symmetry is still present. These additional modes are seen at the same amplitudes when the transform is taken using segment bottoms rather than centers. Looking more closely at the histogram data, we see that the unweighted averaging of the data over the filaments cloaks the symmetry that is in fact present in most of the filaments. In the $N = 2280$ calculation, for example, 13 of the 19 filaments exhibit only the $n = 12$ mode, yet the remaining 6 filaments, all at the outer radius, exhibit all modes. By taking the unweighted average we lose this information. The presence

of the additional "noise" correlates with the stretching undergone by each filament: the 13 filaments with the $n = 12$ mode only all have fewer than 270 segments per filament at $t = 120$, the other 6 filaments all have more than 320 filaments, some as many as 480 segments per filament. Each filament began with 120 segments.

Thus we observe that although these histograms represent useful information for early times, after repeated vortex splitting they oversimplify the results, obscuring the significant features. A more sophisticated analysis of this data is required.

In order to follow the time evolution more easily, we also present plots of amplitude vs. time for the single most unstable mode in the calculations. These plots are based on the same data as the histogram plots.

The fourth type of plot is of the intersection of the vortex filaments with the $\theta = 0$ plane. Although this does not represent the full dynamics of the ring—for that one would need many cross-sections—it does give an idea of the distortion of the originally smooth Lagrangian mesh. In these plots the vertices of each polygon are the filaments, and filaments which began at the same radial station are connected by line segments. Note that these lines are not the vortex segments.

6.3 Summary of Results

In this section we present a summary of overall results from our calculations; in the next section we give the details and data from the individual calculations.

(1) The MLC, with and without AMR, is accurate enough to reproduce stability results for a perturbed vortex ring found using the direct method. Figures 6.2 through 6.7 show data from studies repeating the calculations of Knio and Ghoniem

using the MLC instead of the direct method.

(2) The $N \approx 2000$ calculations do agree with linear stability analysis, but as we see from the plots of the cross-sections of the core, the original Lagrangian mesh becomes very tangled and crosses over itself at finite time. This occurs for $N \approx 14000$ calculations as well, but in the more refined calculations we see the development of additional features of the core distortion which were not seen for $N \approx 2000$. Thus we capture features of the core distortion at $N \approx 14000$ which are not seen for $N \approx 2000$, but the phenomena are not yet fully resolved. Figures 6.7, 6.15 and 6.18 show these results.

(3) At the higher resolution ($N \approx 14000$) the MLC excites the $n = 4$ mode, but this mode and its multiples do not detract from the overall stability results; the amplitudes of these modes grow from round-off error to a finite percentage of the $n = 12$ amplitude, but seem to level off there, and do not interfere with the growth or oscillation of the amplitude of the originally perturbed mode. In the histogram plots we see all multiples of the $n = 4$ mode; this results from mode-mode coupling of the $n = 4$ and $n = 12$ modes. It is the MLC itself, rather than the addition of AMR, which induces these modes. They appear only for the higher resolution, and are not strongly affected by changing $C = D$ from 1.5 to 2.5. Figure 6.15 shows that these additional modes do not detract from the stability results; Figure 6.10 shows that the modes are not due to AMR, and Figure 6.11 shows that they are not due to insufficiently resolved boundary conditions.

(4) Using a later discretization, we see better agreement between the $N \approx 2000$ and the $N \approx 14000$ calculations, and these differ from the earlier calculations in that they show an initial decline in amplitude rather than growth at all time. We

suggest that this may be due to the fact the initial $N \approx 2000$ calculations studied a ring with a different vorticity distribution, due to the method used in [39] to assign the circulation of the filaments.

We also mention in the next section the attempts we made to verify the accuracy of our calculations. At several points we checked the time step to ascertain that it was in fact small enough so that further reduction did not change the results. In addition we show the results from a calculation (Figure 6.12) using a correction radius on a fine grid which was too small relative to the effective radius of the core function; we see in all other calculations that this behavior has been eliminated.

6.4 Individual Calculations

Our goal in the first numerical experiments was to show that the method of local corrections, with or without adaptive mesh refinement, is accurate enough to reproduce the stability results found using the direct method. Figures 6.2 through 6.6 show results from calculations of rings with $N = 2040$, Mesh KG:II, $N_c = 17$ discretization, and circulation calculated by solving the linear system of equations (6.1). Knio and Ghoniem present contrasting results for rings perturbed with $n = 9$ and $n = 12$; $n = 9$ is a stable wave number for a ring of these dimensions, $n = 12$ is the most unstable wave number.

Figure 6.2 shows the vortex ring perturbed with $n = 9$ at times $t = 10, 40, 70, 100$, the same times shown in [39]. This calculation was done using MLC on a uniform $8 \times 8 \times 8$ grid; since there is no increase in the number of vortices and the geometry remains stable, there was no need in this case for AMR. These results look identical

to the results in [39]. We see from the pictures in Figure 6.2 and the time profiles of the amplitude of the modes (Figure 6.3) that the ring perturbed with $n = 9$ is stable; the amplitude of the perturbation decreases rapidly at first, then rises and falls again in an oscillatory pattern. We see also from Figure 6.3a that an initial perturbation of $n = 9$ triggers the first harmonic, $n = 18$, although at a much smaller amplitude. This is again what is seen in [39], and agrees with the linear stability theory.

Figure 6.4 shows the ring with an initial $n = 12$ perturbation at times $t = 30, 60, 90, 120$, also the times chosen in [39]. Here we see that the initial perturbation of the same magnitude as the perturbation in the $n = 9$ calculation grows steadily in amplitude. This distortion of the core increases the number of vortex elements needed; recall that a vortex segment is automatically split into two segments whenever its length exceeds a critical length, which is defined as twice the largest initial length. As the number of vortex elements increases, the optimal level of grid refinement changes, and although the calculation starts on a uniform 8^3 grid, when the number of vortices exceeds 4000 the grid is refined one level.

In Figure 6.5 we show three different views of the calculation at $t = 140$, again for comparison with [39]. Since the calculations in [39] were done with the direct method, they were stopped after $t = 140$, since the number of vortices had by then grown from $N = 2040$ to $N = 6936$. At $t = 140$ we had 6352 vortices, but qualitatively there is very little difference seen between our results and those in [39]. We do observe in Figure 6.6a the onset of all modes at late time; these additional modes are also found in the direct calculation (see Figure 6.13); We see from Figure 6.13 and the discussion in the previous section that these additional modes are

associated with the vortex splitting rather than any grid-induced distortion. In Figure 6.6b we see the growth of the $n = 12$ amplitude only as a function of time. Note the difference in scale between Figure 6.6b($n = 12$) and Figure 6.3b($n = 9$).

In order to check the sensitivity to time step, we also ran the $n = 12$ calculation with a time step of $\Delta t = 0.4$, four times larger than in the previous calculation. All other parameters were kept the same. The results were qualitatively indistinguishable; the only numerical difference observed was that at $t = 140$ there were 6352 vortices for $\Delta t = 0.1$, and 6380 vortices for $\Delta t = 0.4$.

In Figure 6.7 we show the intersection of the 17 filaments with the plane at $\theta = 0$ for the $n = 12$ calculation for times $t = 0$ to $t = 80$. These plots show that by $t = 80$ the original Lagrangian mesh has crossed over itself, and the accuracy of this representation is questionable after that time.

Based on these observations, we refined the initial discretization to explore the effects of higher resolution on the results of the calculation. Again we sought to compare our results to those in [39], but before refining the mesh we switched to an alternate discretization (Mesh KG:III) than that used in the previous calculations. Rather than having equal numbers of filaments at each radial station in the core, we let the number of filaments increase linearly with the radius r from the center of the core, so that the intersegment spacing was more nearly constant in the cross-section. This was found by Ghoniem and Knio to yield the most accurate discretization. Again we found the circulation of the filaments by solving the system of linear equations (6.1) at the vortex locations. In this case we had $\Delta r = .1080R_0$, $\delta = .155R_0$, the values from [39].

To test the effect of the different discretizations at $N \approx 2000$, we first ran the

calculation for $N = 2280$ (Mesh KG:III, $N_c = 19$) to $t = 40$, with the $n = 12$ perturbation of the same magnitude as before, The results are indistinguishable from those found in the earlier $N = 2040$ calculation; Knio and Ghoniem had also found this.

Having established similar results with $N \approx 2000$ using Mesh KG:III, we then increased the number of radial positions of the filaments from two to four, increasing the number of filaments from 19 to 61 (Mesh KG:III, $N_c = 61$). For this $N = 61 \times 120 = 7320$ calculation, we first used a base grid of $8 \times 8 \times 8$ with two levels of refinement. Again we computed circulation of the filaments by solving the system of equations (6.1). This calculation, as for all later calculations with four radial stations, used $\Delta t = .05$, half the time step used for the calculations with two radial stations. The fact that $\Delta t = 0.4$ was sufficiently small for the $N \approx 2000$ calculation might indicate that $\Delta t = 0.1$ should be sufficient for the more refined calculations, but this was not observed. An early calculation was run with $\Delta t = 0.1$ for $N = 15600$, and it was not consistent with the same calculation run with $\Delta t = 0.05$. Thus all later calculations used the smaller time step.

Figure 6.8 shows a picture of the 17 filaments ($N = 2040$) and the 61 filaments ($N = 7320$) at $t = 40$, in order to give an idea of the increased resolution of the refined discretization. Figure 6.9a shows the time profile of the amplitude of the $n = 12$ mode; contrast this to Figure 6.6b. We see a marked difference in behavior: in Figure 6.6b there is steady growth, while in Figure 6.9a, for the higher resolution, we see oscillatory behavior for $t \leq 40$ and no overall increase in amplitude.

In order to understand the difference in behavior between the $N \approx 2000$ and $N = 7320$ results, we performed the $N = 7320$ calculation to $t = 40$ using the

direct $O(N^2)$ method; these results are shown in Figure 6.9b. Both curves exhibit a negative slope for $10 \leq t \leq 20$, then the slope becomes positive; since these calculations were not run to later time it was unclear whether the amplitude would continue to grow or would level off. This is in contrast to the $N = 2040$ calculations.

Knio and Ghoniem in fact observed this effect in their calculations, as can be seen in Figure 14 in [39], where they plot the log of the amplitudes of the $n = 12$ mode vs. time for different initial discretizations. The only result they present from a calculation with four radial stations is for 33 filaments, and they also observe oscillatory behavior rather than growth of this mode. This is in contrast to the results with fewer radial stations, but no further discussion in [39] pursues the possible inference that the coarser calculations are underresolved.

Figure 6.10 shows the difference between calculations using the MLC and those using the direct method. Figure 6.10a shows the amplitudes for the calculations with $N = 7320$ using a $8 - 32$ grid; we see here the $n = 4$ mode and its multiples. Figure 6.10b shows calculations done with the direct method and we see no growth in these other modes; we conclude that it is the fourfold symmetry of the grid in the MLC which induces these modes. Since these modes were not seen in the $N \approx 2000$ calculations on a uniform 8^3 grid, we next sought to determine whether it is the MLC itself or the addition of AMR which produces the $n = 4$ disturbance. Figure 6.10c shows the results of a calculation done on a uniform 32^3 grid for short time. In this figure we see already the introduction of these additional modes, and conclude that it is the presence of the grid in MLC which produces these modes, rather than the addition of AMR to MLC.

We next investigated whether the additional modes were induced by insufficiently

resolved boundary conditions. Figure 6.11 shows calculations for $n = 12$, $N = 65 \times 240 = 15600$ (Mesh KG:II, $N_c = 65$), with circulation computed for each filament as the product of the point value of the vorticity at the location of the filament and the area represented by that filament,

$$\Gamma_i|_{r_0} = 2\pi r_0 \Delta r \Omega(r_0). \tag{6.2}$$

Calculations were done with MLC on grids of 8-32 (Figure 6.11a), 16-32 (Figure 6.11b) and with the direct method (Figure 6.11c); all results are for time $t = 10$. We see again that is the MLC which induces the additional modes, but there is no detectable difference between Figures 6.11a and 6.11b, which indicates the boundary conditions are sufficiently well resolved. We see from all three calculations that the amplitude of the $n = 12$ mode, initially at .02, has decreased markedly.

