PART I: THE NUMERICAL SOLUTION OF HYPERBOLIC SYSTEMS OF CONSERVATION LAWS

PART II: COMPOSITE OVERLAPPING GRID TECHNIQUES

Thesis by

William Douglas Henshaw

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1985

(Submitted May 23, 1985)

Acknowledgements

I would like to thank my advisor Heinz-Otto Kreiss for all his help and suggestions. My stay as a graduate student has been both exciting and fun. I would also like to thank my friends at Caltech. Special thanks are due to Michael Naughton who worked with me on a section of this thesis, and to Geoff Chesshire (*Mr. Geoff*).

Support for this work has been in the form of teaching assistantships and fellowships from the Institute. In addition, research assistantships have been provided by the National Science Foundation under contracts DMS-8312264 and ATM-8201207 and the Navy under contract N00014-83-K-0422.

The computations in this thesis were made on a variety of machines. At Caltech work was performed on the Fluid Dynamics Vax 11/750 in the Applied Mathematics Department and on an IBM 4341. The computational facilities at the National Center for Atmospheric Research in Boulder Colorado were also utilized.

Most of all I would like to thank my parents.

Abstract

Part I

A method is described for the numerical solution of hyperbolic systems of conservation laws in one space dimension. The basis of the scheme is to use finite differences where the solution is smooth and the method of characteristics where the solution is not smooth. The method can accurately represent shocks. Results are presented for the solution of the equations of gas dynamics. The examples illustrate the accuracy of the method when discontinuities are present and the code's performance on difficult problems of interacting shocks and shock formation.

Part II

Techniques for the numerical solution of partial differential equations on composite overlapping meshes are discussed. Methods for the solution of time dependent and elliptic problems are illustrated, including a discussion of implicit time stepping and using the multigrid algorithm for the iterative solution of Poisson's equation. Two model problems are analyzed. The first gives insight into the accuracy of the solution to elliptic equations on overlapping meshes. The second deals with the numerical approximation of boundary conditions for vorticity stream function formulations. Computational results are presented.

Table of Contents

List of Figures and Tables

Part I

Part II

Part I: The Numerical Solution of Hyperbolic Systems of Conservation Laws

Introduction

We consider the numerical solution of the initial value problem for hyperbolic systems of conservation laws written in the form

$$\begin{cases} \mathbf{u}_t + \mathbf{f}(\mathbf{u})_x = 0 \\ \mathbf{u}(x,0) = \mathbf{u}_0(x) \end{cases}$$

Such problems can be difficult to solve numerically since the solutions exhibit discontinuities. However, there are many problems which can be cast into the above form and thus it is of some importance to develop good numerical schemes. The approach taken here is to use a hybrid method which combines finite difference methods with the method of characteristics. Finite differences are easy to implement and accurate when the numerical solution is smooth. The method of characteristics is more difficult to implement but is accurate when there are discontinuities present. The idea is to combine the methods, using finite differences where the solution is smooth and using the method of characteristics otherwise. The finite difference method is applied on a fixed grid. The method of characteristics is used on points which move through the fixed grid. The position and number of these *characteristic* points may vary with time. Shocks appear as perfect discontinuities. They are recognized by the crossing of characteristics and are fitted using the shock relations. Interactions between different shocks are handled in a uniform manner by the use of a Riemann solver.

There are three chapters. Chapter 1 describes the notation that is used and gives background information on the problem. It includes a description of some of the other methods that have been devised to solve the initial value problem. In chapter 2 some of the details of the scheme are presented. The results of some numerical calculations on the equations of gas dynamics are given in the third chapter.

Chapter 1

Background

The background material has been divided into two sections. The first section discusses some of the theoretical background associated with hyperbolic systems of conservation laws. In addition some notation is introduced and some definitions are made. The second section is mainly concerned with giving an overview of the various other numerical schemes that have been devised.

1.1 Theoretical

A system of hyperbolic conservation laws in one space dimension can be written in the form†

$$\mathbf{u}_t + \mathbf{f}(\mathbf{u})_x = 0 \tag{1}$$

Here $\mathbf{u} : \mathbb{R} \times [0, \infty) \rightarrow \mathbb{R}^m$ is a vector with m components, each component being a real valued function of x and t. $\mathbf{f}$ is called the flux function, $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^m$. In terms of their components $\mathbf{u}$ and $\mathbf{f}$ will be written

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} \qquad \mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix}$$

The transpose of a vector $\mathbf{u}$ will be denoted as $\mathbf{u}^T$. The system (1) is said to be hyperbolic if the eigenvalues $\{c_i(\mathbf{u})\}_{i=1}^m$ of the Jacobian matrix

$$J(\mathbf{u}) = \mathbf{f}_\mathbf{u} = \left[\frac{\partial f_i}{\partial u_j}\right]$$

† Within each section the equations are numbered consecutively beginning with (1). When reference is made to an equation in another section the section number is prepended to the equation number. Thus equation (2.5.2) refers to equation (2) of section 2.5.

are real and there is a complete set of eigenvectors. It will be assumed here that the eigenvalues are distinct and can be ordered

$$c_1 < c_2 < c_3 < \cdots < c_m$$

A simple example of a nonlinear hyperbolic equation is the scalar equation

$$u_t + (\frac{1}{2}u^2)_x = 0 \tag{2}$$

or

$$u_t + uu_x = 0$$

Notice that along the line $dx/dt = u$ (the characteristic curve) the equation reduces to the ordinary differential equation $du/dt = 0$. Similarly in the case of the system of equations (1) there are curves in $x - t$ space along which the partial differential equations reduce to ordinary differential equations. Such curves are called characteristics and the resulting equations are called the characteristic equations. These characteristic equations can be determined as follows. Let $\mathbf{a}_i(\mathbf{u})$ denote the left eigenvector of J corresponding to the eigenvalue c_i.

$$\mathbf{a}_i(\mathbf{u}) = \begin{bmatrix} a_{i1}(\mathbf{u}) \\ a_{i2}(\mathbf{u}) \\ \vdots \\ a_{im}(\mathbf{u}) \end{bmatrix}$$

The eigenvector satisfies the eigenvalue equation

$$\mathbf{a}_i^T J = c_i \mathbf{a}_i^T \tag{3}$$

Multiplying the conservation equation (1) by $\mathbf{a}_i(\mathbf{u})^T$ and using the eigenvalue equation gives

$$\mathbf{a}_i^T [\frac{\partial \mathbf{u}}{\partial t} - c_i(\mathbf{u}) \frac{\partial \mathbf{u}}{\partial x}] = 0$$

Each of these equations reduces to an ordinary differential equation along the characteristic curve C_i whose slope in $x - t$ space is $c_i(\mathbf{u})$.

$$\mathbf{a}_i^T \frac{d\mathbf{u}}{dt} = 0 \qquad \text{along} \quad C_i : \frac{dx}{dt} = c_i(\mathbf{u}) \quad i = 1, 2, ..., m \tag{4}$$

or written out in components

$$\sum_{j=1}^{m} a_{ij}(\mathbf{u})\frac{du_j}{dt} = 0 \qquad \text{along} \quad C_i : \frac{dx}{dt} = c_i(\mathbf{u}) \quad i = 1, 2, ..., m$$

These are the desired characteristic equations.

If $\mathbf{f}$ is a nonlinear function of $\mathbf{u}$ then in general classical solutions to the initial value problem do not exist for all time. Derivatives of $\mathbf{u}$ can become infinite in a finite time even for smooth initial data. Often systems such as (1) describe the limiting behaviour of a physical process as some parameter goes to zero. For example the equations of gas dynamics to be discussed later are the limiting equations as the effects of viscosity and heat conduction go to zero. The breakdown of the solution may then be related to the breakdown of some of the assumptions under which the equations were derived. To obtain the physically meaningful solution one could solve a new set of equations which includes those effects that are now important. For example one often really wants the solution to a related viscous problem

$$\mathbf{u}_t + \mathbf{f}(\mathbf{u})_x = \epsilon(\beta(\mathbf{u})\mathbf{u}_x)_x \qquad \beta(\mathbf{u}) \geq 0$$

as the viscosity ϵ tends to zero. Solving these equations accurately can be much more work since one must resolve the shocks. In many cases the structure through the shock is not required. As an alternative it is possible to patch up the current set of equations by extending the notion of what is meant by a solution. This is done by allowing the solution to have discontinuties. At a propagating discontinuity, on either side of which the solution is continuously differentiable, one can appeal to the integral form of the conservation laws to obtain the equations which describe how the discontinuity or *shock* is to be propagated. These are the Rankine- Hugoniot shock relations

$$[\mathbf{f}] = U[\mathbf{u}] \tag{5}$$

$$[\mathbf{f}] \equiv (\mathbf{f}(\mathbf{u}_R) - \mathbf{f}(\mathbf{u}_L))$$

$$[\mathbf{u}] \equiv (\mathbf{u}_R - \mathbf{u}_L)$$

U is the speed of propagation of the discontinuity. $\mathbf{u}_R$ and $\mathbf{u}_L$ are the states to the right and left of the shock. One way to mathematically define a solution to (1) which allows for discontinuities is to introduce the concept of a generalized solution. We call $\mathbf{u}$ a generalized or weak solution of the system (1) with initial conditions $\mathbf{u}(x,0) = \mathbf{u}_0(x)$ if for all smooth test functions $\phi(x,t)$ of compact support

$$\int_{t=0}^{\infty}\int_{x=-\infty}^{\infty}[\mathbf{u}\phi_t + \mathbf{f}(\mathbf{u})\phi_x]dxdt - \int_{x=-\infty}^{\infty}\mathbf{u}_0(x)\phi(x,0)dx = 0 \qquad (6)$$

This expression can be formally obtained in the following manner. Multiply the conservation equation (1) by ϕ and integrate over time and space. Integrate by parts to remove the derivatives from $\mathbf{u}$ and $\mathbf{f}$ and place them onto ϕ. This gives equation (6). Any classical solution of the conservation equation will thus be a generalized solution. The converse of this statement is not true. Having extended the solution space in this manner we run into the trouble that too many solutions are now allowed. We must use other criteria to determine which weak solution is the physically relevant one. This extra condition is called the entropy condition. For our purposes the entropy condition is simply the geometrical statement that the characteristics on either side of the discontinuity must run into (and not out of) the discontinuity. This means that for some index j

$$c_j(\mathbf{u}_L) > U > c_j(\mathbf{u}_R) \qquad (7a)$$

We further require that not too many characteristics run into the discontinuity so that

$$c_j(\mathbf{u}_L) > U > c_{j-1}(\mathbf{u}_L) \qquad (7b)$$

and

$$c_{j+1}(\mathbf{u}_R) > U > c_j(\mathbf{u}_R) \qquad (7c)$$

These conditions ensure that there are the correct number of equations to determine the evolution of the discontinuity. A propagating discontinuity satisfying the

entropy condition (7) will be called a shock. If a discontinuity satisfies (7) with the inequalities replaced by equalities then it is called a *contact discontinuity*. There are alternative ways to define an entropy condition in terms of an entropy function, Lax [1972].

There are a number of good references for further details of the material presented here, for example Lax [1972] and Whitham [1974].

1.2 Numerical

1.2.1 Introduction

In this section we outline some of the numerical schemes that have been devised to solve nonlinear systems of conservation laws. We only discuss methods which suppose that the detailed structure of the solution through shocks is not required. If the shock profiles are of interest then the method of choice might be an adaptive type algorithm such as those developed by Berger [1982], Brown [1982] or Brackbill and Saltzman [1982], in which grid points are positioned so as to resolve rapid variations in the solution. Such methods often are trying to solve the viscous problem which is related to the conservation laws. Methods which use many grids points to resolve shocks suffer from further complications introduced by time stepping restrictions. The objective of the methods described here is instead to generate sharp shock profiles with as few grid points as possible. Since the solutions to be computed are only weak solutions possessing discontinuities and since the conservation equation does not specify which weak solution is required, some care is required in the choice of numerical method. There are a number of ways to classify the myriad of methods that have been developed. One classification divides the methods into the the two groups of *shock tracking* and *shock capturing* schemes. Shock tracking methods partition the computational region into intervals separated by shocks, or other discontinuities. The shocks are propagated using the jump conditions. The

solution between the shocks is calculated using the partial differential equations, using a standard finite difference scheme for example. The basics of shock tracking are discussed in Richtmeyer and Morton [1967]. A good discussion of shock tracking and more generally interface tracking can be found in Hyman [1984]. Other references of interest are Plohr, Glimm and McBryan [1983], Lötstedt [1982] and Ni and Wu [1982]. The shock capturing schemes do not explicitly propagate shocks but rather use a method that will behave correctly when a shock is present. There are basically two categories of shock capturing methods, those methods which use a Riemann solver as a basic building block and those methods which do not.

1.2.2 Shock Capturing Methods

In the basic finite difference approach to the solution of partial differential equations the domain is covered by a grid or mesh. In one space dimension the computational interval is often discretized into a set of grid points $\{x_i\}$ with the grid spacing $\Delta x_i = x_i - x_{i-1}$ either constant or varying in such a way so as to put more grid points where they are needed. The partial derivatives in the PDE are replaced by finite differences. These difference approximations are accurate to some order in the mesh spacing, for functions which are smooth with respect to the chosen mesh. Provided the scheme is stable (at least for the linearized problem) and there are enough grid points to resolve the solution we expect to obtain a numerical solution which closely approximates the true one. The application of this procedure to the solution of nonlinear systems of conservation laws can lead to disastrous results. First of all the solutions are not smooth with respect to any mesh in which the mesh spacings are restricted to be greater than zero. Second, there is no reason to believe that the numerical solution will converge to the physically desired weak solution as opposed to other possible weak solutions. Consider for example the

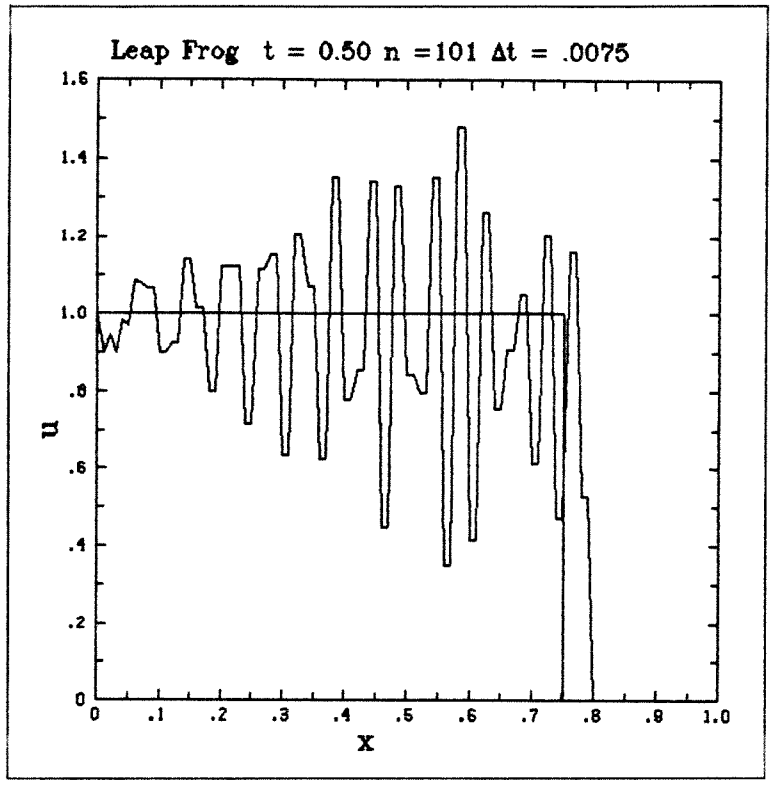


Figure 1.1 Leap frog solution

numerical solution of the scalar problem

$$\begin{aligned} u_t + f(u)_x &= 0 \\ f(u) &= \tfrac{1}{2}u^2 \end{aligned} \tag{1}$$

on the interval $[0,1]$ with piecewise constant initial data

$$u(x,0) = \begin{cases} 1 & x < .5 \\ 0 & x > .5 \end{cases}$$

A common method for the solution of hyperbolic PDE's is the leap frog scheme which in this case takes the form

$$v_i^{n+1} = v_i^{n-1} - \frac{\Delta t}{\Delta x}(f(v_{i+1}^n) - f(v_{i-1}^n))$$

where v_i^n is an approximation to $u(i\Delta x, n\Delta t)$. For simplicity a constant mesh size Δx will be assumed. The method is two levels in time and so requires extra initial conditions. The true solution at times 0 and Δt are used to get the procedure started. In figure 1.1 the numerical solution from leap frog is plotted against the true (weak) solution at time $t = .5$. (The true solution is a step function.)