We present in Figure 6.12a the results of an early run done with $N = 15600$ as above, using a grid of $8-64$, which was the grid indicated by timing results done with $C = D = 1.5$ to give the optimal speed of the algorithm. This choice of correction radius and grid was appropriate for the core function presented in Chapter 1, since the kernel $\mathbf{K}_\delta(r) = \mathbf{K}(r)$ for $r \geq \delta$. The exponential core function we use in this chapter, however, effectively extends out to $\approx 3\delta$, since $\mathbf{K}_\delta$ differs visibly from $\mathbf{K}$ out to that point. Thus for calculations using this core function the physical radius of local corrections must be extended to accommodate the larger effective core radius, either by increasing C or using a coarser grid. In Figure 6.12b we present the results from the same calculation using a grid of $8 - 32$; here the radius of local corrections is large enough relative to the effective core radius to obtain good accuracy.

In the calculations presented above we have used discretizations used by Knio

and Ghoniem in [39]. However, those discretizations are not consistent with standard convergence studies, which allow the core radius δ to decrease as the intersegment spacing Δr decreases; as they refine from two to four radial stations Δr decreases from $.1080R_0$ to $.0705R_0$, yet δ decreases only from $.155R_0$ to $.150R_0$. The reason they must keep δ so large is that solving the system of linear equations (6.1) to determine the circulation of each filament is reasonable for $\delta >> \Delta r$, and once the intervortex spacing is specified, the value of δ is entirely determined in [39] by matching the numerical circulation (the sum of the circulations of the filaments) with the analytically specified circulation (the integral of $\Omega(r)$ over space).

There is another subtlety hidden in this discretization. The size of the ring is defined by σ, the scaling in the exponential function $\Omega(r)$ which defines the vorticity in the cross-section. The analytic circulation of the ring in [39] is 2, but this value is equal to the integral of $\Omega(r)$ over all area from $r = 0$ to $r = \infty$; the integral of $\Omega(r)$ over the area enclosed by $r = \sigma$ has the value 1.55. For the $N \approx 2000$ calculations in [39], the outer segments are at $2\Delta r = .218R_0$. Assuming that each radial station at r carries the vorticity from $r - \Delta r/2$ to $r + \Delta r/2$, these filaments should only represent the core out to $r = .2725R_0$, and thus carry a total circulation of approximately 1.55. However, the method of assigning circulation by solving the system of linear equations, and choosing δ so that the total numerical circulation is 2, means that the vorticity is more concentrated than is indicated by $\Omega(r)$. Thus we would expect the ring to be more unstable.

The final three calculations we present are of rings discretized in a way that permits a more customary convergence study. Figures 6.13 and 6.14 show the results from a ring discretized with two radial stations, $\Delta r = .150R_0$, $\delta = .200R_0$,

and a total circulation of 1.92/1600. We use a larger Δr here to capture 95% of the circulation without having to assign the vorticity at $r > 2.5\Delta r$ to any of the filaments. The circulation of each filament is found as the integral of vorticity over the area represented by each filament,

$$\Gamma_i|_{r_0} = 2\pi \int_{r=r_0-.5\Delta r}^{r=r_0+.5\Delta r} \Omega(r) r dr. \tag{6.3}$$

Figure 6.13a shows the amplitudes of the modes for this calculation using MLC on a uniform $8 \times 8 \times 8$ grid and using the direct method. In this figure we see only one curve; this is because the data from the calculation using MLC is indistinguishable from the data found using the direct method. Figure 6.13b shows the time evolution of the $n = 12$ mode for both calculations from $t = 0$ to $t = 140$; again we see only one curve because the data are virtually identical. However, these calculations do differ from the earlier $N \approx 2000$ calculations in that they now capture the initial decline in amplitude. We suggest this may be due to our discretization (6.3), since we have not required that the full circulation of the ring be concentrated in too small a core. Figure 6.14 shows the intersection of the 19 filaments with the $\theta = 0$ plane for the MLC calculation. Notice the increased resolution of the movement of the outer filaments relative to Figure 6.7; for these calculations we have 12 rather than 8 outer filaments.

Figures 6.15 and 6.16 show the results from a calculation done with $N = 61 \times 240 = 14640$; the discretization of the previous experiment was refined by a factor of two in each coordinate direction. In this discretization there were four radial stations, twice as many filaments as previously at each radius, and 240 segments per filament. In this calculation we also increased the correction and spreading

distances used in the MLC with AMR to be consistent with the effective radius of the core function. All previous calculations had $C = D = 1.5$, here we used $C = D = 2.5$.

Figure 6.15a shows the modes of the previous $N = 2280$ calculation using MLC and the present $N = 14640$ calculation using MLC with AMR at times $t = 0$ through $t = 70$. Figure 6.15b shows the time evolution of the $n = 12$ mode for these two calculations from $t = 0$ to $t = 70$. We see in Figure 6.15a that the $n = 4$ modes and its multiples are still present, though at smaller amplitude than than in Figure 6.10a. The modes that are not multiples of 4 remain at machine precision until $t = 60$, as can be seen by a log plot of the same data (not shown). The onset of the new modes which are not multiples of $n = 4$ occurs at the same time that the creation of new vortex segments becomes significant. We see the new modes first at $t = 60$ in Figure 6.15a; from $t = 0$ to $t = 40$ only 12 segments were created, from $t = 40$ to $t = 50$ 220 segments were created, and from $t = 50$ to $t = 60$ 1022 new vortices were created. See the discussion in the previous section about these modes.

We checked the convergence of the $N = 14640$ calculation here by redoing the calculation between $t = 45$ and $t = 55$ using a time step of $\Delta t = .025$, half of the previous time step. The results were indistinguishable from those with the full time step, indicating that the time step we had been using was sufficiently small.

In Figure 6.15 we see the results from the calculations with $N = 14640$ and $N = 2280$. In both cases the amplitude of the perturbation decreases from its initial value, and then at later time ($t = 30$ in the coarse calculation, $t = 20$ in the finer calculation) the amplitude begins to rise. We see, however, that the finer calculation shows a greater decline; this is indicative that further refinement studies are required.

For the $N = 14640$ calculation we see the onset of the additional modes much earlier than for the $N = 2280$ calculation. This is consistent with our previous explanation of these modes; by $t = 70$ the $N = 2280$ calculation has added only 72 new vortex segments because of stretching, the $N = 14640$ calculation has added 2175 new segments, indicating its greater sensitivity to the stretching of the ring.

Figure 6.16 shows the intersection of the 61 filaments for this calculation with the $\theta = 0$ plane. Contrast this with Figure 6.14 to see the increase in resolution of the deformation of the core for $t = 0$ to $t = 70$. We see that the original Lagrangian mesh crosses over itself at finite time, even at this higher resolution. We also see, however, how much better resolved the calculation is. Note the fine development of the outer "arm" which begins at $t \approx 30$ and the beginning of the second outer arm at $t = 50$; the $N \approx 2000$ calculations miss the first arm completely and resolve the second arm with much less definition. We can see clearly at times $t = 50$ and $t = 55$ that the filaments at different radial stations are each creating arms, yet are out of phase with each other in ϕ. Thus we see again the need for the higher resolution of the radial modes; there is no indication yet that we have resolved correctly the full distortion of the core.

6.5 Discussion and Conclusions

From the results presented in the previous section we conclude that:

(1) The MLC, with and without AMR, is accurate enough to reproduce stability results for a perturbed vortex ring found using the direct method.

(2) While we do resolve more of the features (e.g., the arms) of the core distortion

in the more refined calculations, we believe the dynamics of the core are still not fully resolved, since *i*) the mesh still crosses itself at finite time with 61 filaments; *ii*) the filaments at different initial radii move markedly out of phase with each other, so we do not yet know whether we have captured all of the radial modes; *iii*) many more new vortex segments are created for the finer calculations, indicating its increased sensitivity to the stretching taking place. However, we do not know what calculations with increased refinement will show.

(3) At the higher resolution ($N \approx 14000$) the MLC excites the $n = 4$ mode, but this mode and its multiples do not detract from the overall stability results.

(4) Using the the integral of vorticity to define the circulation of the filaments (Equation 6.3), we see better agreement between the $N \approx 2000$ and the $N \approx 14000$ calculations, and these differ from the earlier calculations in that they show an initial decline in amplitude rather than growth at all time. We suggest that this may be due to the fact that the initial $N \approx 2000$ calculations effectively studied a ring with a more concentrated distribution of vorticity, due to the method used in [39] to assign the circulation of the filaments.

We plan to conduct a more thorough study of the vortex rings, studying the effect of different numerical and physical parameters on the core deformation and stability of the vortex ring. We also plan to implement several diagnostics to investigate the accuracy of our calculations. A recent study [49] discusses appropriate numerical diagnostics for the three-dimensional vortex filament method. The invariants of the Euler equations—total vorticity $\Omega = \int \omega dV$, linear impulse $\mathbf{I} = \frac{1}{2} \int \mathbf{x} \times \omega dV$, and the kinetic energy $E = \frac{1}{2} \int \mathbf{u} \cdot \mathbf{u} dV$ —are presented as the relevant quantities to evaluate numerically; a convergent calculation should conserve these quantities.

The kinetic energy of a vortex filament system can be separated into two parts: the interaction energy, $E_{ij} = \sum_{i,j \neq i} \frac{\Gamma_i \Gamma_j}{8\pi} \frac{\ell_i \cdot \ell_j}{r_{ij}}$, where r_{ij} is the distance between the i^{th} and j^{th} segment, and the self-energy E_{ii}. To correctly measure conservation of energy in a calculation one must evaluate both parts of the energy. Chorin's derivation of the correct method for calculating E_{ii} is presented in [49].

In this thesis we have developed a two- and three-dimensional fast adaptive vortex method. Error and timing results show that the method of local corrections with adaptive mesh refinement greatly reduces the cost of calculations with many vortices while maintaining the accuracy of the direct vortex method. The cost of the direct method is $O(N^2)$; the cost of the MLC with AMR is $O(N)$ for large N.

Calculations of a three-dimensional vortex ring show that while additional features are captured using $N \approx 14000$ rather than $N \approx 2000$, further work is necessary to increase our understanding of the role of the deformation of the core and of the different radial modes in determining the stability of the vortex ring; this highlights the need for fast vortex methods, so that we can study problems of this nature with sufficient resolution.

Parameter	Figure					
	2,3	4-7,8a	8b,9a,10a	9b,10b	10c	11a,12b
Initial number of vortices	2040	2040	7320	7320	7320	15600
$\Delta r/R_0$	.109	.109	.0705	.0705	.0705	.0545
δ/R_0	.155	.155	.150	.150	.150	.0775
Number of radial stations	2	2	4	4	4	4
Vortices per cross-section	17	17	61	61	61	65
Initial segments per filament	120	120	120	120	120	240
Wavenumber of perturbation	9	12	12	12	12	12
ε/R_0	.02	.02	.02	.02	.02	.02
Δt	.1	.1	.05	.05	.05	.05
C=D	1.5	1.5	1.5	1.5	1.5	1.5
Method (Grid)	8-8	8-8	8-32	Direct	32-32	8-32
Method of Assigning Circulation	(6.1)	(6.1)	(6.1)	(6.1)	(6.1)	(6.2)
Maximum Time	100.	140.	40.	40.	10.	40.

Parameter	Figure				
	11b	11c	12a	13,14	15,16
Initial number of vortices	15600	15600	15600	2280	14640
$\Delta r/R_0$	.0545	.0545	.0545	.150	.0833
δ/R_0	.0775	.0775	.0775	.200	.111
Number of radial stations	4	4	4	4	4
Vortices per cross-section	65	65	65	19	61
Initial segments per filament	240	240	240	120	240
Wavenumber of perturbation	12	12	12	12	12
ε/R_0	.02	.02	.02	.02	.02
Δt	.05	.05	.05	.1	.05
C=D	1.5	1.5	1.5	1.5	2.5
Method/Grid	16-32	Direct	8-64	8-8/Direct	8-64
Method of Assigning Circulation	(6.2)	(6.2)	(6.2)	(6.3)	(6.3)
Maximum Time	20.	10.	40.	140.	70.

Table 6.1: Parameters used in vortex ring calculations.