Leap frog is clearly not giving a satisfactory answer. The oscillation which is present is a common numerical artifact for many difference schemes. Obtaining an accurate scheme without such oscillations is the main goal of those attempting to devise good schemes. Leap frog behaves badly despite the fact that it is conservative in the sense that (neglecting boundaries)

$$\sum_i v_i^{n+1} = \sum_i v_i^{n-1}$$

For one level schemes the term *conservation form* refers to methods which can be written in the form

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^n - \frac{\Delta t}{\Delta x} D_+ \mathbf{g}(\ldots \mathbf{u}_{i-1}^n, \mathbf{u}_i^n, \mathbf{u}_{i+1}^n, \ldots) \tag{2}$$

$\mathbf{g}$ is the numerical flux function which depends on the solution at various grid points. D_+ is the forward divided difference operator. We will also use the backward divided difference operator D_-.

$$D_+ u_i = \frac{u_{i+1} - u_i}{\Delta x}$$
$$D_- u_i = \frac{u_i - u_{i-1}}{\Delta x}$$

Consistency of the method implies that $\mathbf{g}(\ldots, \mathbf{u}, \mathbf{u}, \mathbf{u}, \ldots) = \mathbf{f}(\mathbf{u})$. Such a scheme possesses the conservation property

$$\sum_{i=p}^{q} \mathbf{u}_i^{n+1} = \sum_{i=p}^{q} \mathbf{u}_i^n - \frac{\Delta t}{\Delta x}(\mathbf{g}(\ldots, \mathbf{u}_{q+1}, \ldots) - \mathbf{g}(\ldots, \mathbf{u}_{p-1}, \ldots))$$

This property of conservation is often sought in a difference scheme. A good reason for this stems from a theorem due to Lax and Wendroff [1960].

Theorem (Lax and Wendroff). *Assume that as Δx and Δt tend to zero with $\lambda = \Delta t/\Delta x$ fixed, the solution to a consistent scheme in conservation form (2) converges boundedly almost everywhere to some function $\mathbf{u}(x,t)$. Then $\mathbf{u}(x,t)$ is a weak solution of*

$$\begin{cases} \mathbf{u}_t + \mathbf{f}(\mathbf{u})_x = 0 \\ \mathbf{u}(x,0) = \mathbf{u}_0(x) \end{cases} \tag{3}$$

One of the first approaches to the problem of obtaining a shock profile without the unphysical oscillations is the method of ***artificial viscosity*** , Von Neumann and Richtmeyer [1950], Lax and Wendroff [1960], and Lapidus [1967]. The artificial viscosity proposed by Lapidus can be added on to the solution u_i^{n+1} determined by other methods. This results in a corrected solution $\tilde{u}_i^{n+1}$

$$\tilde{u}_i^{n+1} = u_i^{n+1} + \nu(\frac{\Delta t}{\Delta x})(\Delta x)^3 D_-[|D_- u_{i+1}^{n+1}| D_- u_{i+1}^{n+1}]$$

In smooth parts of the flow this artificial viscosity is $O(\Delta t(\Delta x)^2)$ provided that the parameter ν is $O(1)$. However, where the solution varies rapidly it has the effect of smoothing the solution. This is suggested by the observation that the artificial viscosity step is a fractional step in the solution of the diffusive equation

$$u_t = \nu(\frac{\Delta t}{\Delta x})(\Delta x)^3[|u_x|u_x]_x$$

Boris and Book [1973] developed a flux corrected transport algorithm (FCT) to try and keep shocks sharp. Their two stage method consists of a transport (convective) stage followed by an antidiffusive stage. The antidiffusive step essentially involves solving the backwards heat equation for one step. Thus the correction to the first stage is of the form

$$\begin{aligned} \tilde{u}_i^{n+1} &= u_i^{n+1} - \nu(\Delta x)^2 D_+ D_- u_i^{n+1} \\ &= u_i^{n+1} - \nu(\Delta x)^2 D_+ f_{i-\frac{1}{2}} \end{aligned} \tag{4}$$

$$\text{where} \quad f_{i-\frac{1}{2}} = D_- u_i^{n+1}$$

The ient is to remove the smoothing introduced by the transport step. The changes introduced by the antidiffusion are restricted so as to not introduce or accentuate extrema. Thus instead of the flux given in (4) a corrected flux $f_{i+\frac{1}{2}}^c$ is used.

$$\tilde{u}_i^{n+1} = u_i^{n+1} - \nu\Delta x^2 D_+ f_{i-\frac{1}{2}}^c$$

$$f_{i+\frac{1}{2}}^c = \text{sgn}\Delta_{j+\frac{1}{2}} \max(0, \min(\Delta_{j-\frac{1}{2}} \text{sgn}\Delta_{j+\frac{1}{2}} \nu|\Delta_{j+\frac{1}{2}}|, \Delta_{j+3/2} \text{sgn}\Delta_{j+\frac{1}{2}}))$$

$$\Delta_{j+\frac{1}{2}} = D_+ u_i^{n+1}$$

$$\text{sgn}x = \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

Further work on the flux corrected transport algorithm was performed by Zalesak [1979]. The algorithm is stated in a more general form in which it closely resembles the hybridization technique of Harten and Zwass. The hybrid method utilizes a low order scheme which hopefully given monotone shock profiles together with a high order scheme. The hybrid scheme switches from the high order method which is used when the solution is smooth to the low order method when the solution varies rapidly. The details of the switching distinguishes the flux corrected transport algorithm from the hybrid method. Harten [1977] developed the artificial compression method (ACM) to prevent the smearing of contact discontinuities and improve the resolution of shocks. Contact discontinuities are especially difficult to handle numerically since unlike shocks where the characteristics point into each other, at contacts the characteristics are parallel. In the scalar case the following equation is solved

$$u_t + (f(u) + g(u,t))_x = 0$$

where the artificial compression function $g(u,t)$ is chosen to cause the characteristics to point into contact discontinuities and to point more sharply into shocks.

Another approach which has met with some success has been the *upwind difference scheme*. For the scalar problem (1) the basic upwind difference scheme takes the form

$$v_i^{n+1} = v_i^n - \Delta t \begin{cases} D_- f_i^n & \text{if } f'(v_i) > 0 \\ D_+ f_i^n & \text{if } f'(v_i) < 0 \end{cases}$$

The upwind difference approach is essentially an approximate method of characteristics; the spatial differencing is taken in the direction from which the characteristics are propagating (upwind). For the scalar problem, the main difference between the upwind schemes is the manner in which the formulas switch at a sonic point ($f'(u) = 0$). The Engquist Osher scheme is one approach, and is given by

$$u_i^{n+1} = u_i^n - \Delta t(D_+ f_-(u_i^n) + D_- f_+(u_i^n))$$

where

$$f_+(u) = \int_0^u \chi(s) f'(s) ds$$

$$f_-(u) = \int_0^u (1 - \chi(s)) f'(s) ds$$

$$\chi(u) = \begin{cases} 1 & \text{if } f'(u) > 0 \\ 0 & \text{if } f'(u) \leq 0 \end{cases}$$

In this definition it is assumed that f has been transformed so that $f(0) = 0$ and hence $f = f_+ + f_-$. If f' is of one sign in a region the method reduces to an upwind scheme. The Engquist Osher scheme has the desirable feature that it is conservative, while the simple upwind scheme is not. A major difficulty with the upwind difference schemes is the manner in which they are extended to systems. If all eigenvalues of the Jacobian matrix $\mathbf{f_u}$ are of one sign the extension is straightforward. If, however, the eigenvalues are of different signs then information is propagating in both directions and there is no longer a single upwind direction. Extensions to systems thus requires some care, Engquist and Osher [1980], Osher and Solomon [1980], and Harten, Lax and van Leer [1983].

The Scalar Problem

Much more is known about weak solutions to the scalar conservation law than solutions to systems of conservation laws. An understanding of the scalar equation is helpful to understand the more general case. Many methods are initially developed for the scalar problem. There are then ways to extend the method to systems. It is probably fair to say, however, that many methods do not work as well for systems.

Consider the scalar nonlinear hyperbolic initial value problem.

$$u_t + f(u)_x = 0$$

$$u(x, 0) = u_0(x)$$

The equation indicates that $u = constant$ along the curves $dx/dt = a(u)$ where $a(u) = df/du$. The characteristics are thus straight lines. The solution breaks

down when these characteristics cross. The weak solution of interest is often the limit solution of a related viscous equation as the viscosity ϵ tends to zero.

$$u_t + f(u)_x = \epsilon(\beta(u)u_x)_x \qquad \beta(u) > 0$$

Weak solutions of the scalar conservation law possess a monotonicity property in the sense that local maxima (minima) do not become larger (smaller) and no new extreme points can be generated. These weak solutions are total variation non-increasing (TVNI) in time. The variation of a function u of compact support can be defined as

$$TV(u) = \sup_P \sum_i |u(a_i) - u(a_{i-1})|$$

where P is the set of all partitions $\{a_i\}$ of the interval. Thus we have that a weak solution u satisfies $TV(u(t_2)) \leq TV(u(t_1))$ for all t_2 greater than t_1.

Since the weak solutions satisfy these monotonicity properties it seems like a good idea to try and develop numerical schemes which possess similar properties. Reference for example Harten, Hyman, and Lax [1976], and Harten [1977]. Consider a finite difference scheme

$$v_j^{n+1} = H(v_{j-k}^n, v_{j-k+1}^n, \ldots, v_{j+k}^n)$$

which can also be written in the form

$$\mathbf{V}^{n+1} = L\mathbf{V}^n$$

where $\mathbf{V}$ is the vector with components v_j. The scheme is said to be *monotone* if H is an increasing function of its arguments. The method is ***monotonicity preserving*** if $\mathbf{V}^{n+1}$ is monotone whenever $\mathbf{V}^n$ is. The scheme is ***total variation nonincreasing*** if $TV(L\mathbf{V}) \leq TV(\mathbf{V})$ where the total variation for a discrete function $\mathbf{V}$ is defined to be

$$TV(\mathbf{V}) = \sum_i |v_{i+1} - v_i|$$

Harten has shown that monotone schemes are TVNI and that TVNI schemes are monotonicity preserving. It was shown in Harten, Hyman and Lax [1976] that solutions to monotone schemes in conservation form converge to the physically correct weak solution. These schemes solve a viscous equation

$$u_t + f(u)_x = \Delta t(\beta(u,\lambda)u_x)_x \qquad \beta(u,\lambda) \geq 0$$

to *second order accuracy.* The diffusion term involving β is generated from the truncation error terms in the scheme. Hence the overall order of accuracy is only first order. Second order accurate TVNI schemes have been developed by Harten [1982]. They are necessarily nonlinear in nature since linear TVNI schemes are at most first order accurate. Harten's method is in the spirit of a defect correction. The solution given by a first order accurate scheme solves a modified conservation law to second order accuracy.

$$u_t + (f(u) - \Delta t\, g(u,\lambda))_x = 0$$

$$g(u,\lambda) = \beta(u,\lambda)u_x$$

Thus if one solves instead the equation

$$u_t + (f(u) + \Delta t\, g(u,\lambda))_x = 0$$

with a TVNI scheme, then the resulting scheme, which has a spatial bandwidth of 5, is second order accurate where the solution is smooth. The method is thus seen to be similar to the anti-diffusion schemes.

One difficulty in this whole line of approach seems to be the fact that solution to systems of conservation laws do not possess these monotonicity properties. New extrema can be generated. (See for example the shock interaction problem which is solved in the section on numerical results.) To force the solutions to systems to be monotone is incorrect. Numerical schemes, however, tend to generate spurious

wiggles (local maxima and minima) which one does want to remove. Hopefully the application of schemes, which are monotonicity preserving or TVNI for scalar problems, to the solution of systems will give results without as many spurious wiggles. It would seem that care must be taken in enforcing monotonicity constraints in the system case as is done in a number of algorithms.

Methods Based on Riemann Solvers

A major class of shock capturing methods are based upon a Riemann solver. The first method of this type was developed by S.K. Godunov [1959]. In Godunov's method the solution at each time step is approximated by a piecewise linear function.

$$\mathbf{v}^n(x) = \mathbf{v}^n_{i+\frac{1}{2}} \qquad \text{for} \quad x_i < x < x_{i+1}$$

At grid point x_i the solution jumps from $\mathbf{v}_{i-\frac{1}{2}}$ to $\mathbf{v}_{i+\frac{1}{2}}$. Let $\mathbf{w}_i(x,t)$ denote the solution to the Riemann problem which has this same jump for initial conditions.

$$(\mathbf{w}_i)_t + f(\mathbf{w}_i)_x = 0$$

$$\mathbf{w}_i(x,t) = \begin{cases} \mathbf{v}^n_{i-\frac{1}{2}} & \text{for } x < x_i \\ \mathbf{v}^n_{i+\frac{1}{2}} & \text{for } x > x_i \end{cases}$$

For small enough times the solution to the initial value problem with initial condition $\mathbf{v}^n(x)$ can be constructed as a union of the solutions $\mathbf{w}_i$.

$$\mathbf{w}(x,t) = \mathbf{w}_i(x,t) \qquad \text{for} \quad x_{i-\frac{1}{2}} < x < x_{i+\frac{1}{2}} \tag{5}$$

The time restriction is determined by the condition that the solutions from the different Riemann problems do not interact.

$$\frac{\Delta t}{\Delta x} \max(c_i) < \frac{1}{2} \tag{6}$$

The approximate solution at time $t + \Delta t$ is then defined as a piecewise constant function whose value over the interval (x_i, x_{i+1}) is taken as the average of $\mathbf{w}(x, t + \Delta t)$.

$$\mathbf{v}^{n+1}_{i+\frac{1}{2}} = \frac{1}{\Delta x} \int_{x_i}^{x_{i+1}} \mathbf{w}(x, t + \Delta t) dx$$

This expression can be rewritten in the standard form of a difference approximation by using the fact that $\mathbf{w}(x,t)$ is composed of solutions to Riemann problems. The integral simplifies to

$$\mathbf{v}_{i+\frac{1}{2}}^{n+1} = \mathbf{v}_{i+\frac{1}{2}}^{n} - \frac{\Delta t}{\Delta x}(\mathbf{f}(\mathbf{w}(x_{i+1}, t+\Delta t)) - \mathbf{f}(\mathbf{w}(x_i, t+\Delta t)))$$

Since $\mathbf{w}(x_i, t+\Delta t)$ only depends on $\mathbf{w}_{i-\frac{1}{2}}$ and $\mathbf{w}_{i+\frac{1}{2}}$ it can be seen that Godunov's method can be written in conservation form.

Godunov's method by itself tends to smooth out the solution quite a bit and is only first order accurate Richtmeyer [1967]. The method can be extended to higher order. Reference, for example, the work of van Leer [1979] and the *piecewise parabolic method* of Woodward and Colella [1984]. A variation on Godunov's method was created by Glimm [1965] and is known as Glimm's method or the random choice method. At each time level the solution is taken to be piecewise constant. The grid is staggered from one step to the next; the positions of the possible discontinuities being either at the grid points x_i or the half grid points $x_{i+\frac{1}{2}}$. Suppose at time $t = n\Delta t$ the solution is given by the piecewise constant function $\mathbf{v}^n$

$$\mathbf{v}^n(x) = \mathbf{v}_{i+\frac{1}{2}}^{n} \qquad \text{for} \quad x_i \le x \le x_{i+1}$$

As in Godunov's method the exact solution to the conservation laws for the initial condition $\mathbf{w}(x,t) = \mathbf{v}^n(x)$ can be obtained by solving a sequence of Riemann problems. This exact solution is $\mathbf{w}(x,t)$ as given by (5) and (6). The solution at time $t+\Delta t$ is given by a piecewise constant function with discontinuities at $x_{i+\frac{1}{2}}$

$$\mathbf{v}^{n+1}(x) = \mathbf{v}_i^{n+1} \qquad \text{for} \quad x_{i-\frac{1}{2}} \le x \le x_{i+\frac{1}{2}}$$

The solution value in each interval is given by $\mathbf{v}_i^{n+1} = \mathbf{w}(\tilde{x}_i, t+\Delta t)$ where $\tilde{x}_i$ is chosen at random in $(x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}})$. Glimm was able to obtain some weak convergence results for his method in one space dimension for initial data which are close to a

constant state [1965]. The method was adapted for practical use by Chorin [1976] for use in the problems of gas dynamics.

Since much of the information from the Riemann problem is lost in the averaging step of Godunov's method Roe [1981] suggested the use of approximate Riemann solvers . Instead of solving the full nonlinear Riemann problem Roe solves a linearized version of the equations. The linearization depends on the states $\mathbf{u}_L$ and $\mathbf{u}_R$ and is chosen to have certain desirable features. Thus instead of solving a nonlinear Riemann problem one can solve a linear problem of the form

$$\mathbf{u}_t + \tilde{A}(\mathbf{u}_L, \mathbf{u}_R)\mathbf{u}_x = 0$$

The matrix $\tilde{A}$ is chosen to have property U, which means it satisfies the four conditions

(i) $\tilde{A}$ is a linear mapping

(ii) as $\mathbf{u}_L \to \mathbf{u}_R \quad \tilde{A}(\mathbf{u}_L, \mathbf{u}_R) \to \mathbf{f}_\mathbf{u}$

(iii) for all $\mathbf{u}_L$ and $\mathbf{u}_R \quad \tilde{A}(\mathbf{u}_L, \mathbf{u}_R)(\mathbf{u}_L - \mathbf{u}_R) = \mathbf{f}_\mathbf{u}(\mathbf{u}_L) - \mathbf{f}_\mathbf{u}(\mathbf{u}_R)$

(iv) The eigenvectors of $\tilde{A}$ are linearly independent.

This linear problem is much easier to solve since no iterations are required.

Chapter 2

Description of the Scheme

The basic idea of the scheme has already been outlined in the introduction. In this chapter we proceed to give a more complete description. In section 2.1 the structure of the computational grid is discussed. By computational grid we refer to an underlying fixed grid together with extra grid points, called *characteristic points*. The number of and positions of these extra points varies with time. The concept of a *group* of characteristic points is introduced and explained. Later sections describe how these groups of points are advanced in time. This involves a discussion of the method of characteristics, shock fitting, shock interactions and the Riemann problem.

2.1 Grid Structure

The computational grid first consists on an underlying fixed grid. Points on the fixed grid will be denoted by x_i and the corresponding solution values by $\mathbf{u}(i)$. For simplicity this grid is taken to have a constant mesh spacing h so that $x_{i+1} = x_i + h$. In addition to this uniform mesh there are also some extra points which move through the fixed grid as the solution develops. These extra points will be called *characteristic points* since they will be the points where the method of characteristics is applied. They will be located in regions where the solution is not smooth such as around discontinuities. Denote their positions by $x_c(i)$. The solution value at $x_c(i)$ will be called $\mathbf{u}_c(i)$. The characteristic points are not restricted to lie on any grid. Their positions are variable. In fact, when a shock develops two characteristic points will occupy the same location. One of these points will carry the information for the state to the left of the shock and the other holds the state to the right of

the shock. This allows shocks to be represented as true discontinuities. Figure 2.1 shows what the computational grid and a component of the solution might look like. The solution is the *shock tube* example of section 3.2 and consists of two discontinuities, a shock to the right and a contact discontinuity in the center. There is a expansion fan to the left. The characteristic points and their solutions values are marked with circles. Notice that there are two characteristic points located at the position of the shock and at the position the contact discontinuity.

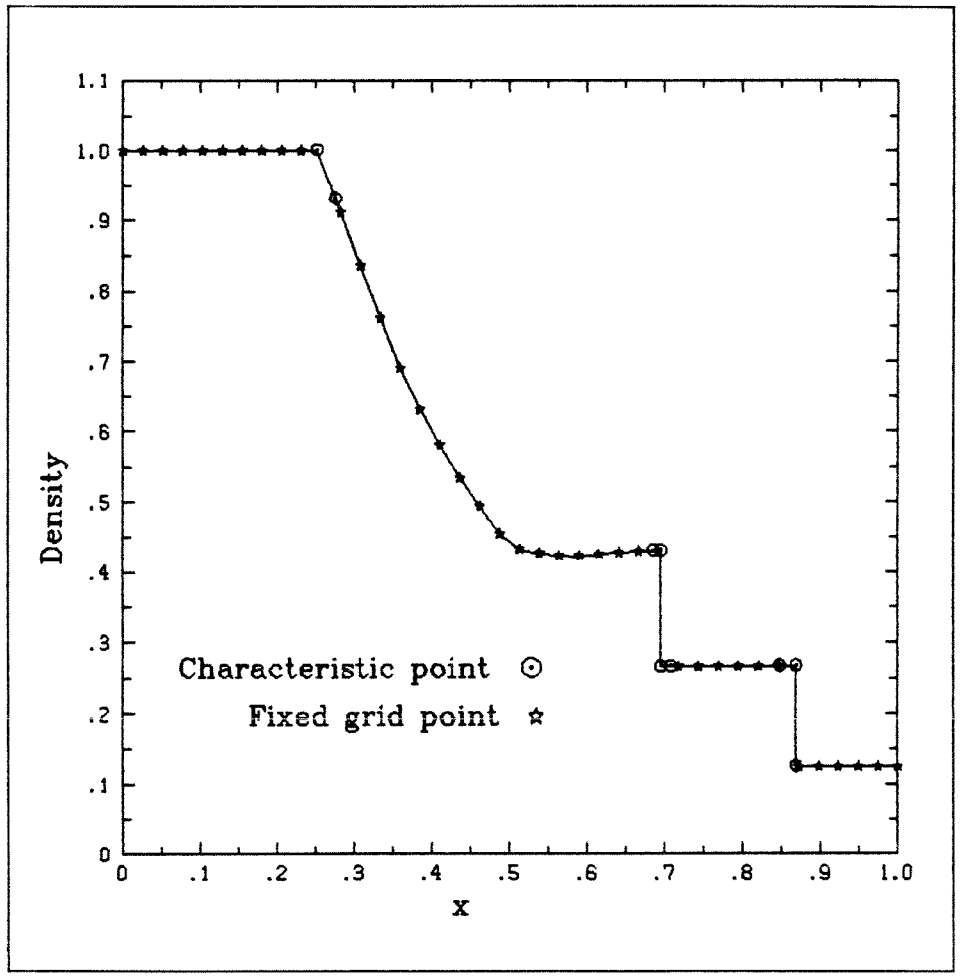


Figure 2.1 Grid Structure

Much of the complexity of the method is associated with the problem of keeping track of the characteristic points as they move through the fixed grid. Here is an outline of how this problem was resolved. Characteristic points will tend to cluster

in certain locations, such as around shocks or around discontinuities in the first derivative. (In figure 2.1 there is a discontinuity in the first derivative at the edge of the expansion fan.) Such a cluster of characteristic points will be identified as a logical entity and will be called a *group*. Each group is separated from other groups by a smooth portion of the solution. Let $\mathbf{g}_i$ denote the i^{th} group. It consists of a number of characteristic points

$$\mathbf{g}_i = \left\{ x_c^i(j) \qquad j = 1, \ldots, n_i \right\}$$

A superscript i has been added to the characteristic point to denote that it belongs to group i. The smallest and largest values of $x_c^i(j)$ in any group define the extent of the group. Fixed grid points which lie *underneath* a group, that is within the extent of the group, will not be used. Such points will be called *inactive*, as opposed to *active* fixed grid points which lie under no group. In figure 2.1 there are three groups, one around the shock, a second at the contact and the third around the corner of the expansion fan.

To advance the solution from one time step to the next the method of characteristics is applied to each group separately. Solution values outside the extent of the group may be needed. These can be obtained from neighbouring points on the fixed grid. The numerical implementation of the method of characteristics is described in section 2.3. The fixed grid points are advanced using a finite difference method which is described in section 2.2. The computational grid is monitored to make sure that all groups and all active fixed grid points satisfy certain conditions. The groups can be constantly changing in size and position. The following lists some of the operations that may be applied to a group or pair of groups.

(i) Merging - if two groups are too close together they are joined to form one new and larger group.

(ii) Splitting - if there is a smooth region in the interior of a group, the group

is separated into two groups. Points on the fixed grid which lie in this smooth region become active points.

(ii) Liquidation - groups may disappear if the solution becomes smooth at all the points which make up the group.

(iv) Trimming - if there is a smooth region on the end of a group characteristic points will be taken away.

(v) Addition - Two adjacent characteristic points in a group are not allowed to get too far apart. Extra points may be added to prevent this from happening.

(vi) Creation - when the solution on the fixed grid becomes ***rough*** a new group may form. (Described in more detail below.)

New groups may appear spontaneously when the solution becomes ***rough***. The smoothness of the solution on the fixed grid is measured by a normalized second undivided difference quotient. A point i on the fixed grid will become the location of a new a characteristic point if

$$\max_{1\le j\le m} \frac{|u_j(x_{i+1}) - 2u_j(x_i) + u_j(x_{i-1})|}{\|u_j\|} > \delta \qquad (1)$$

Here δ is a predetermined constant which will depend on h. $u_j(x_i)$ is the j^{th} component of the solution at position x_i of the fixed grid and $\|u_j\|$ is a global measure of the size of the j^{th} component of the solution. The left hand side of (1) will be $O(h^2)$ where the numerical solution is smooth with respect to the grid. If this quantity becomes large compared to h^2 then the finite difference method is likely losing accuracy and it is time to switch to the method of characteristics. There are alternative ways to define a measure of the smoothness, see for example the discussion in Berger [1982]. The measure that one uses should be related to the accuracy of the finite difference method which is being used. A similar measure to (1) is used to determine when the numerical solution within a group is becoming smooth.

Programming is considerably simplified when the the correct data structures are used. The data structure for holding the groups is straightforward in nature, consisting of pointers and lists. It is convenient to keep the characteristic points ordered by their position so that neighbours are easily found. A useful array to have, which simplifies many group operations, is one which indicates the *status* of each point on the fixed grid. The status will indicate whether the point is active or not and for inactive points will indicate which group it lies underneath of. Denoting this array by $istatus(i)$, say, then

$$istatus(i) = \begin{cases} 0 & \text{if fixed grid point } i \text{ is an active point} \\ k & \text{if point } i \text{ is inactive and sits below group } k \end{cases}$$

Using this array it is easy to check whether two groups are getting close together. Groups are merged when they are less than a few mesh widths apart. In addition the array acts a system of pointers from the fixed grid to the data structure containing the groups.

2.2 The Finite Difference Equations

The numerical solution on the fixed grid is advanced using a finite difference method. The method that is used is the second order Lax Wendroff scheme. When written as a two step process this scheme takes the form

$$\mathbf{u}_{i+1/2}^{n+1/2} = \frac{1}{2}[\mathbf{u}_{i+1}^{n} + \mathbf{u}_{i}^{n}] - \frac{(\Delta t/2)}{h}[\mathbf{f}(\mathbf{u}_{i+1}^{n}) - \mathbf{f}(\mathbf{u}_{i}^{n})]$$

$$\mathbf{u}_{i}^{n+1} = \mathbf{u}_{i}^{n} - \frac{\Delta t}{h}[\mathbf{f}(\mathbf{u}_{i+1/2}^{n+1/2}) - \mathbf{f}(\mathbf{u}_{i-1/2}^{n+1/2})]$$

where

$$\mathbf{u}_{i}^{n} = \mathbf{u}(x_0 + ih, t_0 + n\Delta t)$$

Special forms of the difference equations are used where the fixed grid points meet the characteristic points. Here the grid spacing is not uniform and we use

$$\mathbf{u}_{j+1/2}^{n+1/2} = \frac{1}{2}[\mathbf{u}_{j+1}^{n} + \mathbf{u}_{j}^{n}] - \frac{(\Delta t/2)}{h_j}[\mathbf{f}(\mathbf{u}_{j+1}^{n}) - \mathbf{f}(\mathbf{u}_{j}^{n})] \qquad j = i-1, i$$

$$\mathbf{u}_{i}^{n+1/2} = \frac{1}{2}[\mathbf{u}_{i+1}^{n} + \mathbf{u}_{i-1}^{n}] - \frac{(\Delta t/2)}{h_i + h_{i+1}}[\mathbf{f}(\mathbf{u}_{i+1}^{n}) - \mathbf{f}(\mathbf{u}_{i-1}^{n})]$$

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^n - \frac{\Delta t}{\frac{1}{2}(h_i + h_{i+1})}\left[\frac{h_{i+1}}{h_i}(\mathbf{f}(\mathbf{u}_i^{n+1/2}) - \mathbf{f}(\mathbf{u}_{i-1/2}^{n+1/2})) + \frac{h_i}{h_{i+1}}(\mathbf{f}(\mathbf{u}_{i+1/2}^{n+1/2}) - \mathbf{f}(\mathbf{u}_i^{n+1/2}))\right]$$

where

$$h_i = x_i - x_{i-1}$$

Other difference schemes could be used. Since the scheme is only applied where the solution is smooth, higher order methods might prove to be useful.

2.3 Solving the Characteristic Equations

In this section we discuss the numerical solution of the characteristic equations which were derived in chapter 1.

$$\mathbf{a}_i{}^T \frac{d\mathbf{u}}{dt} = 0 \qquad \text{along} \quad C_i : \frac{dx}{dt} = c_i(\mathbf{u}) \quad i = 1, 2, \ldots, m$$

These equations are a coupled system of nonlinear ordinary differential equations. The system is not, however, in the standard form $d\mathbf{y}/dx = \mathbf{f}(x, \mathbf{y})$. Each characteristic equation only holds upon a curve whose position depends upon the solution. Given the solution $\mathbf{u}$ everywhere at time t the objective is to calculate the solution $\mathbf{u}$ for a particular point $(x, t + \Delta t)$. In the simplest case, when there are no shocks, there will be precisely m characteristic curves which intersect the point $(x, t + \Delta t)$. These characteristics emanate from some (unknown) points (x_i, t). These m characteristics carry enough information to determine the m unknown components of $\mathbf{u}(x, t + \Delta t)$.

To solve the equations numerically we proceed as follows. Suppose we know an approximation to the solution at all grid points at time t. Let $\mathbf{v}(x, t)$ denote the function which equals this solution at each grid point and varies linearly inbetween. Consider the task of determining the solution at some point z at time $t + \Delta t$ by the method of characteristics. To do this the following implicit approximation to the

characteristic equations is solved.

$$\mathbf{a}_i\left(\frac{1}{2}(\mathbf{v}(z,t+\Delta t)+\mathbf{v}(x_i,t)\right)^T(\mathbf{v}(z,t+\Delta t)-\mathbf{v}(x_i,t))=0$$
$$z-x_i=c_i\left(\frac{1}{2}(\mathbf{v}(z,t+\Delta t)+\mathbf{v}(x_i,t))\right)\Delta t \qquad i=1,2,\ldots,m \qquad (1)$$

This will be called the midpoint rule approximation. These nonlinear equations are solved by iteration. There are $2m$ equations for the $2m$ unknowns

$$\begin{cases} \mathbf{v}(z,t+\Delta t) & \text{m unknowns} \\ x_i & i=1,2,\ldots,m \end{cases}$$

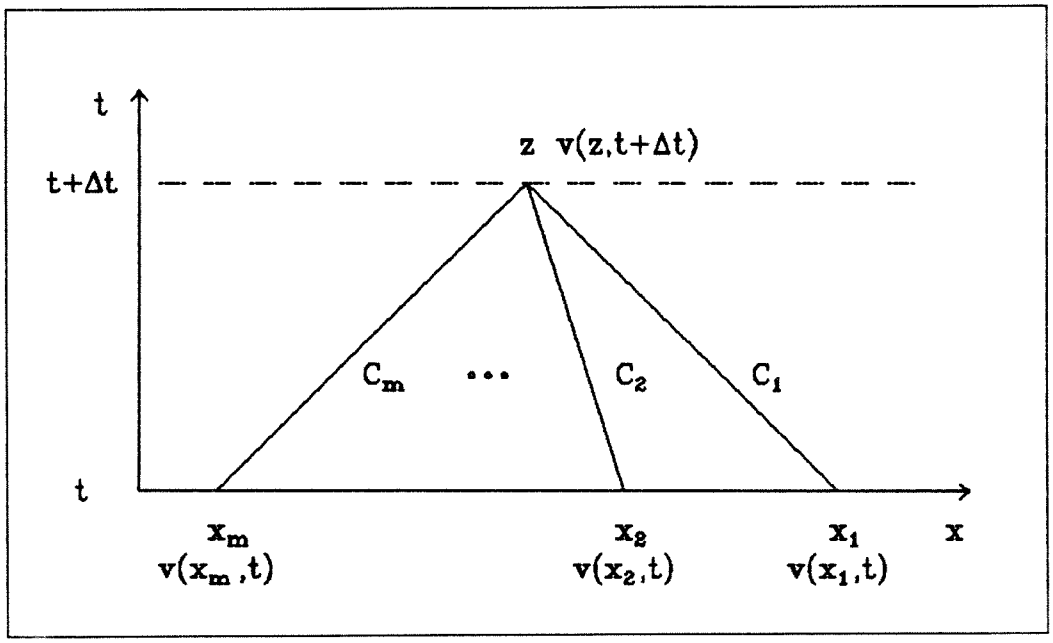


Figure 2.2 Solving the Characteristic Equations

In practice we may also want to solve the same equations when the initial position x_i of one of the characteristics is known and z is unknown. The midpoint rule approximation is a second order accurate approximation to the characteristic equations. Notice that there is not the common type of stability restriction on the ratio of the time step to space step. This restriction is usually called the CFL, Courant-Freidrichs-Lewy, condition. The physical idea behind the CFL condition is that the domain of dependence of the solution to the difference scheme should be

at least as large as the domain of dependence of the actual solution. This results in a restriction of the form $\Delta t/\Delta x \leq constant$. The domain of dependence of the numerical solution of the characteristic equations, however, naturally grows or shrinks depending on the value of Δt.