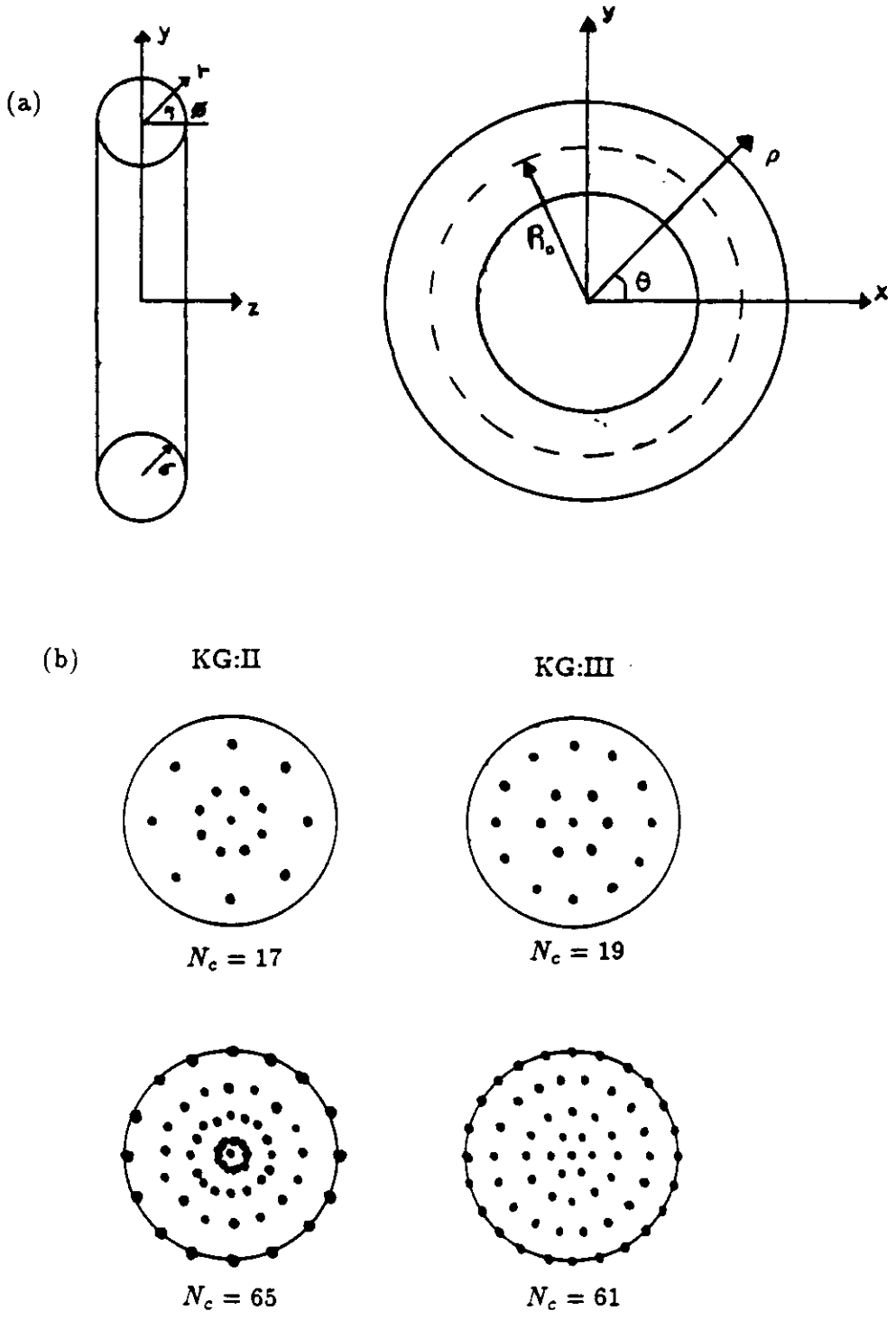


Figure 6.1: (a) Vortex ring and axes. (b) Locations of vortex filaments in the core for discretizations KG:II and KG:III.

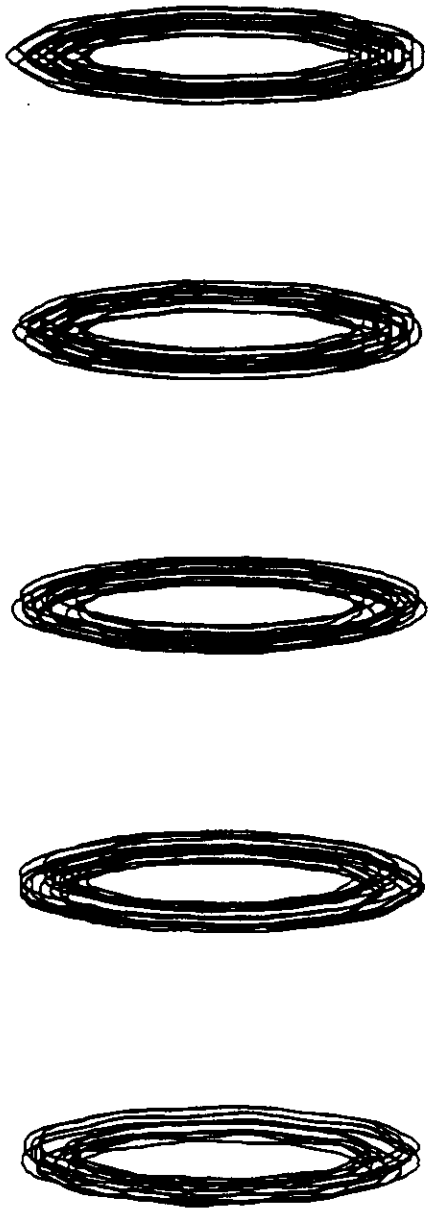


Figure 6.2: Ring with $N = 2040$ (Mesh KG:II), $n = 9$ perturbation at times $t = 0, 10, 40, 70, 100$, circulation calculated by solving the system of linear equations (6.1); calculation done using MLC. Shown at angle $\pi/3$ from the z-axis. Note that the initial perturbation does not grow in amplitude; 9 is a stable wavenumber.

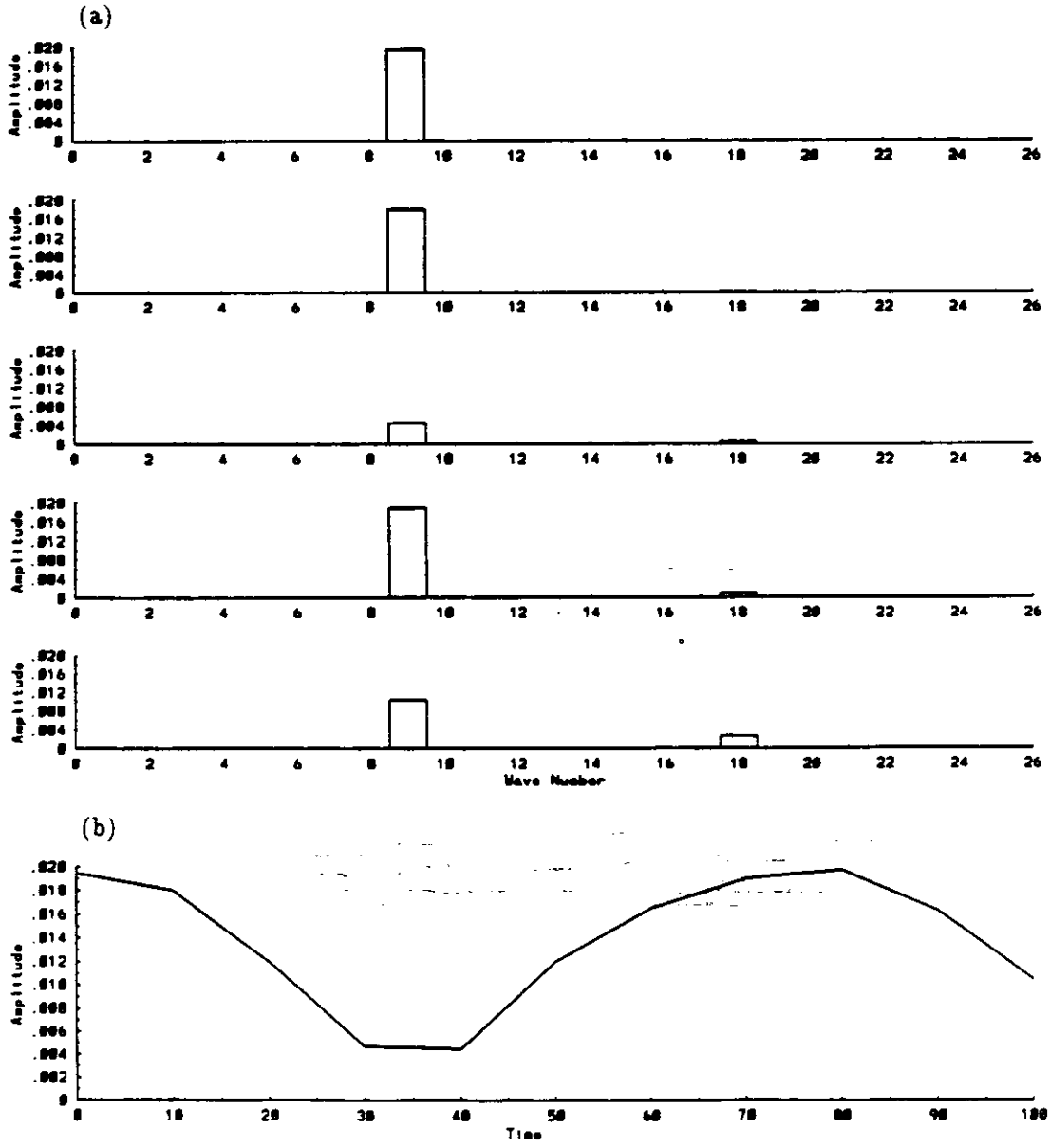


Figure 6.3: (a) Amplitude vs. wavenumber for $n = 9$ ring at $t = 0, 10, 40, 70, 100$; (b) time evolution of $n = 9$ mode from $t = 0$ to $t = 100$. Discretization as in Figure 6.2. The amplitude of the perturbation oscillates rather than grows in time.

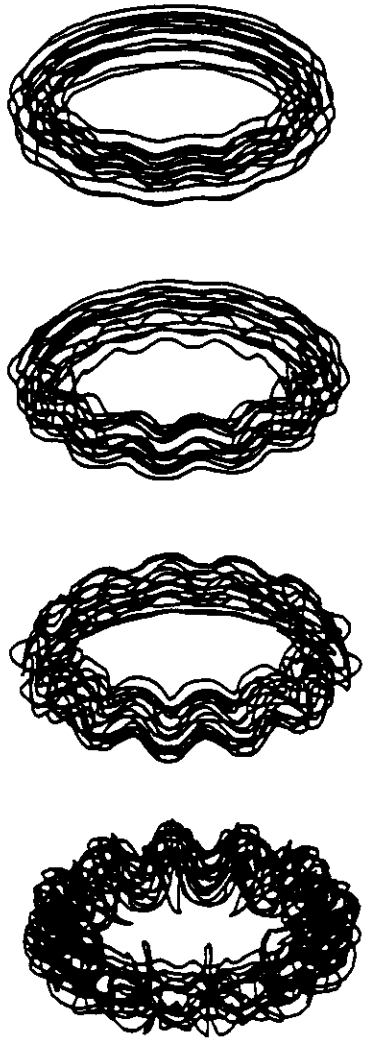


Figure 6.4: Ring with $N = 2040$ (Mesh KG:II), $n = 12$ perturbation, circulation calculated by solving the system of linear equations (6.1), at times $t = 30, 60, 90, 120$, calculation done using MLC. Shown at angle $\pi/3$ from the z-axis. Note that the amplitude of the initial perturbation grows markedly; 12 is an unstable wavenumber.

Figure 6.5: Ring from Figure 6.4 at time $t = 140$ shown at angles $0, \pi/3, \pi/2$ from the z-axis.

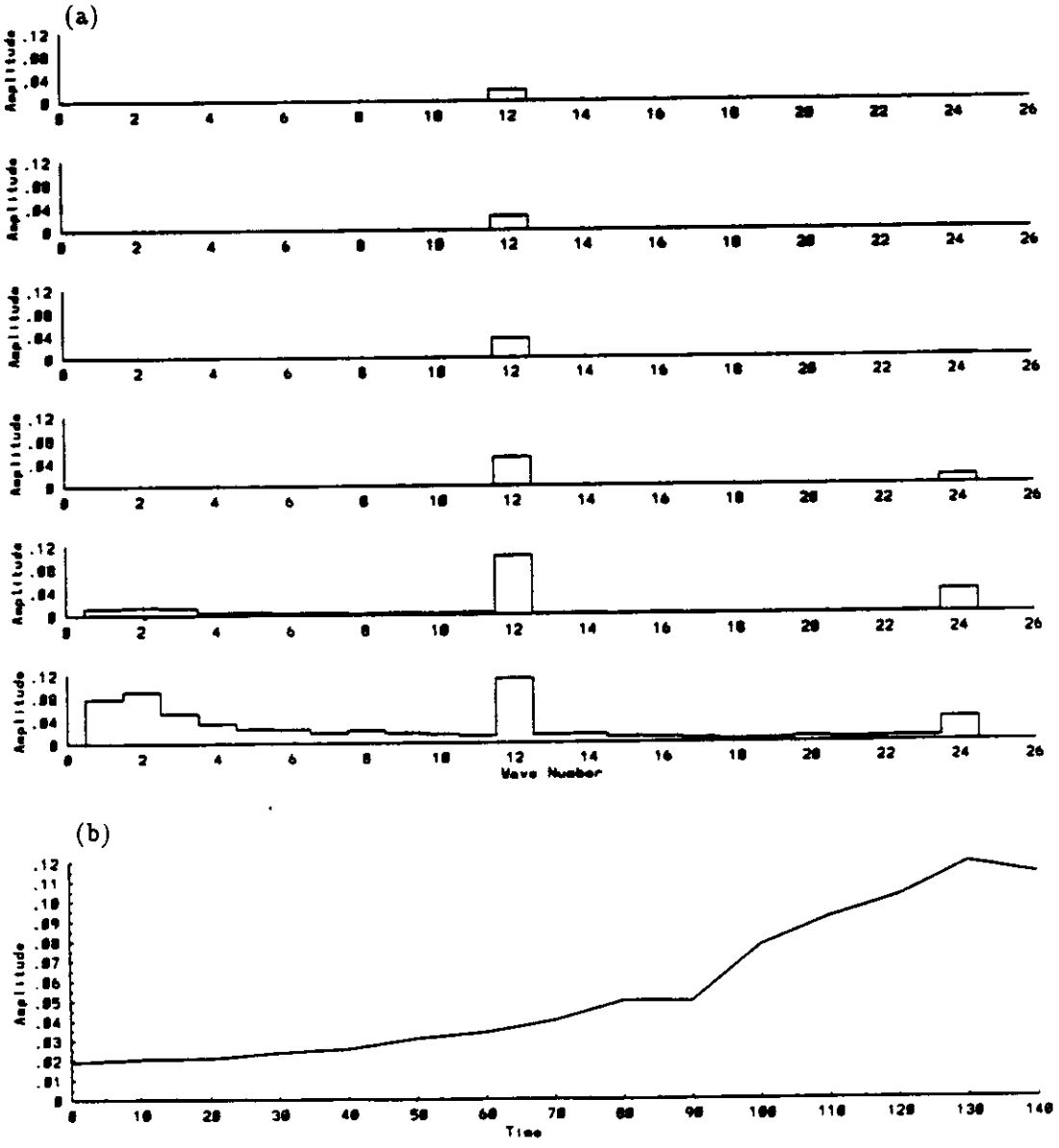


Figure 6.6: (a) Amplitude vs. wavenumber at $t = 0, 30, 60, 90, 120, 140$ for ring with $N = 2040$, $n = 12$ perturbation; (b) time evolution of $n = 12$ mode. Discretization as in Figure 6.4. Note the difference in scale between these plots and Figure 6.3; here the amplitude of the $n = 12$ mode reaches six times its initial value.

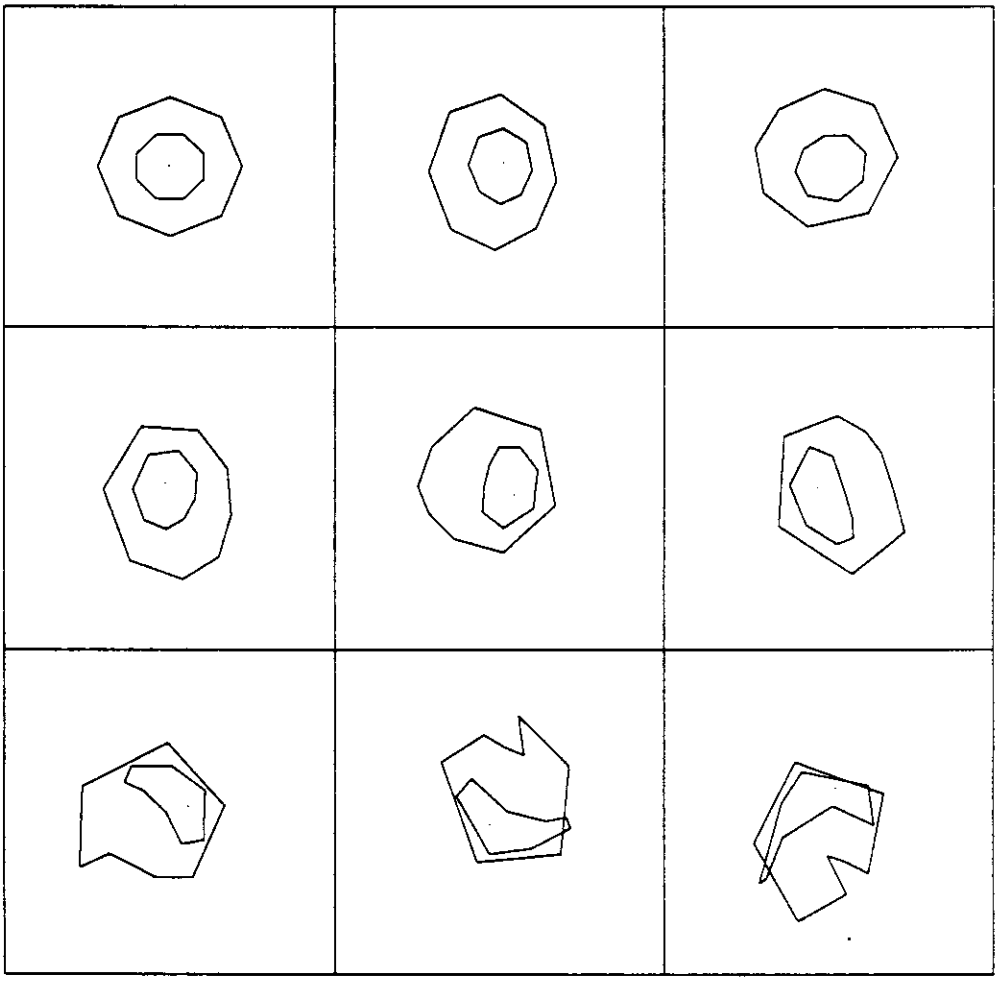


Figure 6.7: Intersection of 17 filaments with the $\theta = 0$ plane for ring with $N = 2040$, $n = 12$ perturbation at times $t = 0, 10, 20, 30, 40, 50, 60, 70, 80$; discretization as in Figure 6.4. Note that the Lagrangian mesh crosses over itself by $t = 70$.

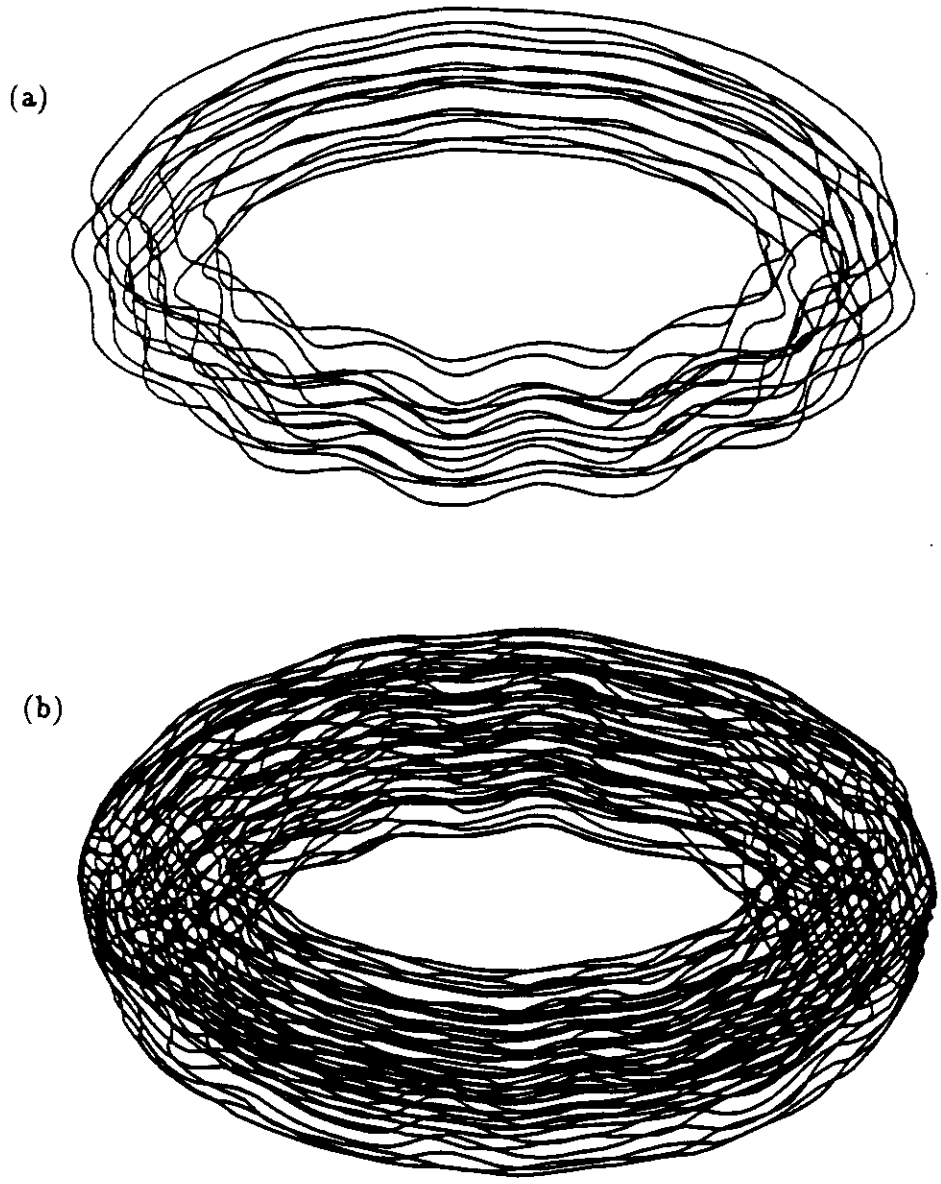


Figure 6.8: (a) Ring with 17 filaments, $N = 2040$ (Mesh KG:II) at $t = 40$. Calculation done using MLC. (b) Ring with 61 filaments, $N = 7320$ (Mesh KG:III) at $t = 40$. Calculation done using MLC with AMR. Both have $n = 12$ perturbation, circulation calculated by solving the system of linear equations (6.1).