The the iteration we use to solve the equations (1) is of the following form

$$\mathbf{a}_i\left(\frac{1}{2}(\mathbf{w}^{(k)} + \mathbf{v}(x_i^{(k)}, t))\right)^T(\mathbf{w}^{(k+1)} - \mathbf{v}(x_i^{(k)}, t)) = 0 \tag{2a}$$

$$x_i^{(k+1)} = z - c_i\left(\frac{1}{2}(\mathbf{w}^{(k)} + \mathbf{v}(x_i^{(k)}, t))\right)\Delta t \tag{2b}$$

where $\mathbf{w}^{(k)} \approx \mathbf{v}(z, t + \Delta t)$ and $x_i^{(k)} \approx x_i$ are the k^{th} iterates. We now consider the question of the convergence of this iteration. To simplify the discussion the unknowns x_i will be effectively eliminated. This can be done by assuming that we iterate (2b) to convergence for each $\mathbf{w}^{(k)}$. This defines x_i as a function of $\mathbf{w}^{(k)}$, $x_i = x_i(\mathbf{w}^{(k)})$. For fixed $\mathbf{w}^{(k)}$ it can be shown that the iteration (2b) will converge provided Δt is sufficiently small and c_i and $\mathbf{v}(x, t)$ are smooth enough. (We will think of $\mathbf{v}(x, t)$ as being a smooth function even though in practice it is piecewise linear.)

The problem is now reduced to solving

$$\mathbf{a}_i\left(\frac{1}{2}(\mathbf{w}^{(k)} + \mathbf{v}_i(\mathbf{w}^{(k)}))\right)^T(\mathbf{w}^{(k+1)} - \mathbf{v}_i(\mathbf{w}^{(k)})) = 0 \qquad i = 1, \ldots m$$

where

$$\mathbf{v}_i(\mathbf{w}^{(k)}) = \mathbf{v}(x_i(\mathbf{w}^{(k)}), t)$$

In matrix form these equations are

$$\mathbf{A}(\mathbf{w}^{(k)})\mathbf{w}^{(k+1)} = \begin{bmatrix} \vdots \\ \mathbf{a}_i\left(\frac{1}{2}(\mathbf{w}^{(k)} + \mathbf{v}_i)\right)^T\mathbf{v}_i \\ \vdots \end{bmatrix} \equiv \mathbf{G}(\mathbf{w}^{(k)})$$

or

$$\begin{aligned} \mathbf{w}^{(k+1)} &= \mathbf{A}^{-1}(\mathbf{w}^{(k)})\mathbf{G}(\mathbf{w}^{(k)}) \\ &\equiv \mathbf{F}(\mathbf{w}^{(k)}) \end{aligned} \tag{3}$$

To show the convergence of the iteration (3) we use the contractive mapping theorem. Before stating this theorem we first define $\mathbf{B}_\rho(\mathbf{w}^{(0)})$ to be the ball of radius ρ about $\mathbf{w}^{(0)}$.

$$\mathbf{B}_\rho(\mathbf{w}^{(0)}) = \{\, \mathbf{w} \,|\, \|\mathbf{w} - \mathbf{w}^{(0)}\| \leq \rho \,\}$$

Theorem (Contractive Mapping). *Suppose* $\mathbf{F} : \mathbb{R}^m \to \mathbb{R}^m$ *and that the following two conditions hold*

(i) $\|\mathbf{F}(\mathbf{w}^{(0)}) - \mathbf{w}^{(0)}\| \leq (1-\theta)\rho$ *for* $\rho > 0$ *and* $0 \leq \theta < 1$

(ii) $\|\mathbf{F}(\mathbf{u}) - \mathbf{F}(\mathbf{v})\| \leq \theta\|\mathbf{u} - \mathbf{v}\|$ *for all* $\mathbf{u}, \mathbf{v} \in \mathbf{B}_\rho(\mathbf{w}^{(0)})$

Then the sequence of iterates $\mathbf{w}^{(0)}, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \ldots$ *defined by the iteration* $\mathbf{w}^{(k+1)} = \mathbf{F}(\mathbf{w}^{(k)})$ *converges to a solution of* $\mathbf{w} = \mathbf{F}(\mathbf{w})$. *Moreover, this is the unique solution found in* $\mathbf{B}_\rho(\mathbf{w}^{(0)})$.

Proposition. *Assume that the eigenvectors* $\mathbf{a}_i$ *are linearly independent and that all the functions* $\mathbf{a}_i$, c_i *and* $\mathbf{v}(x,t)$ *are smooth. Then the iteration (3) to solve the characteristic equations will converge provided the time step* Δt *is sufficiently small.*

The initial guess for $\mathbf{v}(z, t+\Delta t)$ will be chosen to be $\mathbf{w}^{(0)} = \mathbf{v}(z,t)$. It follows easily that

$$\mathbf{G}(\mathbf{w}^{(0)}) = \mathbf{A}(\mathbf{w}^{(0)})\mathbf{w}^{(0)} + O(\Delta t)$$

$$\Rightarrow \qquad \mathbf{F}(\mathbf{w}^{(0)}) = \mathbf{w}^{(0)} + O(\Delta t)$$

Whence

$$\|\mathbf{F}(\mathbf{w}^{(0)}) - \mathbf{w}^{(0)}\| \leq K\Delta t$$

for some constant K. Now consider any two vectors $\mathbf{u}$ and $\mathbf{v}$ in $\mathbf{B}_\rho(\mathbf{w}^{(0)})$. Then

$$\mathbf{F}(\mathbf{u}) - \mathbf{F}(\mathbf{v}) = \mathbf{F}_{\mathbf{w}}(\mathbf{w}^{(0)})(\mathbf{u} - \mathbf{v}) + O(\rho\|\mathbf{u} - \mathbf{v}\|)$$

$$\Rightarrow \qquad \|\mathbf{F}(\mathbf{u}) - \mathbf{F}(\mathbf{v})\| \leq \left(c_1\rho + \|\mathbf{F}_{\mathbf{w}}(\mathbf{w}^{(0)})\|\right) \|\mathbf{u} - \mathbf{v}\|$$

We now show that $\|\mathbf{F}_{\mathbf{w}}(\mathbf{w}^{(0)})\| \leq c_2\Delta t$. Differentiating the equation

$$\mathbf{A}(\mathbf{w})\mathbf{F}(\mathbf{w}) = \mathbf{G}(\mathbf{w})$$

and evaluating at $\mathbf{w} = \mathbf{w}^{(0)}$ gives

$$\mathbf{A}_{\mathbf{w}}(\mathbf{w}^{(0)})\mathbf{F}(\mathbf{w}^{(0)}) + \mathbf{A}(\mathbf{w}^{(0)})\mathbf{F}_{\mathbf{w}}(\mathbf{w}^{(0)}) = \mathbf{G}_{\mathbf{w}}(\mathbf{w}^{(0)})$$

Now

$$\mathbf{A}_{\mathbf{w}}(\mathbf{w}^{(0)})\mathbf{F}(\mathbf{w}^{(0)}) = \mathbf{A}_{\mathbf{w}}(\mathbf{w}^{(0)})\mathbf{w}^{(0)} + O(\Delta t)$$

Thus

$$\mathbf{F}_{\mathbf{w}}(\mathbf{w}^{(0)}) = \mathbf{A}_{\mathbf{w}}(\mathbf{w}^{(0)})^{-1}\left(\mathbf{G}_{\mathbf{w}}(\mathbf{w}^{(0)}) - \mathbf{A}_{\mathbf{w}}(\mathbf{w}^{(0)})\mathbf{w}^{(0)}\right)$$

Since the eigenvectors are linearly independent $\|\mathbf{A}_{\mathbf{w}}(\mathbf{w}^{(0)})^{-1}\|$ can be bounded independently of Δt. Thus it will suffice to show that $\mathbf{G}_{\mathbf{w}}(\mathbf{w}^{(0)}) - \mathbf{A}_{\mathbf{w}}(\mathbf{w}^{(0)})\mathbf{w}^{(0)}$ is $O(\Delta t)$. The ij^{th} elements of $\mathbf{A}_{\mathbf{w}}(\mathbf{w}^{(0)})\mathbf{w}^{(0)}$ and $\mathbf{G}_{\mathbf{w}}(\mathbf{w}^{(0)})$ are

$$\left[\mathbf{A}_{\mathbf{w}}(\mathbf{w}^{(0)})\mathbf{w}^{(0)}\right]_{ij} = \left[\frac{\partial \mathbf{a}_i{}^T}{\partial w_j}\right]_{\mathbf{w}^{(0)}} \mathbf{w}^{(0)}$$

and

$$\begin{aligned}\left[\mathbf{G}_{\mathbf{w}}(\mathbf{w}^{(0)})\right]_{ij} &= \left[\frac{\partial}{\partial w_j}\left(\mathbf{a}_i{}^T\mathbf{v}_i(\mathbf{w})\right)\right]_{\mathbf{w}^{(0)}} \\ &= \left[\mathbf{A}_{\mathbf{w}}(\mathbf{w}^{(0)})\mathbf{w}^{(0)}\right]_{ij} + \mathbf{a}_i(\mathbf{w}^{(0)})^T\left[\frac{\partial \mathbf{v}_i}{\partial w_j}\right]_{\mathbf{w}^{(0)}}\end{aligned}$$

But

$$\begin{aligned}\left[\frac{\partial \mathbf{v}_i}{\partial w_j}\right]_{\mathbf{w}^{(0)}} &= \left[\frac{\partial x_i(\mathbf{w})}{\partial w_j}\frac{d\mathbf{v}}{dx}\right]_{\mathbf{w}^{(0)}} \\ &= O(\Delta t)\end{aligned}$$

since

$$\left[\frac{\partial x_i}{\partial w_j}\right]_{\mathbf{w}^{(0)}} = O(\Delta t)$$

Hence

$$\|\mathbf{F}(\mathbf{u}) - \mathbf{F}(\mathbf{v})\| \leq (c_1\rho + c_2\Delta t)\,\|\mathbf{u} - \mathbf{v}\|$$

For ρ and Δt small enough $\theta \equiv c_1\rho + c_2\Delta t$ can be made less than 1. Then by choosing Δt even smaller one can ensure that

$$K\Delta t \leq (1 - c_1\rho - c_2\Delta t)\rho$$

and the requirements of the contractive mapping theorem are satisfied.

As mentioned previously, each group of characteristic points is advanced as a unit. The steps to advance a group to the next time level (once shocks have been fitted) are as follows :

(i) Calculate $\mathbf{v}(z, t + \Delta t)$ as outlined above where z lies at the end of the j^{th} characteristic curve which begins from a given characteristic point. Do this for each for each characteristic $(j = 1, 2, \ldots, m)$ and for each point in the group.

(ii) Each characteristic point at time t has spawned m new points at time $t + \Delta t$. These points are not all kept; points are removed where the solution is smoothest.

2.4 Discontinuities

The solutions to hyperbolic system of conservation laws can possess discontinuities. These may be present from time zero if the initial conditions contain jumps. The discontinuites may also develop in time if the system is nonlinear. Discontinuities are treated in a special manner by the program.

Consider an isolated discontinuity which is propagating through the flow. The states on either side of the discontinuity are assumed to be smooth. The speed of the discontinuity, U, is then given by the jump conditions $[\mathbf{f}] = U[\mathbf{u}]$. The discontinuity is assumed to satisfy the entropy condition. This condition was given in section 1.1. In words the entropy condition states that there is one and only one extra characteristic running into the discontinuity. We will usually call a discontinuity which satisfies the jump conditions and the entropy condition a ***shock***. The numerical procedure for advancing a shock (***shock fitting***) is described in section 2.4.1. We will see that the jump conditions and the one extra characteristic equation provide enough equations to determine the shock speed and the states to the left and right

of the shock.

Another type of discontinuity can exist at a given point in time. This discontinuity does not satisfy the jump conditions. It may arise in initial conditions or when two shocks collide. In order to determine the solution at the next time level a more general Riemann problem is solved. The discontinuity will then be resolved into shocks and expansion fans.

There is a good chance that there will be a shock in any given group of characteristic points. Before the characteristic points are advanced to the next time level, the group is first scanned for the existence of shocks. Shocks are indicated by the crossing of two characteristics of the same *family* (i.e. lying on the same numbered characteristic).

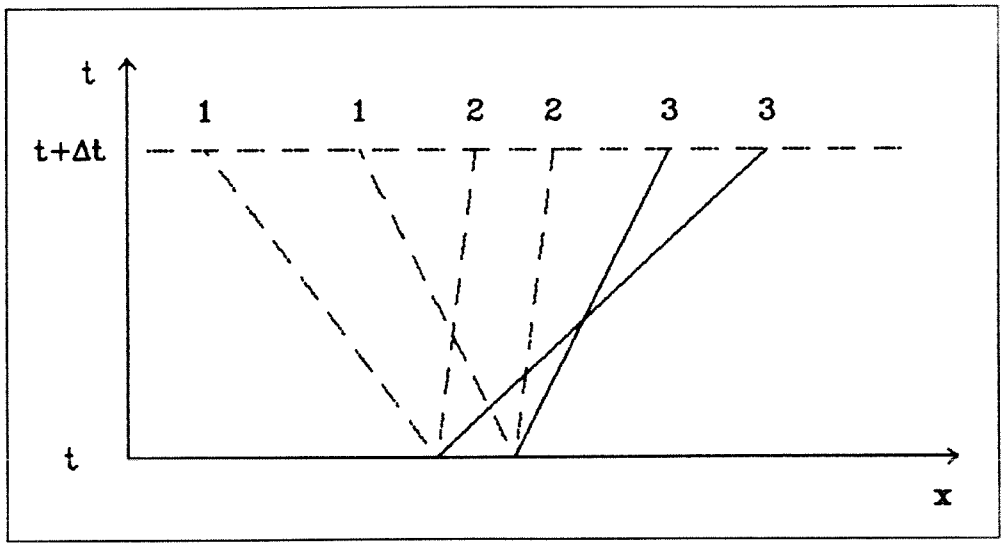


Figure 2.3 Recognizing a Shock

The program considers two cases when characteristics are found to cross.

(i) Most often the crossing has occurred where a shock has been previously fitted. In this case the states on either side will not be arbitrary but will satisfy the shock relations

$$[\mathbf{f}] - U[\mathbf{u}] = 0$$

(Actually since the shock velocity U is not known, the ratios of the jump in f_i to the jump in u_i are checked to see if they are the same. This also gives a good initial guess for U.)

(ii) If the conditions of (i) are not satisfied (as might be the case when shocks collide or with discontinous initial conditions).

In case (i) a single shock is fitted to determine the solution at the next time level. This shock fitting procedure is discussed in the next section. In case (ii) the more general Riemann problem must be solved. (Actually shock fitting is just the solution of a special Riemann problem when only one shock appears in the solution.) A numerical technique for the solution of the Riemann problem is given in section 2.4.4. Before describing the procedure, we first discuss shock interactions (section 2.4.2) and give some background material on the Riemann problem (section 2.4.3).

2.4.1 Fitting a Single Shock

Suppose the shock occurs on characteristic k (a k-shock). The shock fitting problem requires the determination of the states to the left and right of the shock at the next time level as well as the shock speed. The appropriate characteristic equations to use in determining these values are those which correspond to the characteristic curves which do not cross the shock. Hence for a shock occurring on the characteristic family k (called a *k-shock*) there will be equations for characteristics 1 to k coming from the right of the shock and equations for characteristics k to m coming from the left of the shock.

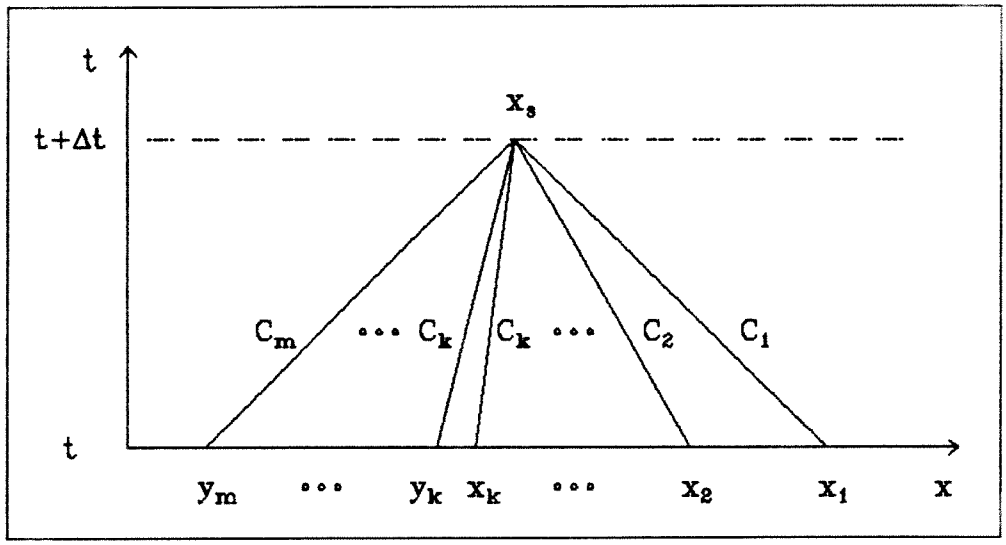


Figure 2.4 Shock Fitting

There are $3m + 2$ unknowns

$$\begin{cases} \mathbf{u}_L, \mathbf{u}_R & \text{states to the left and right of the shock} \\ U & \text{shock velocity} \\ x_1, x_2, \ldots, x_k, y_k, y_{k+1}, \ldots, y_m & \text{initial positions of the characteristics} \end{cases}$$

The $3m + 2$ equations which must be solved are

$$\mathbf{a}_i(\frac{1}{2}(\mathbf{u}_R + \mathbf{v}(x_i, t))^T(\mathbf{u}_R - \mathbf{v}(x_i, t)) = 0 \qquad i = 1, 2, \ldots, k \tag{1a}$$

$$\mathbf{a}_i(\frac{1}{2}(\mathbf{u}_L + \mathbf{v}(y_i, t))^T(\mathbf{u}_L - \mathbf{v}(y_i, t)) = 0 \quad i = k, k+1, \ldots, m \tag{1b}$$

$$[\mathbf{f}(\mathbf{u}_R) - \mathbf{f}(\mathbf{u}_L)] - U[\mathbf{u}_R - \mathbf{u}_L] = 0 \qquad \text{(m equations)} \tag{1c}$$

$$x_s - x_i = c_i(\frac{1}{2}(\mathbf{u}_R + \mathbf{v}(x_i, t)))\Delta t \qquad i = 1, 2, \ldots, k \tag{1d}$$

$$x_s - y_i = c_i(\frac{1}{2}(\mathbf{u}_L + \mathbf{v}(y_i, t)))\Delta t \qquad i = k, k+1, \ldots, m \tag{1e}$$

The position of the shock at $t + \Delta t$ is x_s.

$$x_s \equiv x_s^0 + \frac{1}{2}[U + U^{(0)}]\Delta t$$

In this last equation x_s^0 is the shock position at time t and $U^{(0)}$ is the shock velocity at time t. $U^{(0)}$ can be determined from the jump conditions. These equations are

solved by a quasi-Newton method. By which to say that only the first $2m+1$ equations and variables are used in the Newton step. The variables x_i and y_i are updated after each Newton step from the equations which describe the position of the characteristic. This was done for simplicity.

We now derive some conditions under which the iteration to solve the shock fitting equations (1) will converge. We assume that at time t the solution $\mathbf{v}(x,t)$ is smooth in the neighbourhood of the shock, x_s^0, except for a jump at x_s^0. As in section 2.3 we first reduce the number of variables by assuming that the x_i and y_i can be determined, from equations (1d) and (1e), in terms of the other unknowns

$$\mathbf{w} = \begin{bmatrix} \mathbf{u}_L \\ U \\ \mathbf{u}_R \end{bmatrix}$$

One must be a bit careful in this case since the solution $\mathbf{v}(x,t)$ is discontinuous at $x = x_s^0$. Thus we must assume that $\mathbf{w} \in \mathbf{B}_\rho(\mathbf{w}^{(0)})$ where $\mathbf{w}^{(0)}$ is the solution when $\Delta t = 0$. Then if ρ is small enough and the entropy condition is satisfied it is not hard to show that any jump in $\mathbf{v}(x,t)$ is not a factor since x_i and y_i will always stay on the proper side of the discontinuity.

The iteration thus reduces to a Newton iteration to solve the nonlinear system

$$\mathbf{F}(\mathbf{u}) = 0 \qquad \mathbf{F}(\mathbf{u}) = \begin{bmatrix} F_1(\mathbf{u}) \\ \vdots \\ F_{2m+1}(\mathbf{u}) \end{bmatrix} \tag{2}$$

where the first $m-k+1$ components of $\mathbf{F}$ are equations (1a), the second m components are equations (1c) and the last k components are equations (1b). Assuming that the initial guess $\mathbf{w}^{(0)}$ is good enough the Newton iteration

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \mathbf{F}_{\mathbf{w}}^{-1}(\mathbf{w}^{(k)})\mathbf{F}(\mathbf{w}^{(k)}) \qquad ; \qquad \mathbf{w}^{(k)} = \begin{bmatrix} \mathbf{u}_L^{(k)} \\ U^{(k)} \\ \mathbf{u}_R^{(k)} \end{bmatrix}$$

will converge if we can show that

$$\|\mathbf{F}_{\mathbf{w}}^{-1}(\mathbf{w}^{(0)})\| \leq M$$

for some constant M (independent of Δt).

The Jacobian Matrix $\mathbf{F}_{\mathbf{w}}^{(0)} \equiv \mathbf{F}_{\mathbf{w}}(\mathbf{w}^{(0)})$ is

$$\mathbf{F}_{\mathbf{w}}^{(0)} = \begin{bmatrix} \mathbf{a}_k^T(L) & 0 & 0 \cdots 0 \\ \vdots & \vdots & \vdots \\ \mathbf{a}_m^T(L) & 0 & 0 \cdots 0 \\ -\mathbf{f}_{\mathbf{u}}(L) + U^{(0)} I & -[\mathbf{u}] & \mathbf{f}_{\mathbf{u}}(R) - U^{(0)} I \\ 0 \cdots 0 & 0 & \mathbf{a}_1^T(R) \\ \vdots & \vdots & \vdots \\ 0 \cdots 0 & 0 & \mathbf{a}_k^T(R) \end{bmatrix}$$

where we have used the shorthand notations

$$\mathbf{a}_i(L) = \mathbf{a}_i(\mathbf{u}_L^{(0)}, U^{(0)}) \qquad \mathbf{f}_{\mathbf{u}}(L) = \mathbf{f}_{\mathbf{u}}(\mathbf{u}_L^{(0)})$$

$$\mathbf{a}_i(R) = \mathbf{a}_i(\mathbf{u}_R^{(0)}, U^{(0)}) \qquad \mathbf{f}_{\mathbf{u}}(R) = \mathbf{f}_{\mathbf{u}}(\mathbf{u}_R^{(0)})$$

and

$$[\mathbf{u}] = \mathbf{u}_R^{(0)} - \mathbf{u}_L^{(0)}$$

Recall that $\mathbf{a}_i$ is a left eigenvector so that

$$\mathbf{a}_i^T(L)\mathbf{f}_{\mathbf{u}}(L) = c_i(L)\mathbf{a}_i^T(L)$$

and

$$\mathbf{a}_i^T(R)\mathbf{f}_{\mathbf{u}}(R) = c_i(R)\mathbf{a}_i^T(R)$$

These facts will allow us to determine some conditions for the nonsingularity of $\mathbf{F}_{\mathbf{w}}^{(0)}$. The procedure to determine these conditions is to transform $\mathbf{F}_{\mathbf{w}}^{(0)}$, by taking linear combinations of the rows, to a new matrix. If this new matrix is nonsingular then so is $\mathbf{F}_{\mathbf{w}}^{(0)}$. Let A_{LR} be the matrix

$$A_{LR} = \begin{bmatrix} \mathbf{a}_1^T(R) \\ \vdots \\ \mathbf{a}_k^T(R) \\ \mathbf{a}_{k+1}^T(L) \\ \vdots \\ \mathbf{a}_m^T(L) \end{bmatrix}$$

We multiply A_{LR} into middle rows ($m-k+2$ to $2m-k+1$) of $\mathbf{F}_{\mathbf{w}}^{(0)}$. Then we use the fact that

$$A_{LR}\left(-\mathbf{f_u}(L)+U^{(0)}I\right)=\begin{bmatrix}\mathbf{a}_1^T(R)(-\mathbf{f_u}(L)+U^{(0)}I)\\ \vdots\\ \mathbf{a}_k^T(R)(-\mathbf{f_u}(L)+U^{(0)}I)\\ (U^{(0)}-c_{k+1}(L))\mathbf{a}_{k+1}^T(L)\\ \vdots\\ (U^{(0)}-c_m(L))\mathbf{a}_m^T(L)\end{bmatrix}$$

There is a corresponding result for

$$A_{LR}\left(U^{(0)}I-\mathbf{f_u}(L)\right)$$

Using these results the following proposition can be shown.

Proposition. *The Jacobian matrix for the Newton iteration,* $\mathbf{F}_{\mathbf{w}}^{(0)}$, *is nonsingular if*

(i) $\mathbf{a}_i^T(R)(-\mathbf{f_u}(L)+U^{(0)}I)\mathbf{a}_i(L)\neq 0 \qquad i=1,\ldots,k-1,$

(ii) $\mathbf{a}_k(R)^T(\mathbf{u}_R^{(0)}-\mathbf{u}_L^{(0)})\neq 0$ *and*

(iii) $\mathbf{a}_i^T(L)(\mathbf{f_u}(R)-U^{(0)}I)\mathbf{a}_i(R)\neq 0 \qquad i=k+1,\ldots,m$

To see that these conditions might be reasonable consider the case when $\mathbf{f_u}$ is symmetric. Then the first and last conditions reduce to

$$(c_i(L)-U^{(0)})\,\mathbf{a}_i^T(R)\mathbf{a}_i(L)\neq 0 \qquad i=1,\ldots,k-1$$

and

$$(c_i(R)-U^{(0)})\,\mathbf{a}_i^T(L)\mathbf{a}_i(R)\neq 0 \qquad i=k+1,\ldots,m$$

Now by the entropy condition $c_i(L)-U^{(0)}\neq 0$ and $c_i(R)-U^{(0)}\neq 0$ for $i\neq k$. Hence we are left with a condition on the eigenvectors. Condition (ii) is not unreasonable since for a weak k-shock the jump in $\mathbf{u}$ is in the direction of $\mathbf{a}_k$.

At a true contact discontinuity the characteristics which form the discontinuity (the characteristic numbered k) do not point in to the contact but are instead

parallel to it. Computationally the characteristics may point slightly in or out. The program recognizes such a situation as a possible shock. Thus contact discontinuites can be treated like shocks, except in the following regard. It was found that any initial errors in the contact did not disappear. At a shock, where the characteristics point in, the errors get *washed* into the shock. It is the more global behaviour of the solution which determines the states in the immediate vicinity of the shock. To improve the values at a contact the two parallel characteristics were made to point slightly inward. This procedure is in the spirit of the artificial compression method, Harten [1977]. Actually the program just makes sure that the k characteristics on either side of any discontinuity must point in at least a small amount. For normal shocks no change is made. For contacts, the point at which the k characteristics originate are moved slightly away from the discontinuity.

2.4.2 Shock Interactions

The shock relations describe how isolated discontinuties propagate. However, this information does not yet completely specify the solution. For what happens when two shocks merge or collide? Further details must be given to indicate how the interactions between discontinuities are to be handled. Below we describe the manner in which shock interactions proceed. It is then shown that this description describes a weak solution.

Consider the situation when two shocks collide. Let $\mathbf{u}_l$, $\mathbf{u}_m$ and $\mathbf{u}_r$ be the states in the smooth regions between the shocks. The subscripts are to refer to *left*, *middle* and *right*. The speeds of the shocks are simply determined by the states immediately in front of and in back of each shock. The shock speeds are given by the Rankine-Hugoniot relations. The shocks will continue to move together until the section between them vanishes. Now the states on either side of this discontinuity will not in general satisfy the jump conditions for a shock. This problem must then be considered as a general Riemann problem to be solved. The solution to this

Riemann problem should generate the appropriate shocks, contacts and fans that result from the collision.

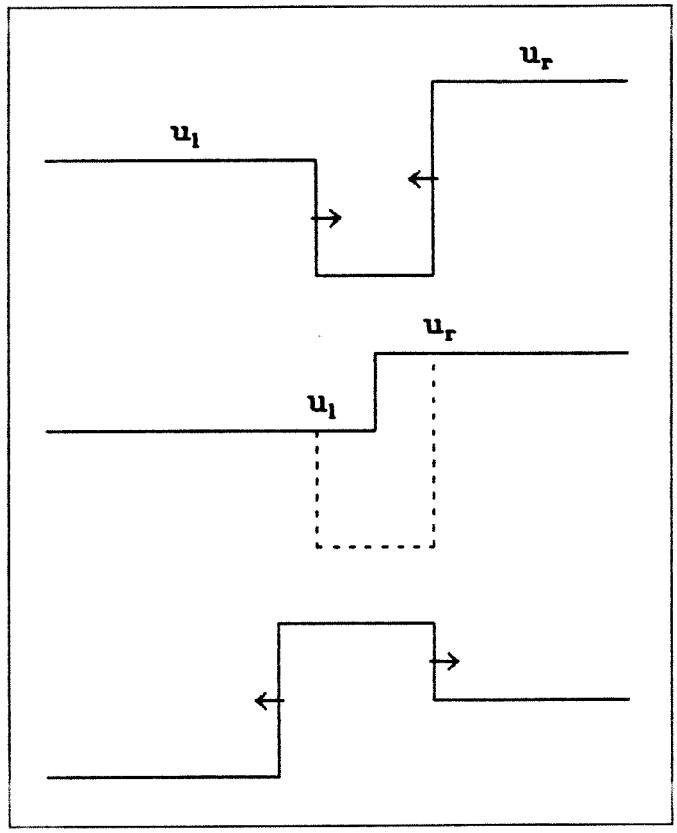


Figure 2.5 Shock Interactions

To show that the above procedure is reasonable one can show that the scenario given above for the collision of discontinuities describes a weak solution to the conservation laws.

Proposition (Shock Interations). *Let $U_1(t) > 0$ and $U_2(t) < 0$ be the speeds of two shocks which collide at time $t = 0$. The positions of these shocks for $t < 0$ are denoted by $\xi_1(t)$ and $\xi_2(t)$ and given by*

$$\frac{d\xi_1}{dt} = U_1(t) \qquad \xi_1(0) = 0$$
$$\frac{d\xi_2}{dt} = U_2(t) \qquad \xi_2(0) = 0$$

Let $\mathbf{u}_l(x,t)$, $\mathbf{u}_m(x,t)$ and $\mathbf{u}_r(x,t)$ be solutions of the system of conservation laws

for $t < 0$ in the regions between the shocks,

$$(\mathbf{u}_l)_t + \mathbf{f}(\mathbf{u}_l)_x = 0 \qquad -\infty < x < \xi_1(t)$$

$$(\mathbf{u}_m)_t + \mathbf{f}(\mathbf{u}_m)_x = 0 \qquad \xi_1(t) < x < \xi_2(t)$$

$$(\mathbf{u}_r)_t + \mathbf{f}(\mathbf{u}_r)_x = 0 \qquad \xi_2(t) < x < \infty$$

which also satisfy the jump conditions

$$(\mathbf{f}(\mathbf{u}_m(\xi_1(t),t)) - \mathbf{f}(\mathbf{u}_l(\xi_1(t),t))) = U_1(t)(\mathbf{u}_m(\xi_1(t),t) - \mathbf{u}_l(\xi_1(t),t))$$

$$(\mathbf{f}(\mathbf{u}_r(\xi_2(t),t)) - \mathbf{f}(\mathbf{u}_m(\xi_2(t),t))) = U_2(t)(\mathbf{u}_r(\xi_2(t),t) - \mathbf{u}_m(\xi_2(t),t))$$

Let $\mathbf{v}(x,t)$ be a solution to the initial value problem which results at the instant the shocks collide.

$$\mathbf{v}_t + \mathbf{f}(\mathbf{v})_x = 0 \tag{1}$$

$$\mathbf{v}(x,0) = \begin{cases} \mathbf{u}_l(x,0) & x < 0 \\ \mathbf{u}_r(x,0) & x > 0 \end{cases}$$

Then the following function defines a weak solution to the hyperbolic system of conservation laws

$$\mathbf{u} = \begin{cases} \mathbf{u}_l & x < \xi_1(t) \\ \mathbf{u}_m & \xi_1(t) < x < \xi_2(t) \\ \mathbf{u}_r & x > \xi_2(t) \end{cases} \qquad \text{for} \quad t > 0$$

$$\mathbf{u} = \mathbf{v}(x,t) \qquad \text{for} \quad t > 0$$

Proof.