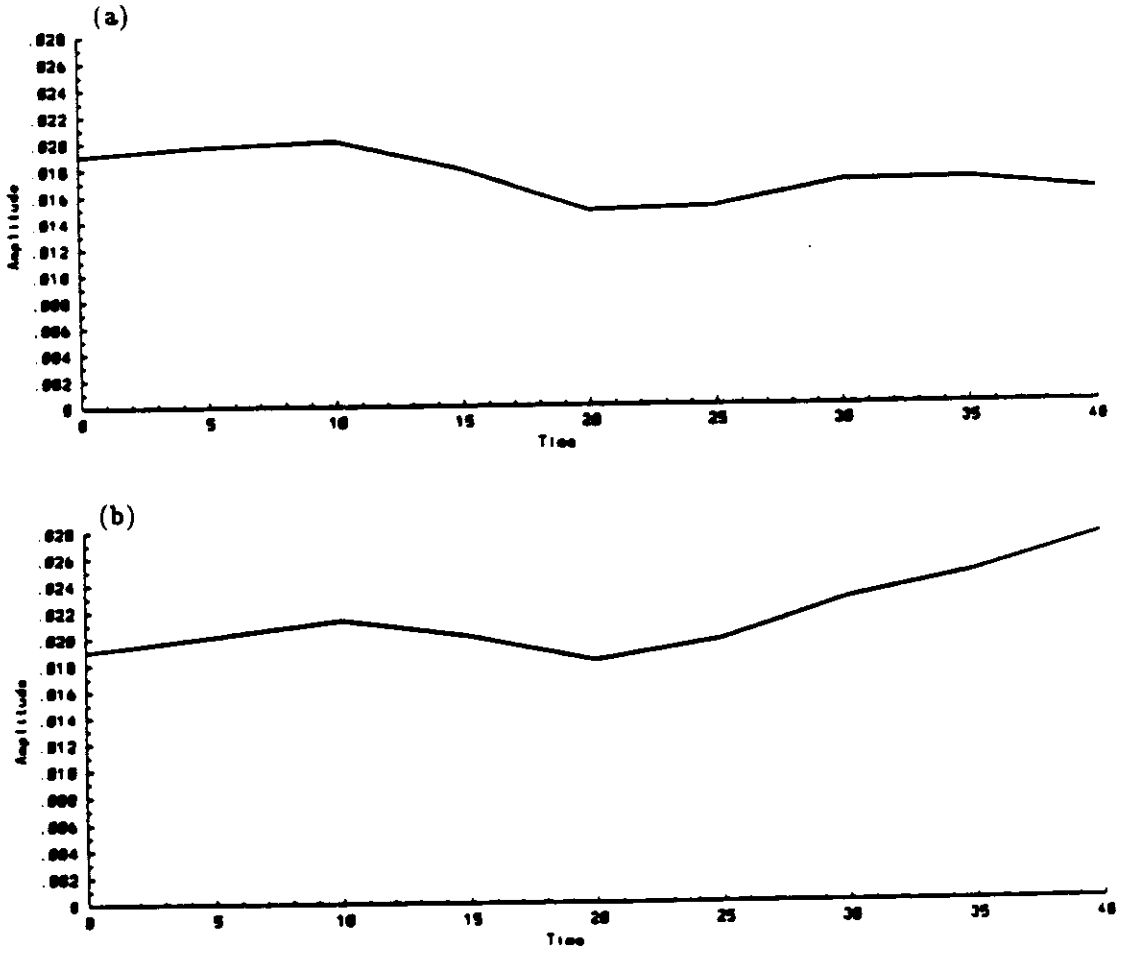


Figure 6.9: Time evolution of $n = 12$ mode for calculations using (a) MLC with AMR, (b)direct method; discretization as in Figure 6.8b. Note that the amplitude shows a slight decline in each case, reaching a minimum at $t = 20$.

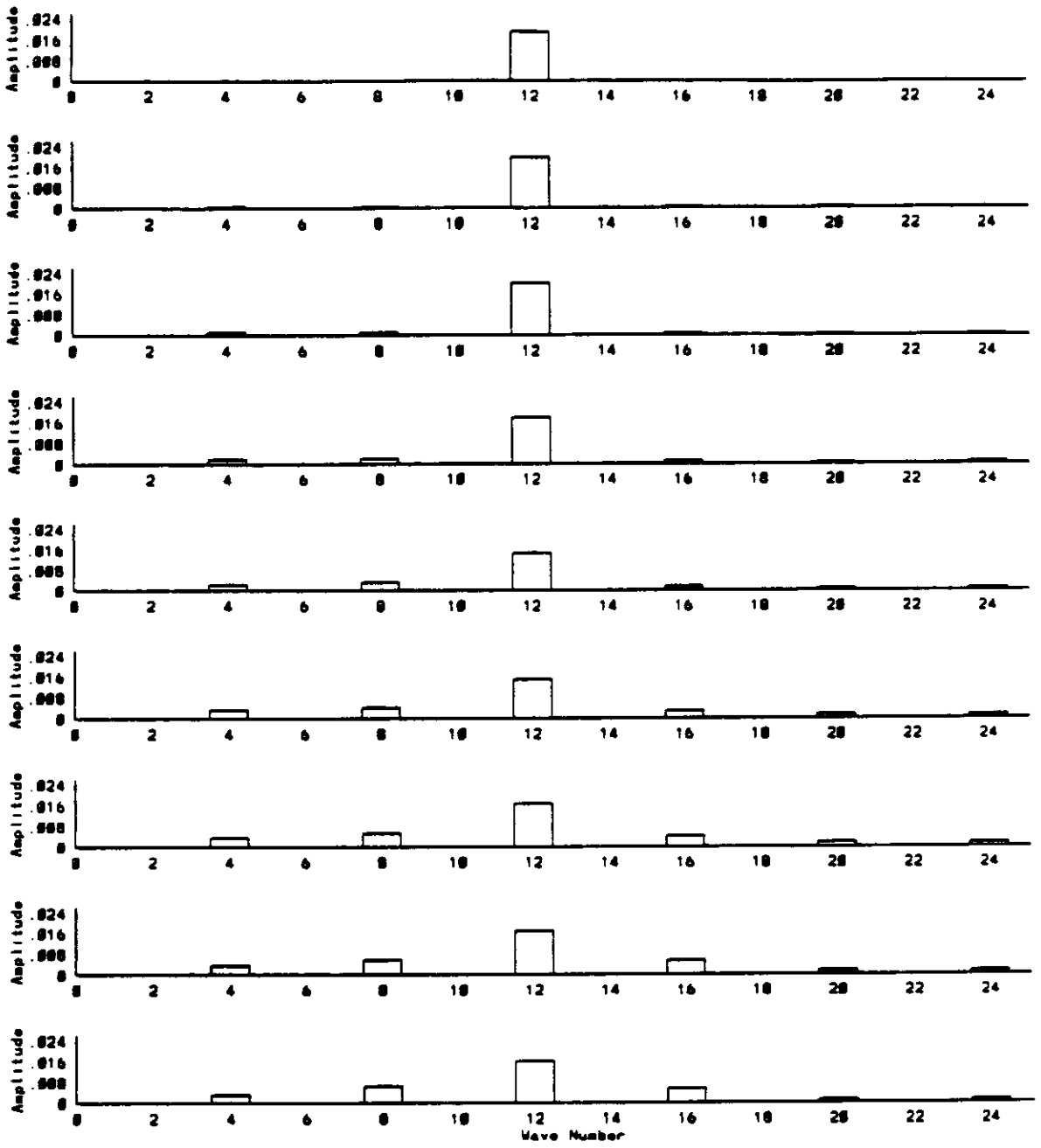


Figure 6.10: (a) Amplitude vs. wavenumber for calculations of $N = 7320$ ring, $n = 12$ perturbation, at times $t = 0, 5, 10, 15, 20, 25, 30, 35, 40$, using MLC with AMR. Discretization as in Figure 6.8b. Note the $n = 4$ mode and its multiples in (a) and (c); these are present for calculations using MLC and MLC with AMR, but not for calculations with the direct method. This verifies that MLC, not the addition of AMR to MLC, is responsible for the additional modes. Note also that the amplitudes of the additional modes do not keep growing; they seem to level off by $t = 30$.

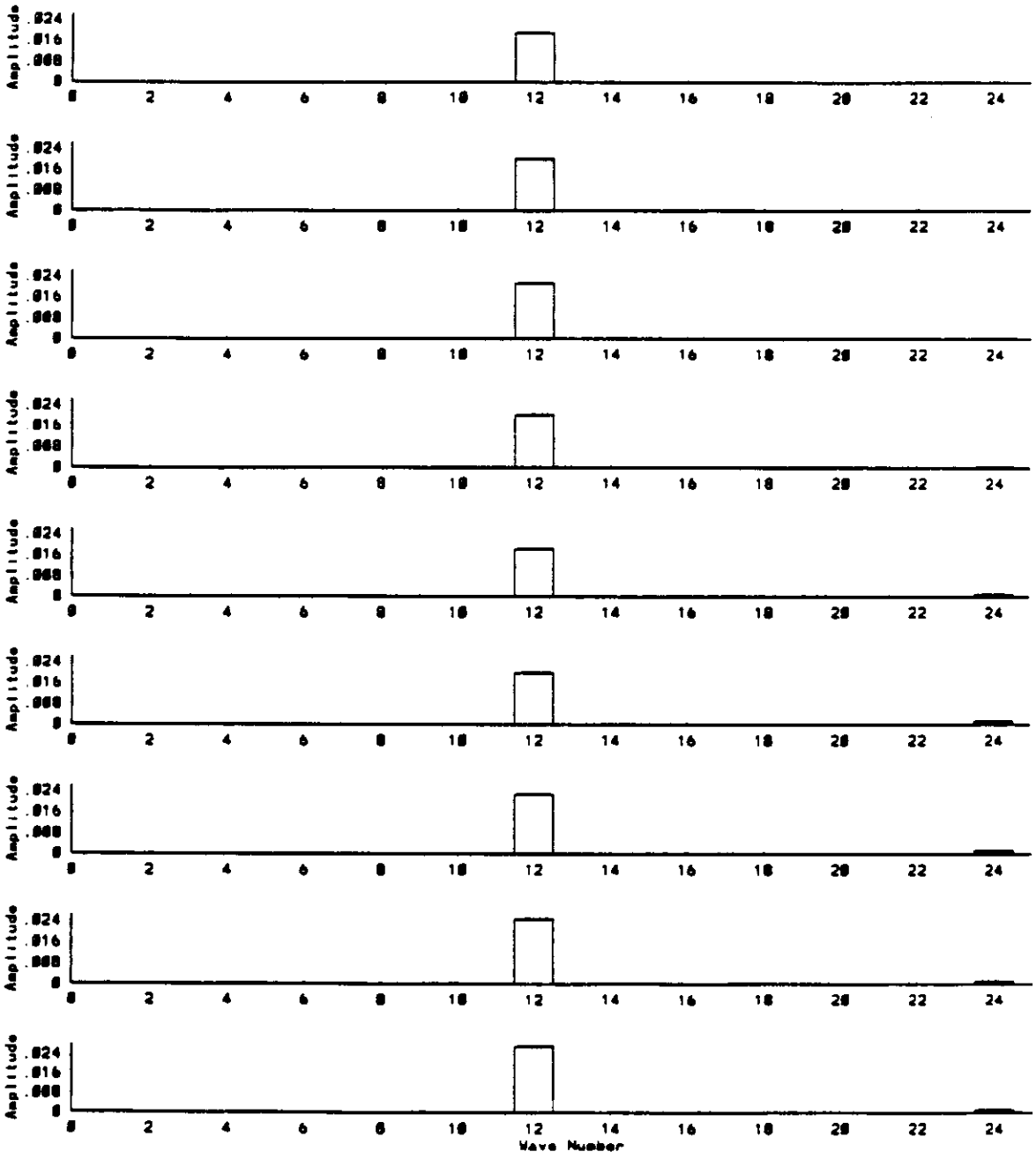


Figure 6.10: (b) Amplitude vs. wavenumber for calculations of $N = 7320$ ring, $n = 12$ perturbation, at times $t = 0, 5, 10, 15, 20, 25, 30, 35, 40$, using the direct method. Discretization as in Figure 6.8b. Note that with the direct method we do not see the $n = 4$ mode and its multiples.

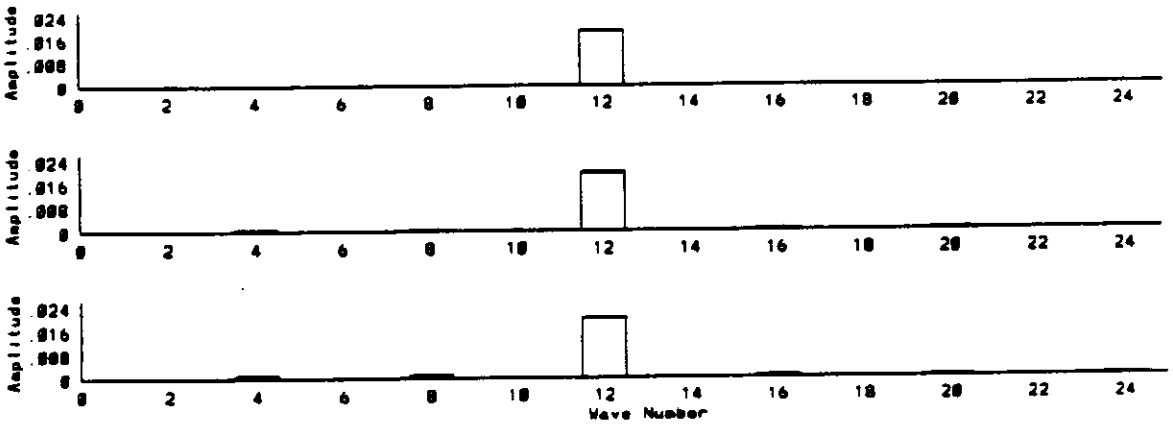


Figure 6.10: (c) Amplitude vs. wavenumber for calculations of $N = 7320$ ring, $n = 12$ perturbation, at times $t = 0, 5, 10$ using MLC on a uniform grid. Discretization as in Figure 6.8b. Note the $n = 4$ mode and its multiples in (a) and (c); these are present for calculations using MLC and MLC with AMR, but not for calculations with the direct method. This verifies that MLC, not the addition of AMR to MLC, is responsible for the additional modes.