We need to show that the following expression is zero for all smooth test functions ϕ with compact support.

$$I = \int_{t=-\infty}^{\infty} \int_{x=-\infty}^{\infty} [\mathbf{u}\phi_t + \mathbf{f}(\mathbf{u})\phi_x] dx dt$$

First split the integral in time into an integral from $-\infty$ to 0 and an integral from 0 to ∞. Denote the first double integral by I_1 and the second by I_2.

$$I = \int_{t=-\infty}^{0} \int_{x=-\infty}^{\infty} [\mathbf{u}\phi_t + \mathbf{f}\phi_x] dx dt + \int_{t=0}^{\infty} \int_{x=-\infty}^{\infty} [\mathbf{u}\phi_t + \mathbf{f}\phi_x] dx dt$$
$$\equiv I_1 + I_2$$

For $t < 0$ the solution u is formed from $\mathbf{u}_l$, $\mathbf{u}_m$ and $\mathbf{u}_r$. Substitute $\mathbf{u}_l$,$\mathbf{u}_m$ and $\mathbf{u}_r$ for u in the first integral.

$$I_1 = \int_{t=-\infty}^{0} \{ \int_{x=-\infty}^{\xi_1} [\mathbf{u}_l \phi_t + \mathbf{f}(\mathbf{u}_l)\phi_x] dx + \int_{x=\xi_1}^{\xi_2} [\mathbf{u}_m \phi_t + \mathbf{f}(\mathbf{u}_m)\phi_x] dx + \int_{x=\xi_2}^{\infty} [\mathbf{u}_r \phi_t + \mathbf{f}(\mathbf{u}_r)\phi_x] dx \} dt$$

These integrals can be simplified using the relations

$$\int_{x=-\infty}^{\xi_1} [\mathbf{u}_l \phi_t] dx = \frac{d}{dt} \int_{x=-\infty}^{\xi_1} \mathbf{u}_l \phi dx - \mathbf{u}_l \phi(\xi_1) \frac{d\xi_1}{dt} - \int_{x=-\infty}^{\xi_1} (\mathbf{u}_l)_t \phi dx$$

and

$$\int_{x=-\infty}^{\xi_1} \mathbf{f}(\mathbf{u}_l)\phi_x dx = [\mathbf{f}(\mathbf{u}_l)\phi]_{\xi_1} - \int_{x=-\infty}^{\xi_1} \mathbf{f}(\mathbf{u}_l)_x \phi dx$$

Similar expressions are obtained for the space integrals from ξ_1 to ξ_2 and from ξ_2 to ∞. After substitution the expression for I_1 becomes

$$\begin{aligned} I_1 = & \int_{x=-\infty}^{\xi_1} \mathbf{u}_l(x,0)\phi(x,0)dx + \int_{x=\xi_1}^{\xi_2} \mathbf{u}_m(x,0)\phi(x,0)dx + \int_{x=\xi_2}^{\infty} \mathbf{u}_r(x,0)\phi(x,0)dx \\ & + \int_{t=-\infty}^{0} \{ -\frac{d\xi_1}{dt}[\mathbf{u}_l(\xi_1,t) - \mathbf{u}_m(\xi_1,t)]\phi(\xi_1,t) - \frac{d\xi_2}{dt}[\mathbf{u}_m(\xi_2,t) - \mathbf{u}_r(\xi_2,t)] \\ & + [\mathbf{f}(\mathbf{u}_l(\xi_1,t)) - \mathbf{f}(\mathbf{u}_m(\xi_1,t))]\phi(\xi_1,t) + [\mathbf{f}(\mathbf{u}_m(\xi_2,t)) - \mathbf{f}(\mathbf{u}_r(\xi_2,t))]\phi(\xi_2,t) \} dt \end{aligned}$$

Hence

$$I_1 = \int_{x=-\infty}^{\infty} \mathbf{u}(x,0)\phi(x,0)dx$$

The integral I_2 is easily simplified since $\mathbf{u}$ is equal to $\mathbf{v}$ for $t > 0$ and $\mathbf{v}$ is the solution to the Riemann problem (1).

$$\begin{aligned} I_2 &= \int_{t=0}^{\infty} \int_{x=-\infty}^{\infty} [\mathbf{u}\phi_t + \mathbf{f}(\mathbf{u})\phi_x] dx dt \\ &= \int_{t=0}^{\infty} \int_{x=-\infty}^{\infty} [\mathbf{v}\phi_t + \mathbf{f}(\mathbf{v})\phi_x] dx dt \\ &= -\int_{x=-\infty}^{\infty} \mathbf{v}(x,0)\phi(x,0)dx \\ &= -\int_{x=-\infty}^{\infty} \mathbf{u}(x,0)\phi(x,0)dx \end{aligned}$$

Hence $I = I_1 + I_2 = 0$. This shows that the description given for the interaction of discontinuities describes a weak solution to the conservation laws. We must further ask whether this weak solution is the *physically correct* solution. This requires that the two shocks and the solution $\mathbf{v}$ be solutions which satisfy the entropy condition. Actually the above proposition is a special case of the result that two weak solutions, one defined for $t < t_0$ and the other for $t > t_0$, which agree at $t = t_0$ can be combined to form a new weak solution.

2.4.3 The Riemann Problem

We now discuss some aspects of the Riemann problem. In the next section a numerical algorithm for computing a solution will be described. Recall that the Riemann problem is the solution to the initial value problem when the initial data consist of two constant states.

$$\mathbf{u}_t + (\mathbf{f}(\mathbf{u}))_x = 0 \tag{1}$$

$$\mathbf{u}(x,0) = \begin{cases} \mathbf{u}_L & x < 0 \\ \mathbf{u}_R & x > 0 \end{cases}$$

From the form of the initial conditions one might expect that the solution will depend solely on the similarity variable $\xi = x/t$. Assume a solution of this form, $\mathbf{u}(x,t) = \mathbf{v}(\xi)$. The partial derivatives of $\mathbf{u}$ with respect to time and space become

$$\mathbf{u}_t = (-\xi/t)\frac{d\mathbf{v}}{d\xi}$$

$$\mathbf{u}_x = 1/t\frac{d\mathbf{v}}{d\xi}$$

Upon substitution, equation (1) becomes

$$J(\mathbf{v})\frac{d\mathbf{v}}{d\xi} = \xi\frac{d\mathbf{v}}{d\xi} \tag{2}$$

where $J(\mathbf{v})$ is the Jacobian matrix of $\mathbf{f}(\mathbf{v})$

$$J(\mathbf{v}) = \mathbf{f}_\mathbf{v}$$

Equation (2) is in the form of an eigenvalue equation with eigenvector $\mathbf{v}(\xi)$ and eigenvalue ξ. Possible solutions are the trivial solution

$$\mathbf{v} = constant \tag{3}$$

or

$$\frac{d\mathbf{v}}{d\xi} = \alpha(\xi)\mathbf{r}_i(\mathbf{v}) \tag{4a}$$

$$\xi = c_i(\mathbf{v}) \tag{4b}$$

$\mathbf{r}_i$ is the right eigenvector of J corresponding to the eigenvalue c_i and $\alpha(\xi)$ is some scalar depending on ξ. Given initial values ξ_0 and $\mathbf{v}_0$ we can try to solve equation (4a) to determine a solution that depends on ξ_0, $\mathbf{v}_0$ and $\alpha(\xi)$. However, the second equation (4b) places constraints on the solution. Differentiating (4b) with respect to ξ gives

$$\nabla {c_i}^T \frac{d\mathbf{v}}{d\xi} = 1$$

Hence, using (4a)

$$\alpha(\xi)\nabla {c_i}^T \mathbf{r}_i(v) = 1$$

Provided that $\nabla {c_i}^T \mathbf{r}_i(v)$ is never zero this equation determines what α should be. This gives the initial value problem

$$\begin{aligned} &\frac{d\mathbf{v}}{d\xi} = \frac{1}{\nabla {c_i}^T \mathbf{r}_i(v)}\mathbf{r}_i(v) \\ &\mathbf{v} = \mathbf{v}_0 \quad \text{at} \quad \xi = \xi_0 \quad \text{and} \quad \xi_0 = c_i(\mathbf{v}_0) \end{aligned} \tag{5}$$

Lax [1972] calls the system ***strictly nonlinear*** if $\nabla {c_i}^T \mathbf{r}_i(v)$ is always nonzero. The solution to (5) will be a *classical* similarity solution to the Riemann problem. This solution is called a *fan* or *expansion fan*. Another similarity solution to (1) which is only a solution in the weak sense is the shock solution

$$\mathbf{v} = \begin{cases} \mathbf{v}_L & \text{for } \xi < U \\ \mathbf{v}_R & \text{for } \xi > U \end{cases}$$

The states $\mathbf{v}_L$ and $\mathbf{v}_R$ and the constant shock speed U must satisfy the jump conditions

$$(\mathbf{f}(\mathbf{v}_R) - \mathbf{f}(\mathbf{v}_L)) = U(\mathbf{v}_R - \mathbf{v}_L)$$

In addition, for a physically valid solution we require that the entropy condition is satisfied.

Let us suppose that these two types of solutions are all we have to work with to try and construct a weak solution to the full Riemann problem (1). The solution will consist of a sequence of fans or shocks separated by constant states. There will be a shock or fan corresponding to each family of characteristics. Beginning with the state $\mathbf{u}_L$ we generate a solution $\mathbf{u}(\mathbf{u}_L, \xi_1)$ through a 1-wave. A 1-wave is a fan or shock solution formed on characteristic 1. The parameter ξ_1 in this solution is not yet determined but will either be the value of ξ at which the fan solution stops or the speed of the shock. The state $\mathbf{u}(\mathbf{u}_L, \xi_1)$ will now be connected through a 2-wave to a state $\mathbf{u}(\mathbf{u}_L, \xi_1, \xi_2)$. Continuing in this manner one obtains a solution $\mathbf{u}(\mathbf{u}_L, \xi_1, \xi_2, \ldots, \xi_m)$ which depends on the m parameters ξ_i. We will have a solution provided we can choose the ξ_i so that $\mathbf{u}(\mathbf{u}_L, \xi_1, \xi_2, \ldots, \xi_m) = \mathbf{u}_R$. Lax [1972] has shown that the ξ_i can indeed be chosen to satisfy this equation provided $\|\mathbf{u}_L - \mathbf{u}_R\|$ is sufficiently small and the system is strictly nonlinear. In this case the system is essentially linear in behaviour and

$$\mathbf{u}(\mathbf{u}_L, \xi_1, \xi_2, \ldots, \xi_m) = \mathbf{u}_L + \sum_{i=1}^{m} \mathbf{r}_i(\mathbf{u}_L)\xi_i + O({\xi_i}^2)$$

Since the right eigenvectors $\mathbf{r}_i$ are linearly independent we can solve for the ξ_i to make the above expression equal to $\mathbf{u}_R$

In general the form of the solution to the Riemann problem is not known. For many systems of interest the solution is of the form of constant states separated by fans or shocks. The numerical procedure described in the next section assumes this form. If one has a system for which the Riemann solution is of a different form it would be necessary to replace this solver with one that will perform correctly.

2.4.4 The General Riemann Solver

In this section a numerical procedure for solving the Riemann problem is discussed. Efficient algorithms have been devised for specialized systems. An iteration procedure for the equations of gas dynamics with certain types of gas laws was given by Godunov [1959]. Improvements and extensions of this scheme were made by, for example, van Leer [1979] and Colella and Glaz [1982]. The algorithm presented here applies to more general systems, those for which the Riemann solution is of the form of constant states separated by fans or shocks. This more general scheme suffers from the drawback of being slower than those optimized for particular systems.

Now consider solving the Riemann problem numerically. To each characteristic family $i = 1, .., m$ there will be a possible shock or fan appearing in the solution. The shock or fan may degenerate to zero strength. To begin with one does not know which characteristics lead to shocks and which to fans. This complicates matters in two ways. Firstly the equations that must be solved depend on whether there is a shock or fan; through a fan the characteristic equations are solved while across a shock the Rankine Hugoniot jump conditions hold. Secondly, the number of equations varies with the number of shocks, as will be seen shortly. Suppose for the moment that one knows which characteristics form shocks and which form fans. Let n_s be the number of shocks in the solution. The unknowns to be solved for are

(1) $\mathbf{u}^k \qquad k = 1, 2, \ldots, m-1$ The constant state which separates the k-wave (fan or shock) from the (k+1)-wave. Define $\mathbf{u}^0$ to be $\mathbf{u}_L$ and $\mathbf{u}^m$ to be $\mathbf{u}_R$.

(2) $U_{i_k} \qquad k = 1, 2, \ldots, n_s$ The velocity of the shock which occurs on characteristic $i_k, i_k \in (1, 2, \ldots, m)$.

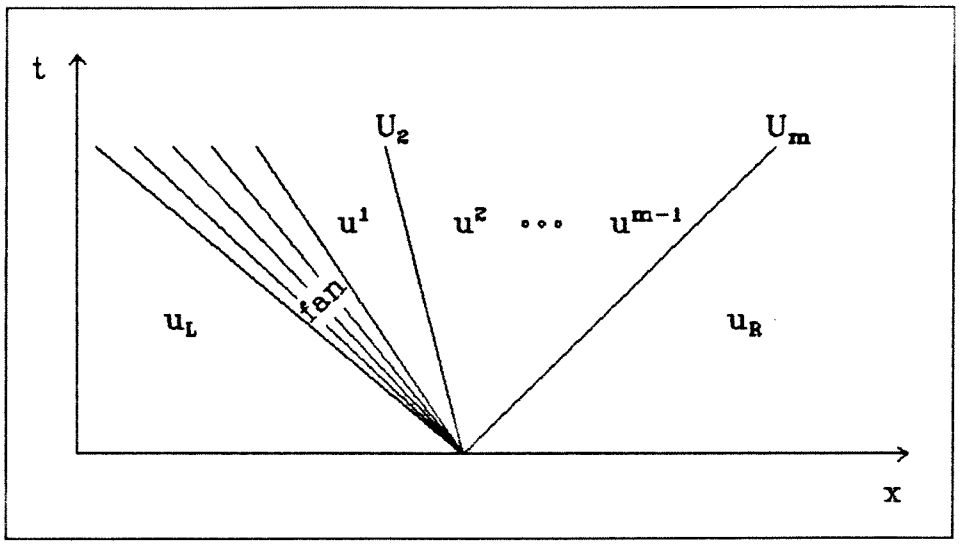


Figure 2.6 Form of the Solution to the Riemann Problem

The equations to solve are of the form

(1) The Jump conditions across each shock (on characteristic k say)

$$[\mathbf{f}(\mathbf{u}^{k+1}) - \mathbf{f}(\mathbf{u}^k)] - U_k[\mathbf{u}^{k+1} - \mathbf{u}^k] = 0$$

(2) The characteristic equations through each expansion fan (on characteristic k)

$$\mathbf{a}_i(\mathbf{u})^T \frac{d\mathbf{u}}{dt} = 0 \qquad \text{on} \quad C_i \quad i = 1, 2, \ldots, k-1, k+1, \ldots, m$$

where C_i is the characteristic which passes through the k-fan. A simple approximation to this equation is thus

$$\mathbf{a}_i(\frac{1}{2}(\mathbf{u}^{k+1} + \mathbf{u}^k))^T(\mathbf{u}^{k+1} - \mathbf{u}^k) = 0$$

The number of equations and the number of unknowns are both equal to $m(m-1) + n_s$. In practice it was found necessary to to use a more accurate formula for the characteristic equations which are calculated through strong expansion fans (since the solution changes appreciably through the fan). This is done by adding

additional states through the fan. If there is a fan between the states $\mathbf{u}^{k-1}$ and $\mathbf{u}^k$ denote the extra states by $\mathbf{u}^{k_\nu}$ $\nu = 1, 2, \ldots, k_e$. The number of extra states k_e is determined by the angle of opening of the fan. The states are positioned through the fan along the lines

$$x/t = \alpha_\nu c_k(\mathbf{u}^{k-1}) + (1 - \alpha_\nu) c_k(\mathbf{u}^k)$$

where $\alpha_\nu = \nu/(k_e + 1)$ for the ν^{th} extra state. The extra equations that are solved are

$$\mathbf{a}_i\left(\frac{1}{2}(\mathbf{u}^{k_{\nu+1}} + \mathbf{u}^{k_\nu})\right)^T(\mathbf{u}^{k_{\nu+1}} - \mathbf{u}^{k_\nu}) = 0 \qquad i = 1, 2, \ldots, k-1, k+1, \ldots, m$$

$$c_k(\mathbf{u}^{k_\nu}) = \alpha_\nu c_k(\mathbf{u}^{k-1}) + (1 - \alpha_\nu) c_k(\mathbf{u}^k)$$

In these expressions we have defined

$$\mathbf{u}^{k_0} = \mathbf{u}^{k-1} \qquad \mathbf{u}^{k_{k_e+1}} = \mathbf{u}^k$$

Notice that Δt appears nowhere in the equations. This is to be expected as the solution to the Riemann problem is self similar in variable x/t.

These equations are solved by Newton's Method. The initial guess at the solution is sometimes not very good and it has been found necessary to start the iteration with a few steps of the method of steepest descent. Then during the Newton iteration a partial Newton step is taken if the corrections are still too large. As the iteration process proceeds the number of unknowns may vary. This may be the result of the number of shocks, n_s, changing or the number of extra states through a fan changing. Contact discontinuities in particular tend to oscillate between being shocks and being fans. Since Newton's method is used, a Jacobian matrix must be inverted. It was found that early on in the iteration the Jacobian could be nearly singular (due apparently to the number of unknowns being incorrect). When this situation was encountered the *weakest* shock was replaced by a fan, thus reducing the number of unknowns by one.

This solver has worked well when tested on the equations of gas dynamics. Table 2.1 gives some results for a particular Riemann problem. This is the problem solved in example 1 of the third chapter. The solution consists of an expansion fan, a contact discontinuity, and a shock. There are two interior constant states. In the table the values of the density, momentum and energy for these constant states are given. The state between the fan and contact discontinuity is denoted by a subscript 2, and the state between the contact and the shock by a subscript 3. Results are given for 0,1,2,3 or 4 extra states through the fan.