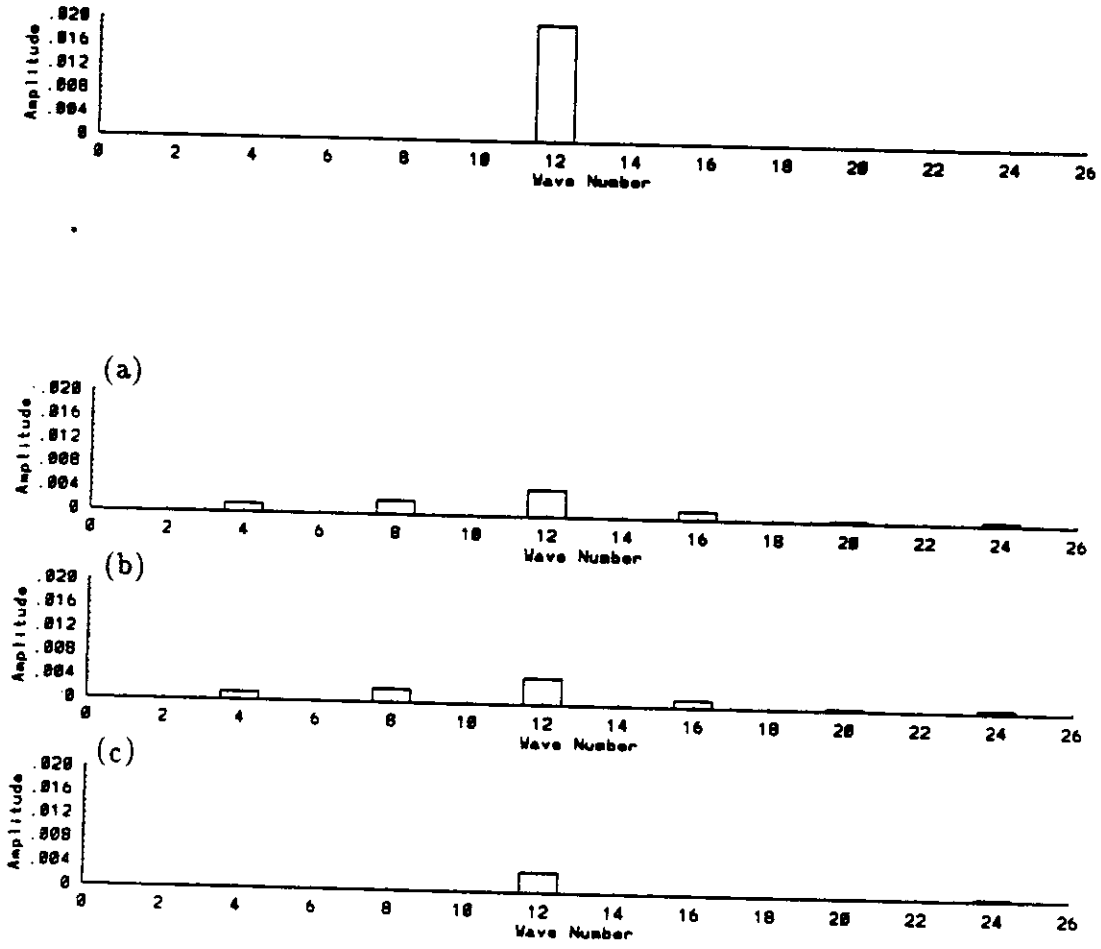


Figure 6.11: Amplitude vs. wavenumber for calculations of ring with $N = 15600$ (Mesh KG:II), $n = 12$ perturbation, circulation of each filament calculated as the product of the point value of vorticity at the location of the filament and the area represented by that filament, at time $t = 0$, and (a) at $t = 10$ for MLC with AMR on 8-32 grid, (b) at $t = 10$ for MLC with AMR on 16-32 grid, and (c) at $t = 10$ for the direct method. Note that the extra $n = 4$ mode and its multiples appear for MLC with AMR on a 8-32 grid and on a 16-32 grid, but not for the direct method. This verifies that the boundary conditions on the $8 \times 8 \times 8$ grid are sufficiently resolved, since we see no difference with the base grid of $8 \times 8 \times 8$ and $16 \times 16 \times 16$.

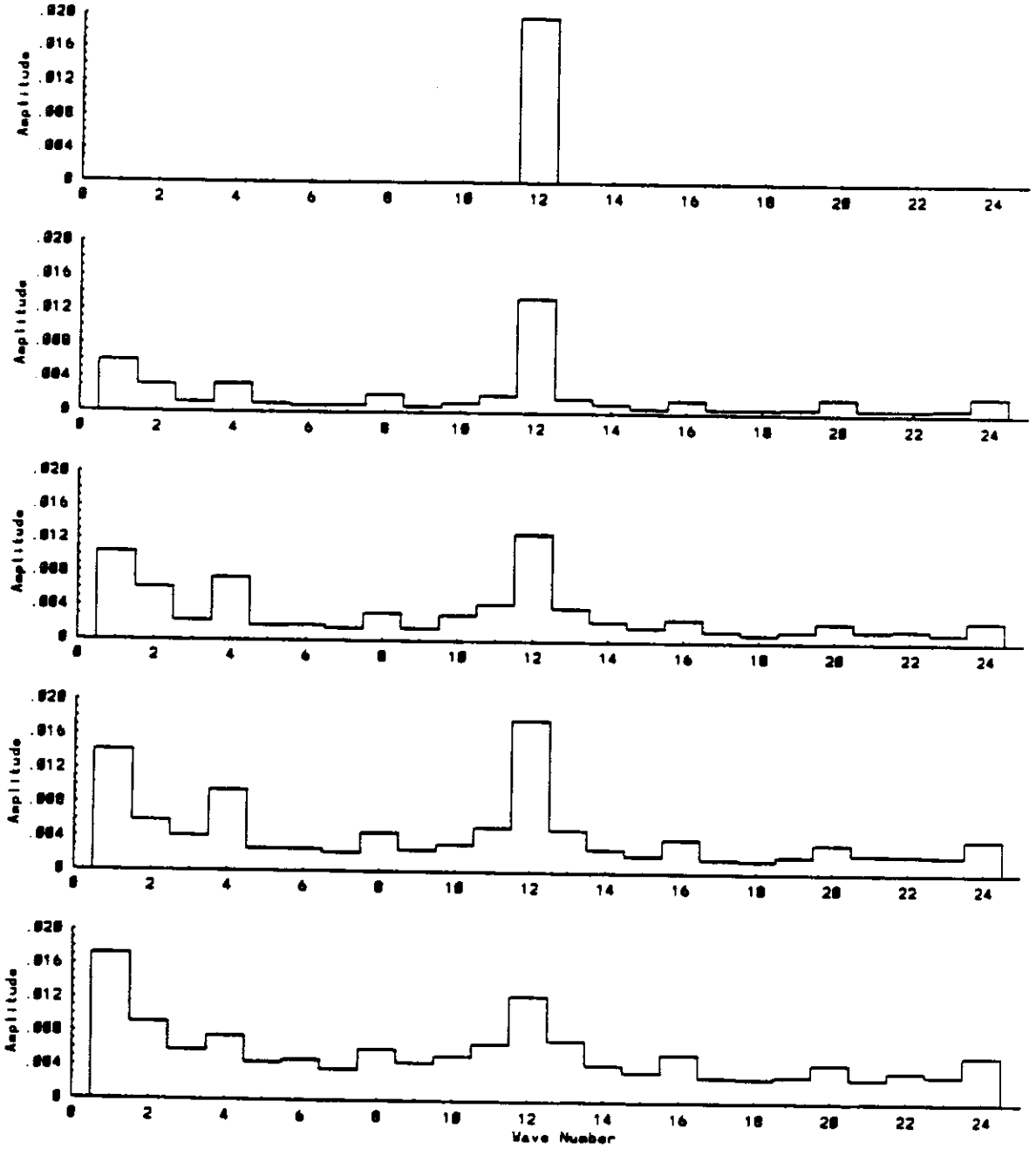


Figure 6.12: (a) Amplitude vs. wavenumber for calculations of ring with $N = 15600$ (Mesh KG:II), $n = 12$ perturbation, at times $t = 0, 10, 20, 30, 40$, using MLC with AMR on an 8-64 grid. We show this to demonstrate that the correction radius must be large enough relative to the core radius of the vortices to ensure accuracy.

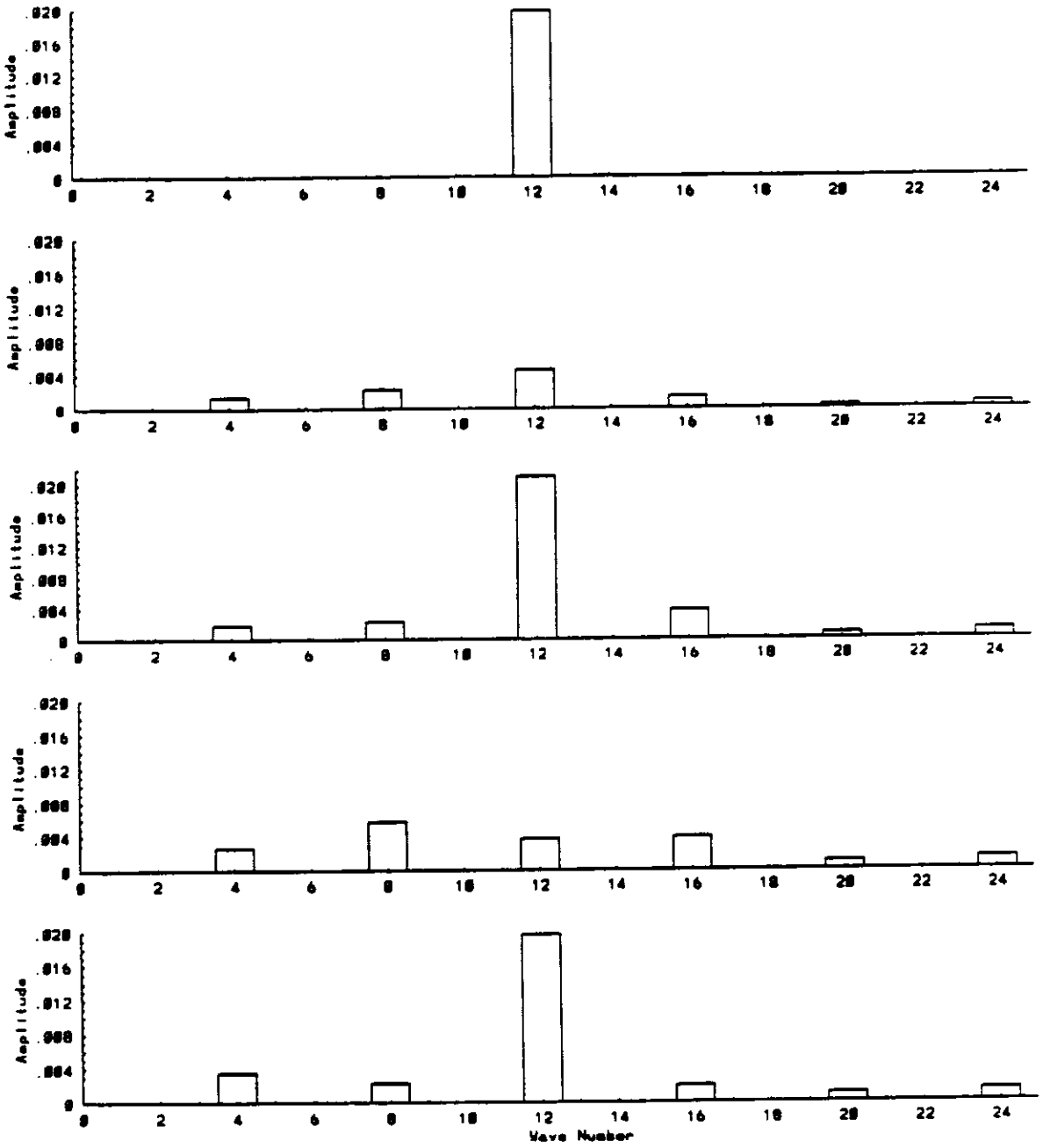


Figure 6.12: (b) Amplitude vs. wavenumber for calculations of $N = 15600$ (Mesh KG:II) ring, $n = 12$ perturbation, at times $t = 0, 10, 20, 30, 40$, using MLC with AMR on an 8-32 grid. Contrast with Figure 6.12a.

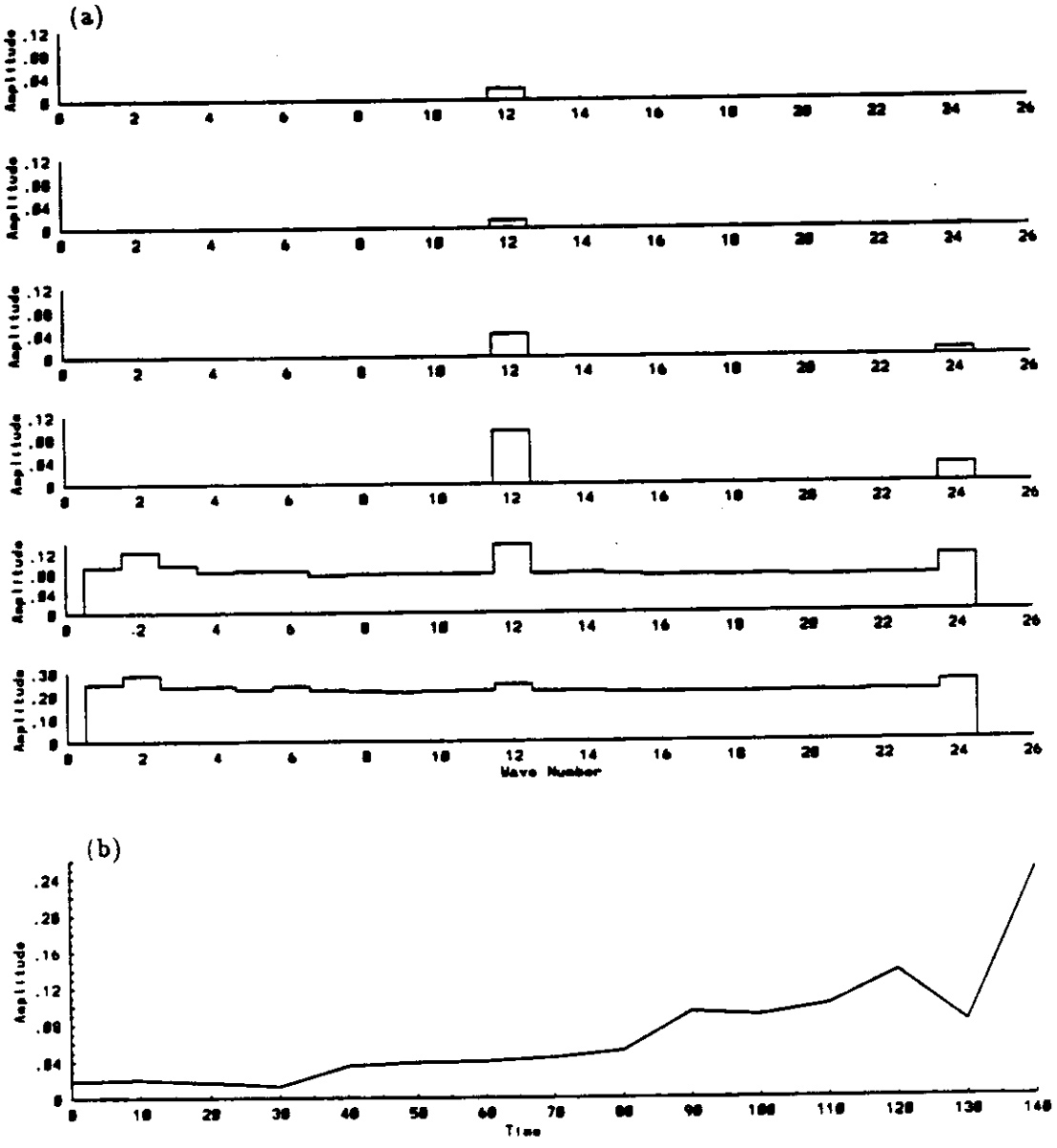


Figure 6.13: (a) Amplitude vs. wavenumber for calculations with $N = 2280$ (Mesh KG:III), $n = 12$ perturbation, circulation calculated by integrating the vorticity over the area represented by each filament (Equation 6.3), at times $t = 0, 30, 60, 90, 120, 140$, using MLC and the direct method. (b) Time evolution of $n = 12$ mode for calculations using MLC and the direct method. Both curves are plotted; they are indistinguishable.

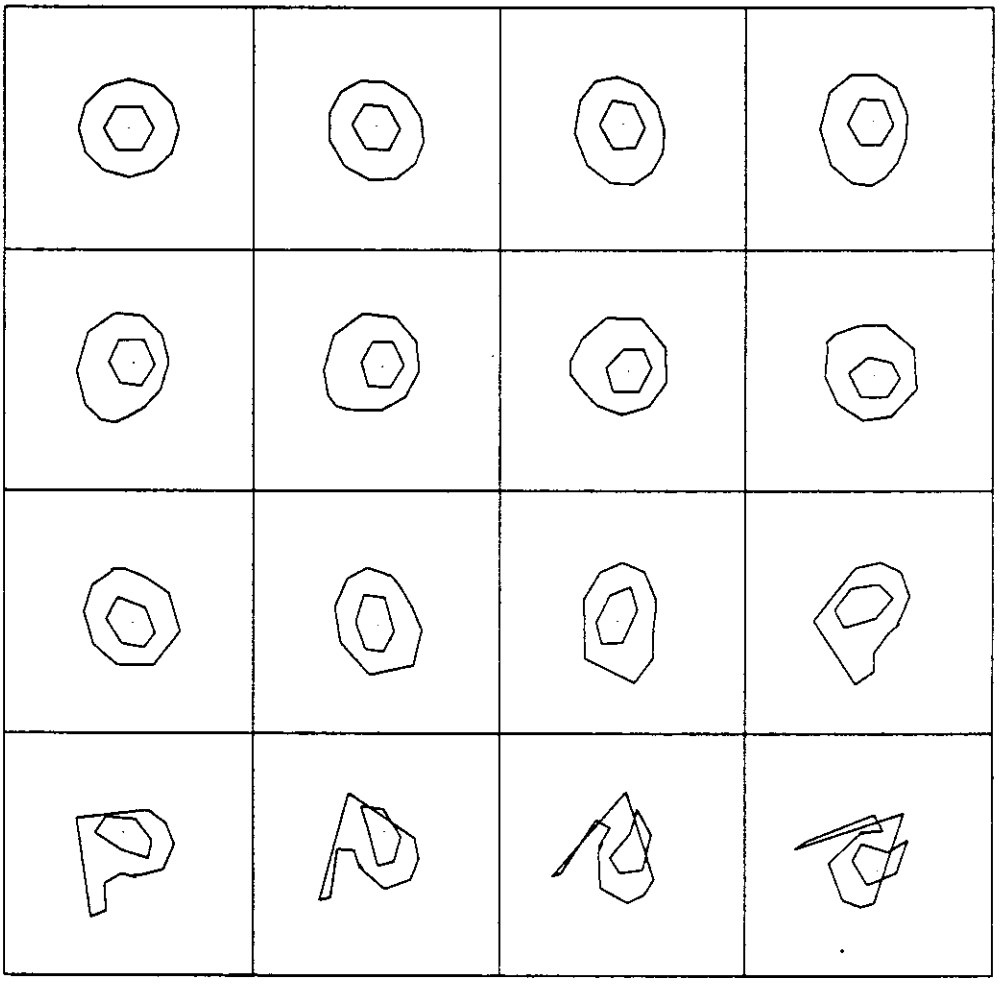


Figure 6.14: Intersection of 19 filaments with the $\theta = 0$ plane for ring with $N = 2280$, $n = 12$ perturbation, at times $t = 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75$; data as in Figure 6.13. Calculation done using MLC; discretization as in Figure 6.13. We see that the Lagrangian mesh crosses itself by $t = 65$, similarly to the calculations shown in Figure 6.7

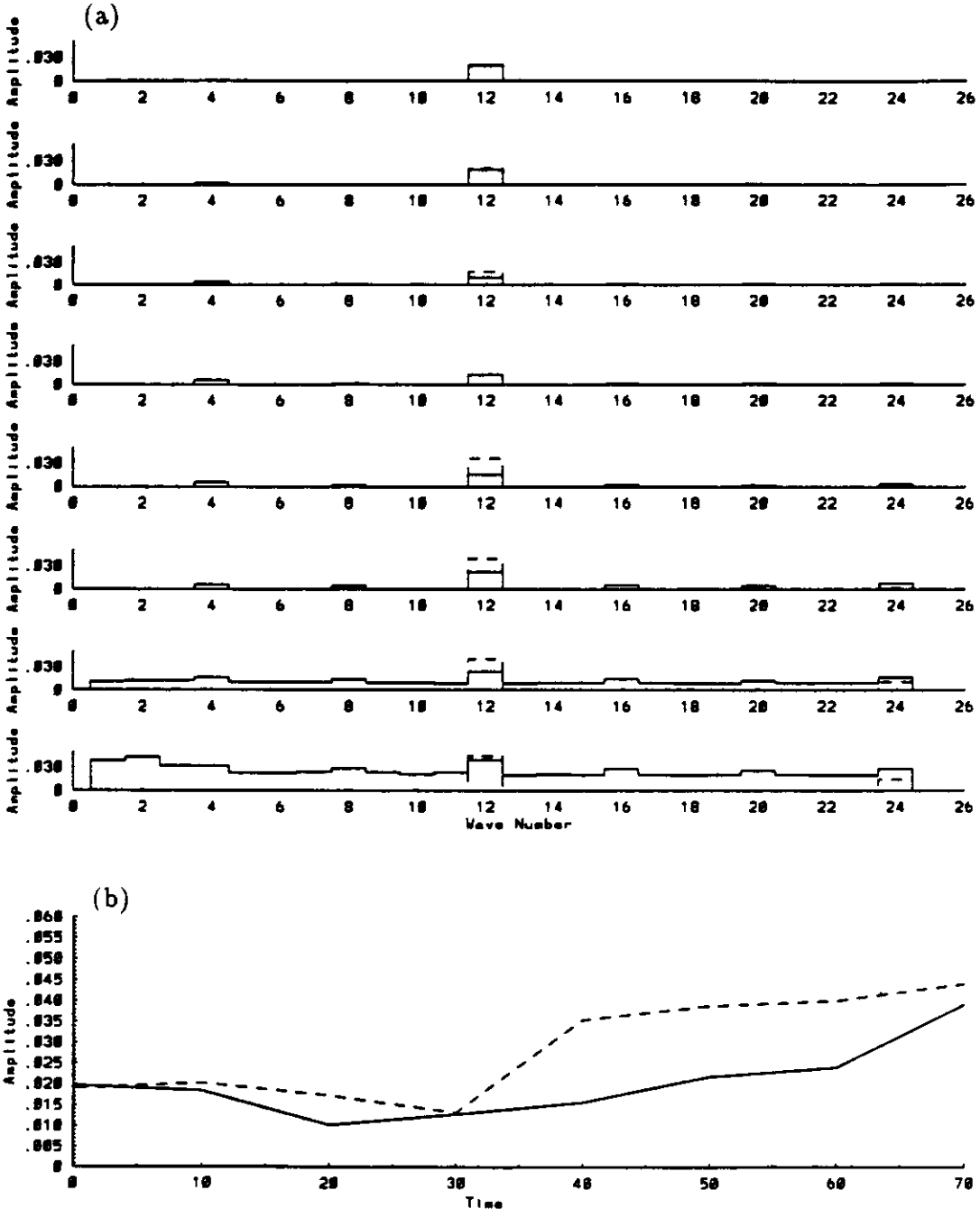


Figure 6.15: (a) Amplitude vs. wavenumber for calculations of ring with N = 14640 (Mesh III), n = 12 perturbation, circulation calculated by integrating the vorticity over the area represented by each filament (Equation 6.3), at times t = $0, 10, 20, 30, 40, 50, 60, 70$, using MLC with AMR (8-64) with $C = D = 2.5$ (solid). This is contrasted with the N = 2280 calculation (dashed). (b) Time evolution of n = 12 mode for the calculation with N = 14640 using MLC with AMR (solid) and the calculation with N = 2280 using MLC (dashed).