Riemann Problem					
True	Calculated				
	$k_e = 0$	$k_e = 1$	$k_e = 2$	$k_e = 3$	$k_e = 4$
$\rho_2 = .4263$	.4746	.4397	.4324	.4298	.4285
$\rho_3 = .2656$	.2654	.2658	.2657	.2657	.2656
$m_2 = .3954$	.4399	.4084	.4014	.3988	.3976
$m_3 = .2463$	.2460	.2469	.2466	.2465	.2464
$E_2 = .9412$	.9610	.9485	.9447	.9432	.9425
$E_3 = .8720$	.8712	.8735	.8729	.8726	.8724

Table 2.1 Results from the Riemann Solver

Chapter 3

Computational Results

3.1 The Equations of Gas Dynamics

In this section results are presented for the numerical solution of the equations of gas dynamics. The three components of **u** have the more common names of density, momentum and pressure.

$$\mathbf{u} = \begin{bmatrix} \rho \\ m \\ E \end{bmatrix} \qquad \begin{aligned} \rho &= density \\ m &= momentum \\ E &= Energy \end{aligned}$$

The flux function **f** for an inviscid polytropic gas is

$$\mathbf{f}(\mathbf{u}) = \begin{bmatrix} m \\ m^2/\rho + p \\ m/\rho(E+p) \end{bmatrix}$$

where the pressure p and velocity u are defined as

$$p = (\gamma - 1)[E - \frac{1}{2}m^2/\rho] \qquad \gamma = 1.4$$

$$u = m/\rho$$

The computer code requires expressions for the coefficients which appear in the characteristic equations. These can be obtained as follows. First the Jacobian matrix is determined.

$$\mathbf{f_u}(\mathbf{u}) = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{2}(\gamma-3)u^2 & (3-\gamma)u & \gamma - 1 \\ -\gamma mE/\rho + (\gamma-1)m^3/\rho^3 & \gamma E/\rho - \frac{3}{2}(\gamma-1)m^2/\rho^2 & \gamma u \end{bmatrix}$$

The eigenvalues and left eigenvectors of this matrix can be calculated in a straightforward manner.

$$c_1 = u - a \qquad \mathbf{a}_1 \equiv \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \end{bmatrix} = \begin{bmatrix} -u(\frac{1}{2}u + a/(\gamma-1)) \\ u + a/(\gamma-1) \\ -1 \end{bmatrix}$$

$$c_2 = u \qquad \mathbf{a}_2 \equiv \begin{bmatrix} a_{21} \\ a_{22} \\ a_{23} \end{bmatrix} = \begin{bmatrix} -\gamma E/\rho - \frac{1}{2}(\gamma+1)u^2 \\ u \\ -1 \end{bmatrix}$$

$$c_3 = u + a \qquad \mathbf{a}_3 \equiv \begin{bmatrix} a_{31} \\ a_{32} \\ a_{33} \end{bmatrix} = \begin{bmatrix} -u(\frac{1}{2}u - a/(\gamma-1)) \\ u - a/(\gamma-1) \\ -1 \end{bmatrix}$$

The speed of sound a is defined by

$$a^2 \equiv \gamma(\gamma-1)[E/\rho - \frac{1}{2}u^2]$$

The characteristic equations are then

$$\sum_{j=1}^{3} a_{ij}(u)\frac{du_j}{dt} = 0 \qquad \text{along} \qquad \frac{dx}{dt} = c_i(u) \qquad i = 1,2,3$$

Four examples have been chosen to illustrate the performance of the computer code. The first example is the solution of a Riemann problem. The second example shows the collision of two shocks of equal strength. In these first two numerical tests the results are compared to the exact solution. As a third example the formation of a shock is shown. A more complicated problem of interacting shocks, fans and contact discontinuities is given as the final example. Examples three and four are compared to the results obtained using a more standard finite difference code with many points.

3.2 Example 1 *Shock Tube*

This first example is taken from the paper by Sod [1978]. This Riemann problem has become a standard test case. The initial conditions are

$t = 0$	$x \leq .5$	$x \geq .5$
ρ	1.0	.125
m	0.0	0.0
E	2.5	.25

Table 3.1 Shock Tube Initial Conditions

The solution for times greater than zero consists of a shock wave travelling to the right followed by a contact discontinuity and a rarefaction wave. The density and energy (and momentum) are discontinuous across the contact, while the velocity and pressure are not.

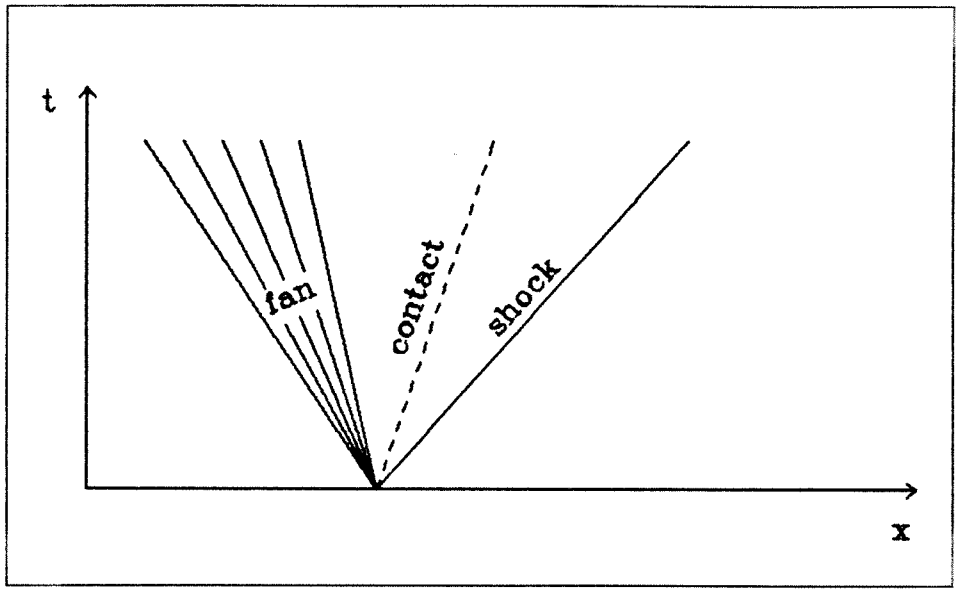
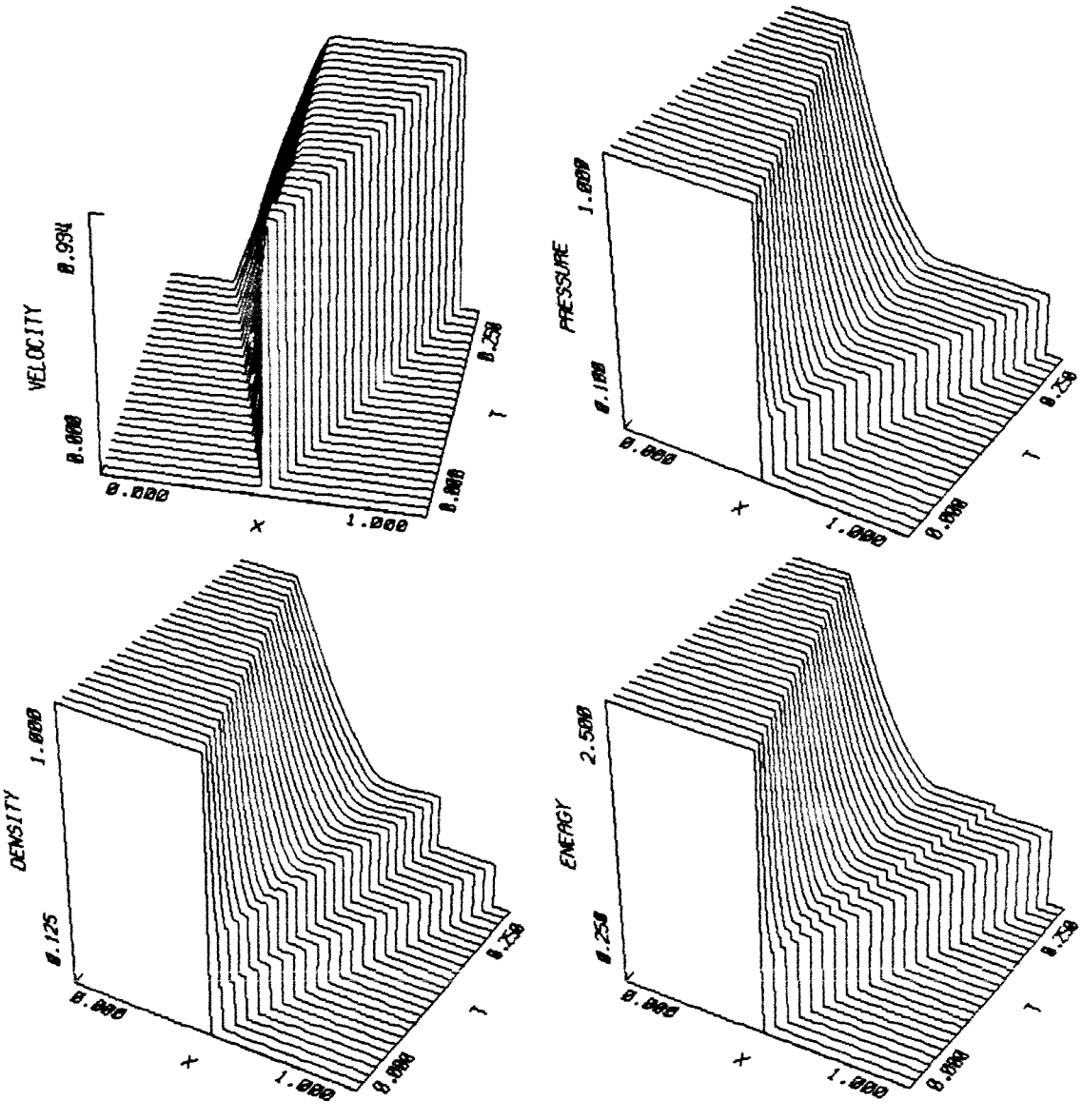


Figure 3.1 Shock Tube

The time evolution of ρ, u, E and p are shown in figure 3.2. Indicated on the figure are the time step dt, the number of points on the fixed grid n, and the number of characteristic points nc. nc is given as a range of values. This range indicates the maximum and minimum number of characteristic points that were needed over the entire run. Initially there is only one *group* of characteristic points. As the discontinuities separate the group becomes larger and the number of characteristic points increases. When the smooth regions in this group becomes large enough the group splits and the number of characteristic points decreases. By the final time shown there are three groups. Comparison to the exact solution is made at time $t = .2$ for $n = 40$ in figure 3.3.



Shock Tube

dt = 0.0100 n = 40 nc = 3 to 24

Figure 3.2 Shock Tube - Time Evolution

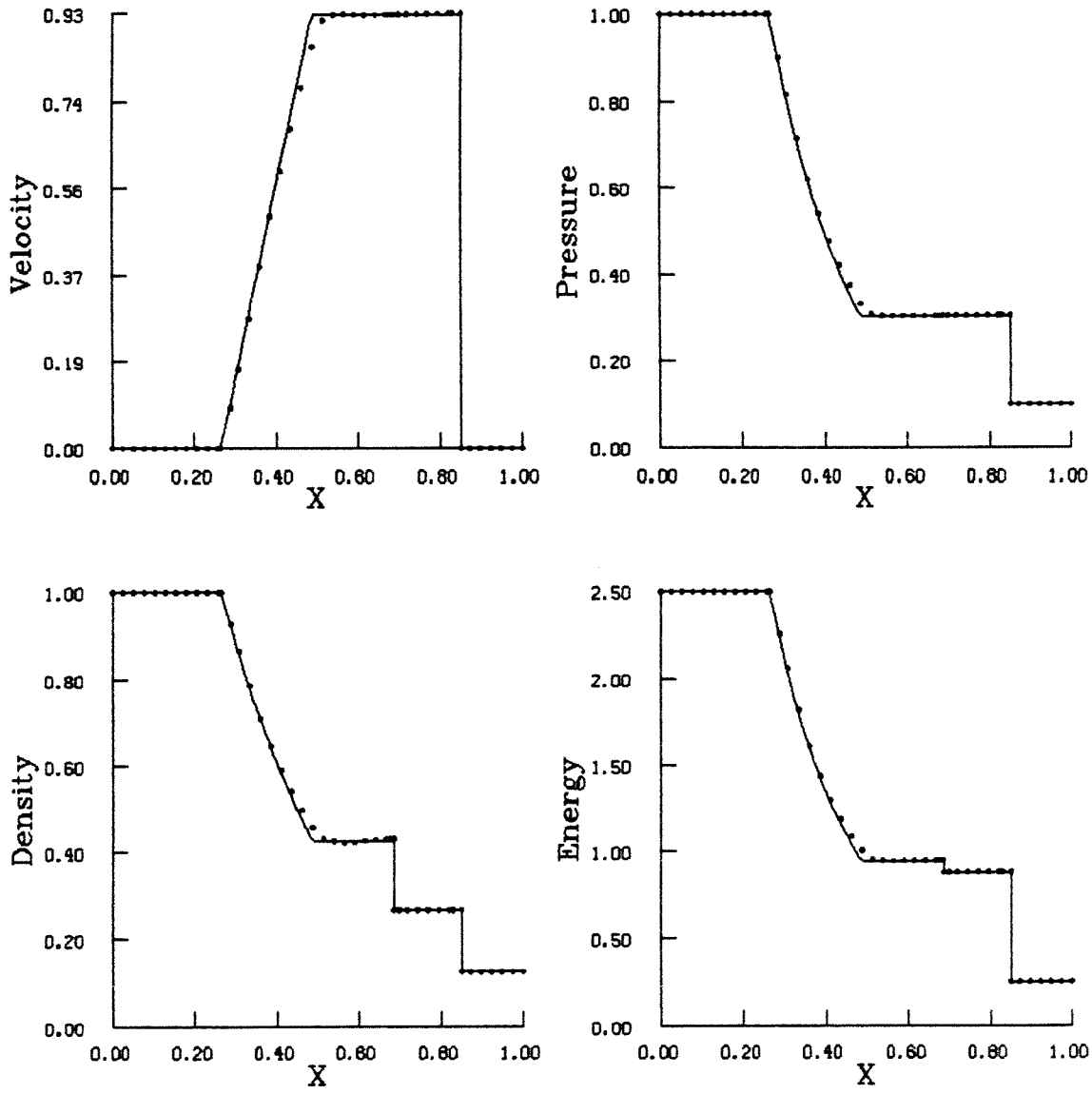


Figure 3.3 Shock Tube - Comparison to Exact Solution

3.3 Example 2 *Shock Collision*

This example was chosen to demonstrate how the program handles the collision of shocks. In particular we look at the collision of two shocks of equal strength travelling in opposite directions. The initial conditions are given as

$t = 0$	$x \leq .33$	$.33 \leq x \leq .67$	$x \geq .67$
ρ	.2656	.125	.2656
m	.2463	0.0	-.2463
E	.8720	.25	.8720

Table 3.2 Shock Collision Initial Conditions

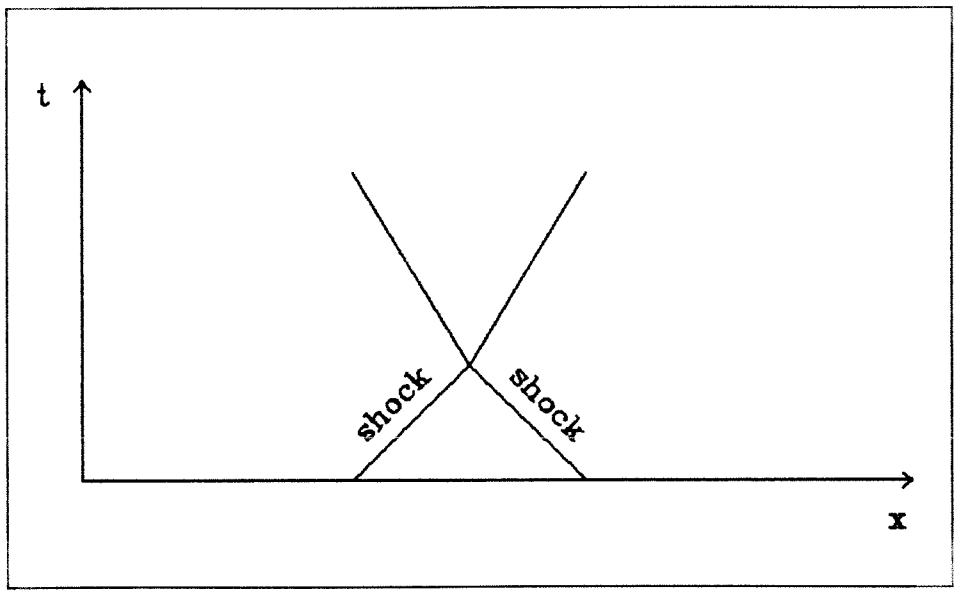


Figure 3.4 Shock Collision

Figure 3.5 shows the numerical solution proceeding in time. A comparison is made with the true solution at time $t = 2.5$ in figure 3.6. At this time the shocks have already collided and are now moving apart. Due to the way that interactions are handled (described in an earlier section) there is a slight error in the shock

position after the collision has taken place. (Before the collision the shock positions are exact up to the precision of the computer .)