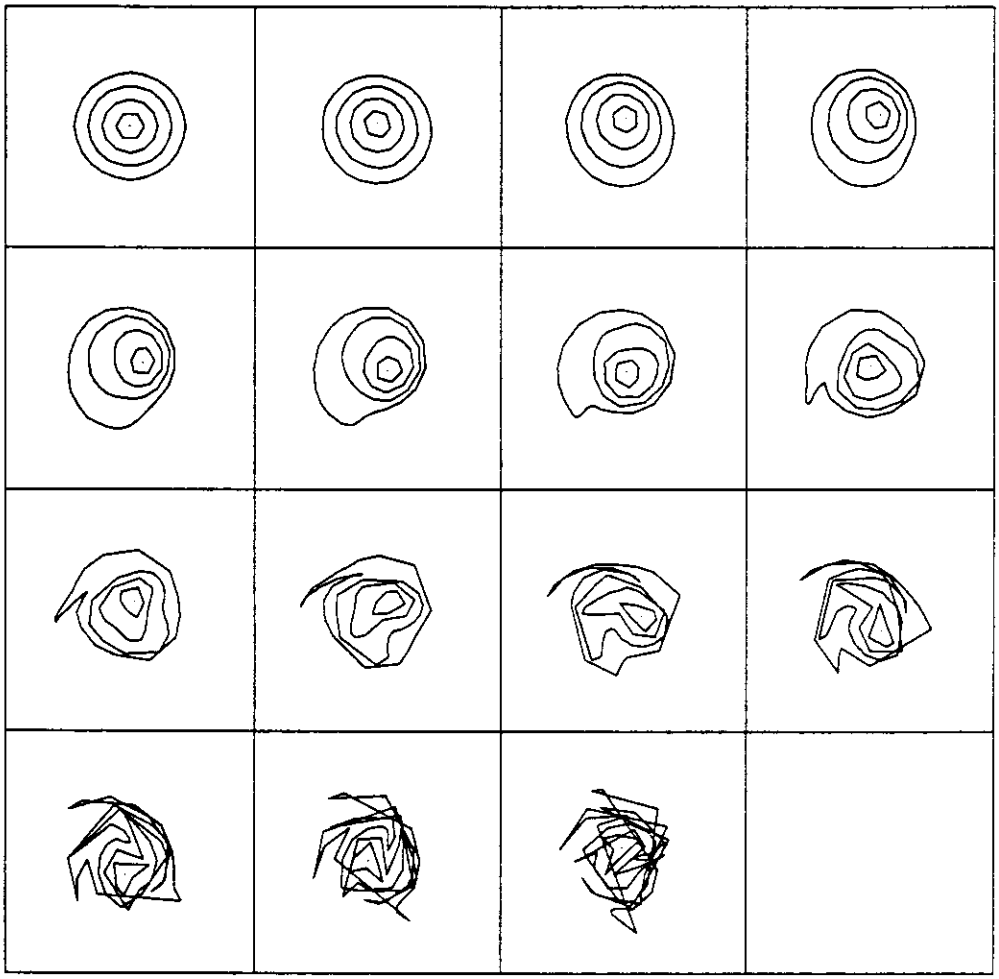


Figure 6.16: Intersection of 61 filaments with the $\theta = 0$ plane for ring with $N = 14640$, $n = 12$ perturbation, at times $t = 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70$. Calculation done using MLC with AMR; discretization as in Figure 6.15. The Lagrangian mesh crosses itself by $t = 55$, but note the new features of the core distortion not seen in Figure 6.14.

References

[1] C. Anderson, "A Method of Local Corrections for Computing the Velocity Field Due to a Distribution of Vortex Blobs", *J. Comput. Phys.*, **62** (1986), p. 111-123.

[2] C. Anderson "An Implementation of the Fast Multipole Method Without Multipoles," *CAM Report 90-14, Department of Mathematics, UCLA* (July 1990).

[3] C. Anderson "Domain Decomposition Techniques and the Solution of Poisson's Equation in Infinite Domains," *Domain Decomposition Methods,* eds. P. Chan, R. Glowinski, J. Periaux, O. Widlund, Proceedings of the Second International Symposium on Domain Decomposition Methods, Los Angeles, CA, 1988, p. 129.

[4] C. Anderson and C. Greengard, "The vortex ring problem at infinite Reynolds number", *Comm. Pure and Applied Maths.*, xlii, p. 1123.

[5] C. Anderson and C. Greengard, "On Vortex Methods," *SIAM J. Numer. Anal.*, **22** (1985), p. 413-439.

[6] S.B. Baden, "Run-time Partitioning of Scientific Continuum Calculations Running on Multiprocessors," LBL-23625, Lawrence Berkeley Laboratory, 1987. (Ph.D. Dissertation in the Computer Science Department, U.C. Berkeley)

[7] S.B. Baden, "Very Large Vortex Calculations in Two Dimensions," *Lecture Notes in Mathematics,* Springer-Verlag, New York, 1988. Proceedings of the UCLA Workshop on Vortex Methods, Los Angeles, CA, May 20-22, 1987.

[8] J.T.. Beale and A. Majda, "Vortex Methods I: convergence in three dimensions," *Math. Comput.*, **39** (1982), p. 1-27.

[9] J.T.. Beale and A. Majda, "Vortex Methods II: higher order accuracy in two and three dimensions," *Math. Comput.*, **39** (1982), p. 29-52.

[10] J.T. Beale and A. Majda, "High Order Accurate Vortex Methods with Explicity Velocity Kernels," *J. Comput. Phys.*, **58** (1985), p. 188-208.

[11] M.J. Berger and J. Oliger, "Adaptive mesh refinement for hyperbolic partial differential equations", *J Comput. Phys.*, **54** (1984), p. 484.

[12] M.J. Berger and P. Colella, "Local adaptive mesh refinement for shock hydrodynamics", *J Comput. Phys.*, **82** (1989), p. 64-84.

[13] M.J. Berger and I. Rigoutsos, "An Algorithm for Point Clustering and Grid Generation", *NYU Technical Report 501*, (April 1990). To appear, *IEEE Trans. Systems, Man and Cybernetics.*

[14] M.J. Berger and A. Jameson, "An Adaptive Multigrid Method for the Euler Equations," *Lecture Notes in Physics* 218, Springer-Verlag. Presented at the 9th International Conference on Numerical Methods in Fluid Dynamics, Paris, June 1984.

[15] J. Bell, M.J. Berger, J.S. Saltzman, and M. Welcome, "Three Dimensional Adaptive Mesh Refinement for Hyperbolic Conservation Laws, ", to appear.

[16] J.B Bell, P. Colella, J.A. Trangenstein, and M. Welcome, "Adaptive mesh refinement for moving quadrilateral grids," *Proceedings, 9th AIAA Computational Fluid Dynamics Conference,* Buffalo, NY, June 13-15, 1989, p. 471-479.

[17] J. Bolstad, Ph.D. thesis, Dept. of Computer Science, Stanford University, 1982 (unpublished).

[18] A. Brandt, "Multi-level adaptive solutions to boundary value problems," *Mathematics of Computation*, **31** (1977), p. 333-390.

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

[20] T.F. Buttke, "Fast Vortex Methods in Three Dimensions,"submitted for publication.

[21] J. Carrier, L. Greengard and V. Rokhlin, "A Fast Adaptive Multipole Algorithm for Particle Simulations," *SIAM J. Sci. Stat. Comput.* **9** (1988), p. 669-686.

[22] A.J. Chorin, "Numerical study of slightly viscous flow", *J. Fluid Mech.*, **57** (1973), p. 785-796.

[23] A.J. Chorin, "Vortex Models and Boundary Layer Instability", *SIAM J. Sci. Stat. Comp.*, **1** (1980), p. 1-24.

[24] A.J. Chorin, "Estimates of Intermittency, Spectra, and Blow-Up in Developed Turbulence," Vortex," *Commun. Math. Phys.*, **34** (1981), p. 853-866.

[25] A.J. Chorin, "The Evolution of a Turbulent Vortex," *Commun. Math. Phys.*, **83** (1982), p. 571-535.

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

[27] L. Collatz, *The numerical treatment of differential equations*, Springer-Verlag, New York, 1966.

[28] H. Cottet, These de 3eme cycle, p. 6, Universite Pierre et Marie Curie, Paris, 1982.

[29] H. Cottet and P.S. Raviart, "Particle Methods for one-dimensional Vlasov-Poisson equations," *SIAM J. Numer. Anal.*, **21** (1984), p. 52-76.

[30] J.P. Davis, *Interpolation and Approximation*, Dove, New York, 1975.

[31] C. Greengard, "Convergence of the vortex filament method," *Math. Comp.*, **47** (1986), p. 387.

[32] L. Greengard and V. Rokhlin, "A fast algorithm for particle simulations," *J. Comput. Phys.*, **73** (1987), p. 325-348.

[33] W.D. Gropp, "A Test of Moving Mesh Refinement for 2-D Scalar Hyperbolic Problems," *SIAM J. Sci. Stat. Comput.* **4** (1980), p. 191-197.

[34] O.H. Hald " " *SIAM J. Numer. Anal.* **16** (1979), p. 726.

[35] O.H. Hald and V.M. Del Prete, *Math. Comput.* **32** (1978), p. 791.

[36] R.W. Hockney and J.W. Eastwood, *Computer Simulations Using Particles*, McGraw-Hill, New York, 1981.

[37] J. Katznelson, "Computational Structure of the N-Body Problem," *SIAM J. Sci. Stat. Comput.* **4** (1989), p. 787-815.

[38] R. Krasny, "A study of singularity formation in a vortex sheet by the point vortex approximation," *J. Fluid Mech.*, **167** (1986), p. 65.

[39] O.M. Knio and A.F. Ghoniem, "Numerical Study of a Three-Dimensional Vortex Method," *J. Comput. Phys.* **86** (1990) p. 75-106.

[40] A. Leonard, "Vortex Methods for Flow Simulation," *J. Comput. Phys.*, **37** (1980), p. 289-335.

[41] A. Leonard, "Computing Three-Dimensional Incompressible Flows with Vortex Elements," *Ann Rev. Fluid Mech.*,**17** (1985), p. 523-559.

[42] H. Lugt, *Vortex Flow in Nature and Technology*, Wiley-Interscience, New York, 1983.

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

[44] T. Maxworthy, "The structure and stability of vortex rings," *J. Fluid Mech.* **51** (1972) p. 15.

[45] T. Maxworthy, "Turbulent Vortex Rings," *J. Fluid Mech.* **64** (1974) p. 227.

[46] T. Maxworthy, "Some experimental studies of vortex rings," *J. Fluid Mech.* **81** (1977) p. 625.

[47] M. Perlman, "On the Accuracy of Vortex Methods," *J. Comput. Phys.* **59** (1985), p. 200-223.

[48] H.B. Phillips, *Vector Analysis*, J. Wiley and Sons, Inc., NY ,1933, p. 187.

[49] A. Qi, "Three Dimensional Vortex Methods for the Analysis of Wave Propagation on Vortex Filaments," Ph.D. thesis in the Dept. of Mathematics, University of California, Berkeley, 1990.

[50] L. Rosenhead, *Proc. R. Soc. London Ser. A* **134** (1935), p.170.

[51] P.G. Saffman, "The number of waves on unstable vortex rings," *J. Fluid Mech.* **84** (1978) p. 465.

[52] J. Strain, "Fast Potential Theory I: Poisson Solvers for a Cube," 1990 (unpublished).

[53] J. Strain, "Fast Potential Theory II: Layer Potentials and Discrete Sums," 1990 (unpublished).

[54] L. van Dommelen and E. Rundensteiner,"Fast Adaptive Summation of Point Forces in Two-Dimensional Poisson Equations," *J. Comput. Phys.* **83** (1989), p. 126-147.

[55] M. van Dyke, *An Album of Fluid Motion*, Parabolic, Stanford, CA,1982.

[56] S.E. Widnall and J.P. Sullivan, *Proc. Roy. Soc. London Ser. A*, **332** (1973), p. 335.

[57] S.E. Widnall, D.B. Bliss and A Zalay, "The instability of short waves on a vortex ring," *J. Fluid Mech.* **66** (1974) p. 35.

[58] S.E. Widnall and C.-Y. Tsai, "The instability of the thin vortex ring of constant vorticity," *Proc. Roy. Soc. London Ser. A*, **287** (1977), p. 273.