3.4 Example 3 *Shock Formation*

This numerical experiment was performed to see how a smooth profile can steepen up to form a shock, figure 3.7. For the first few steps there are no characteristic points present. Once the shock starts to form, characteristic points are put in. Eventually the solution develops a jump. Many shock tracking codes would probably not be able to handle a shock formation problem since they require a priori knowledge of the positions of the shocks. The code developed here is well suited for this problem.

This solution is compared to the result using the Lax Wendroff method on a fixed grid with 400 points (figures 3.8 and 3.9). Artificial viscosity of the type developed by Lapidus is used in the Law Wendroff solution. The value of this artificial viscosity is given as *vnu* in the figures. It is seen that the method accurately predicts the shock development.

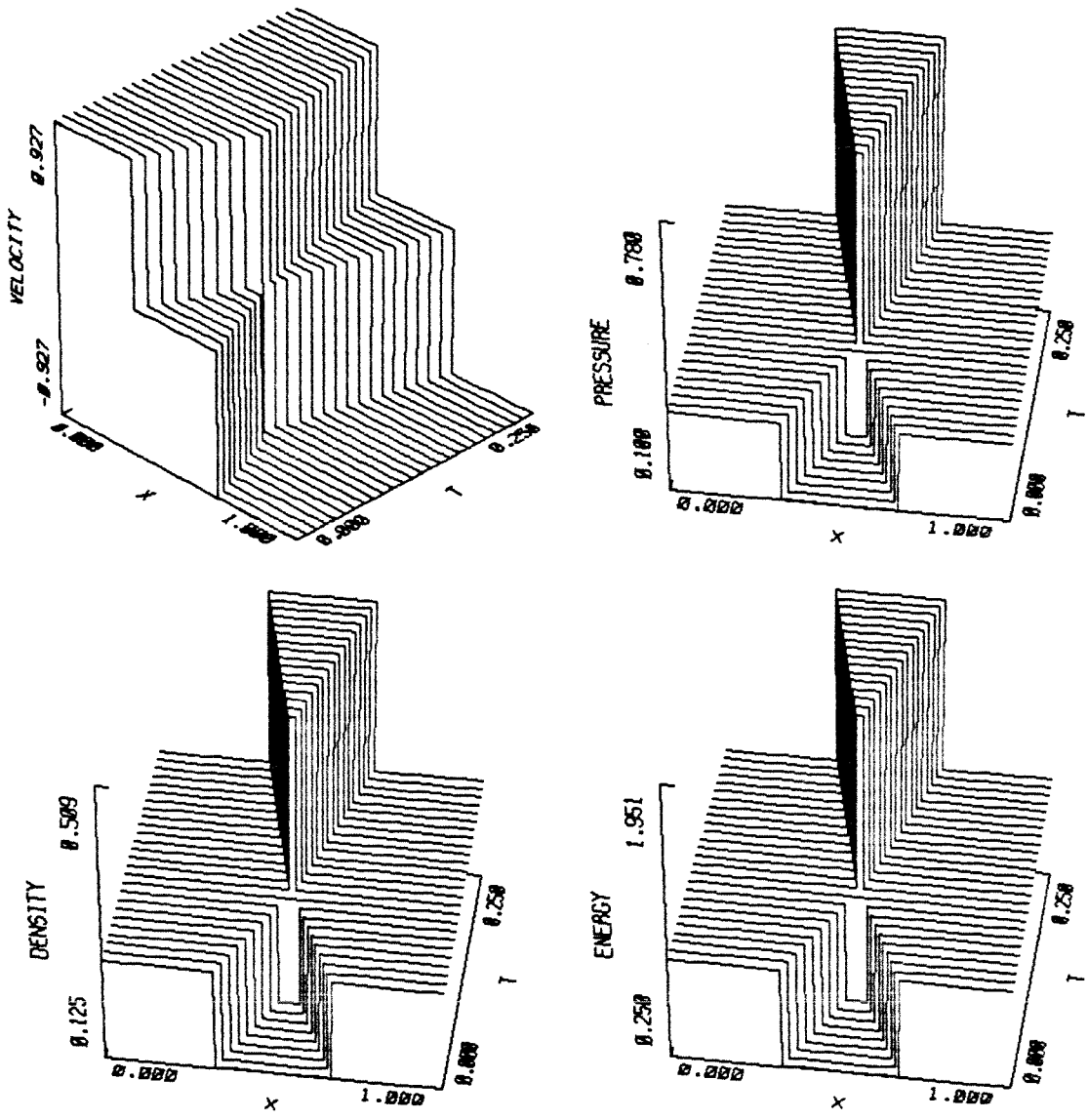
3.5 Example 4 *Interactions*

In this final example a shock moving from the right hits the flow that is generated by the shock tube problem of example 1. The initial conditions are

$t = 0$	$x \leq .5$	$.5 \leq x \leq .753$	$x \geq .753$
ρ	1.0	.125	.2656
m	0.0	0.0	$-.2463$
E	2.5	.25	.8720

Table 3.3 Initial Conditions for Shock Interactions

The resulting interactions of shocks and contact discontinuities are fairly complicatdd and are shown schematically in the following $x - t$ diagram. This should



Shock Collision

dt = 0.0100 n = 30 nc = 5 to 11

Figure 3.5 Shock Collision - Time Evolution

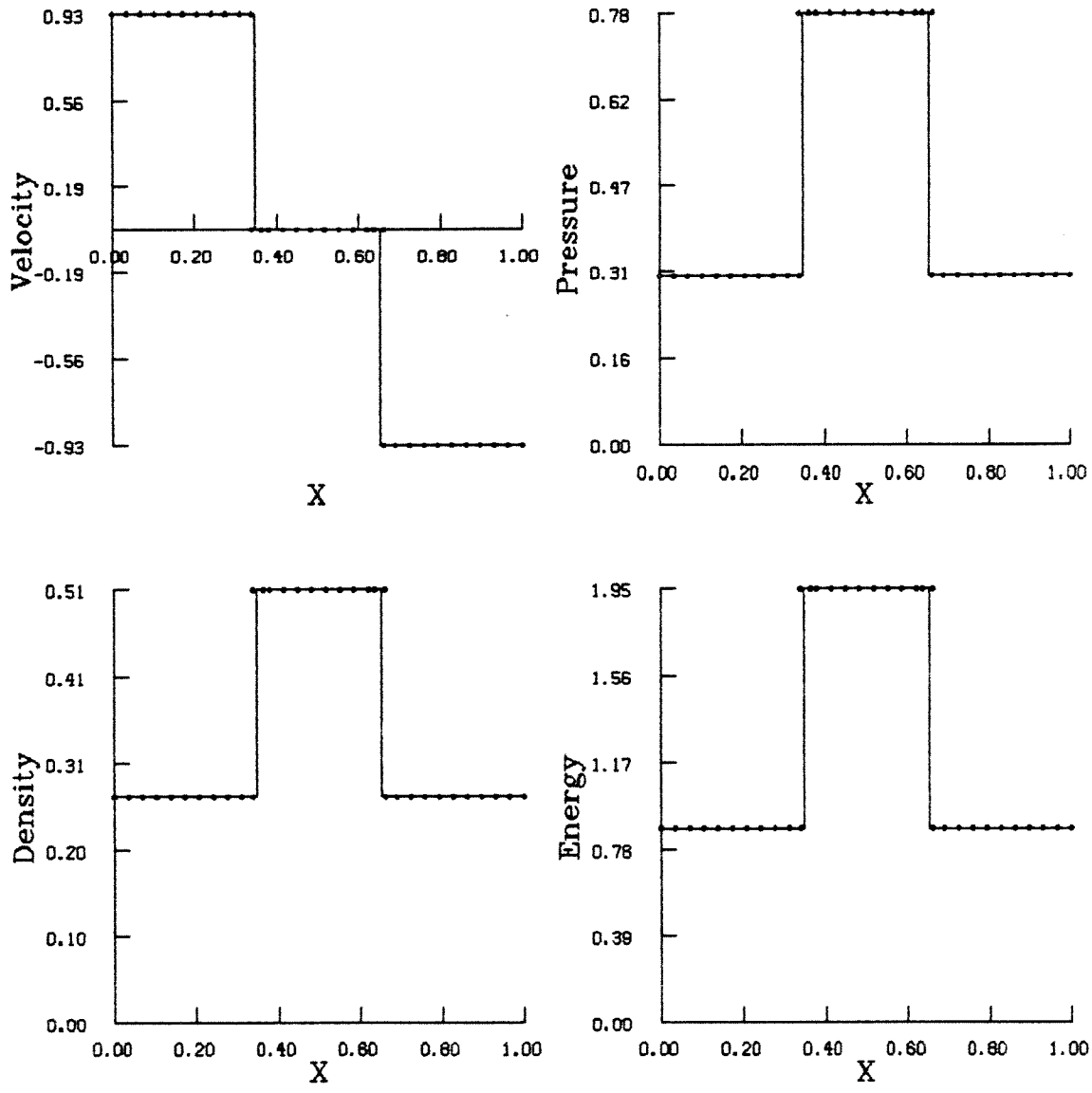
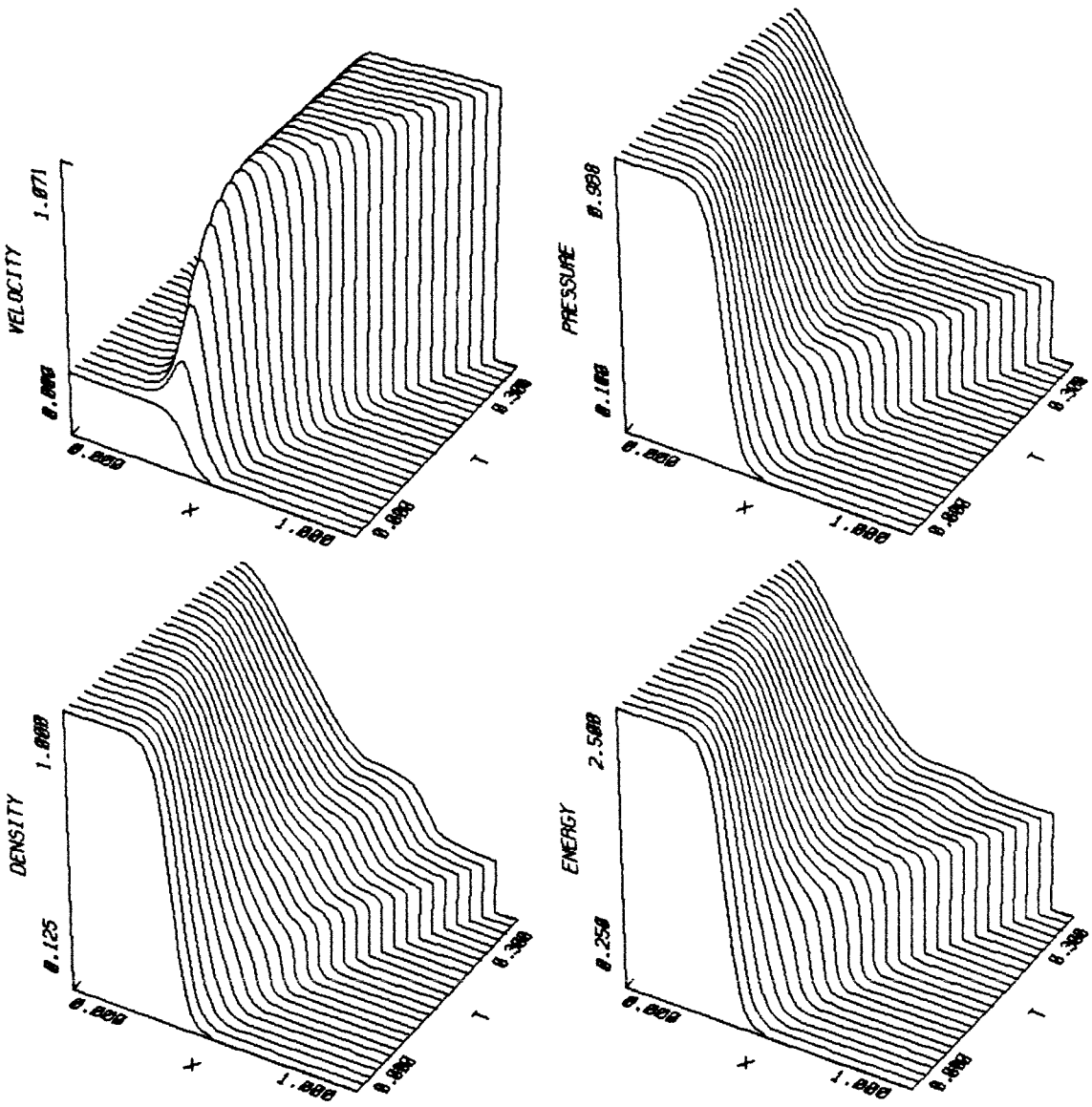


Figure 3.6 Shock Collision - Comparison to Exact Solution



Shock Formation

dt = 0.0040 n = 81 nc = 1 to 9

Figure 3.7 Shock Formation - Time Evolution

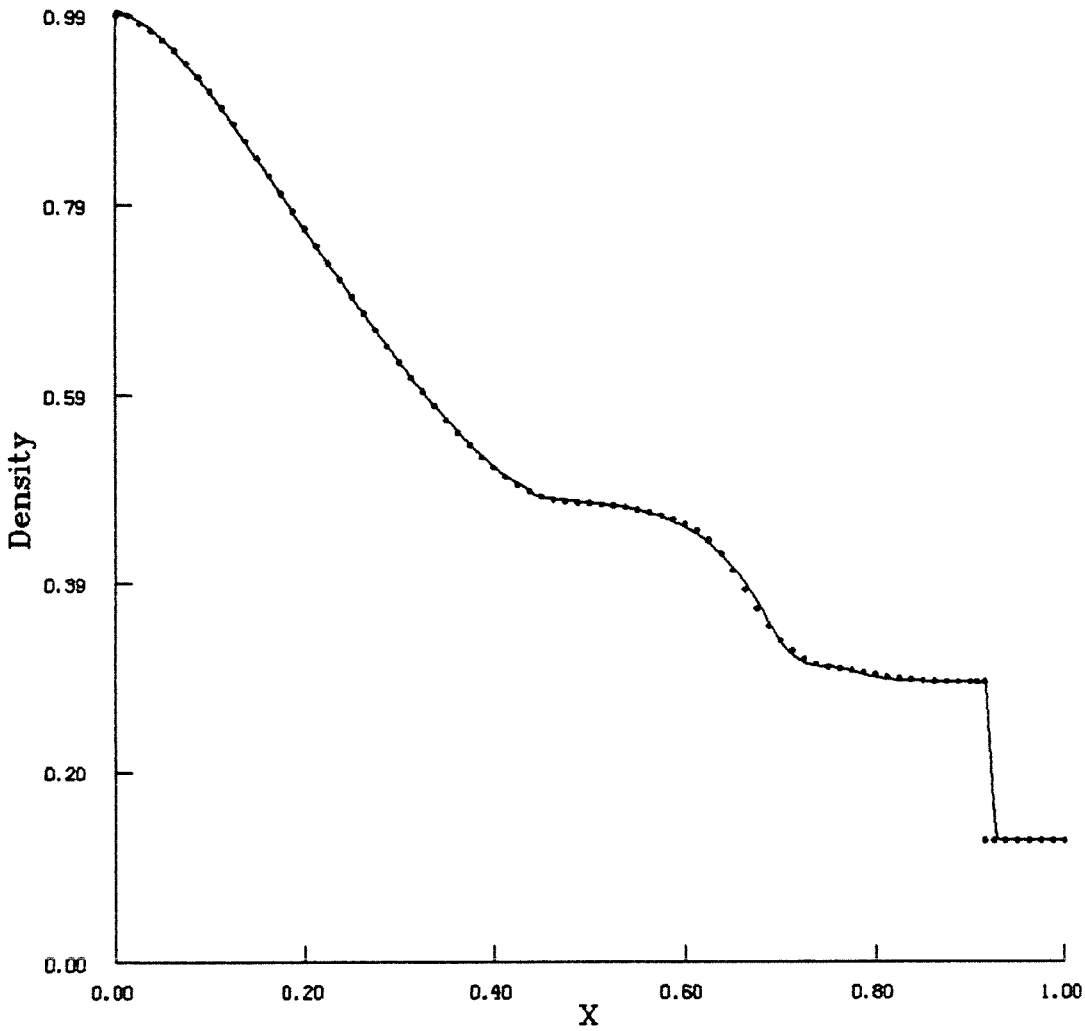


Figure 3.8 Shock Formation - Comparison to LW Solution

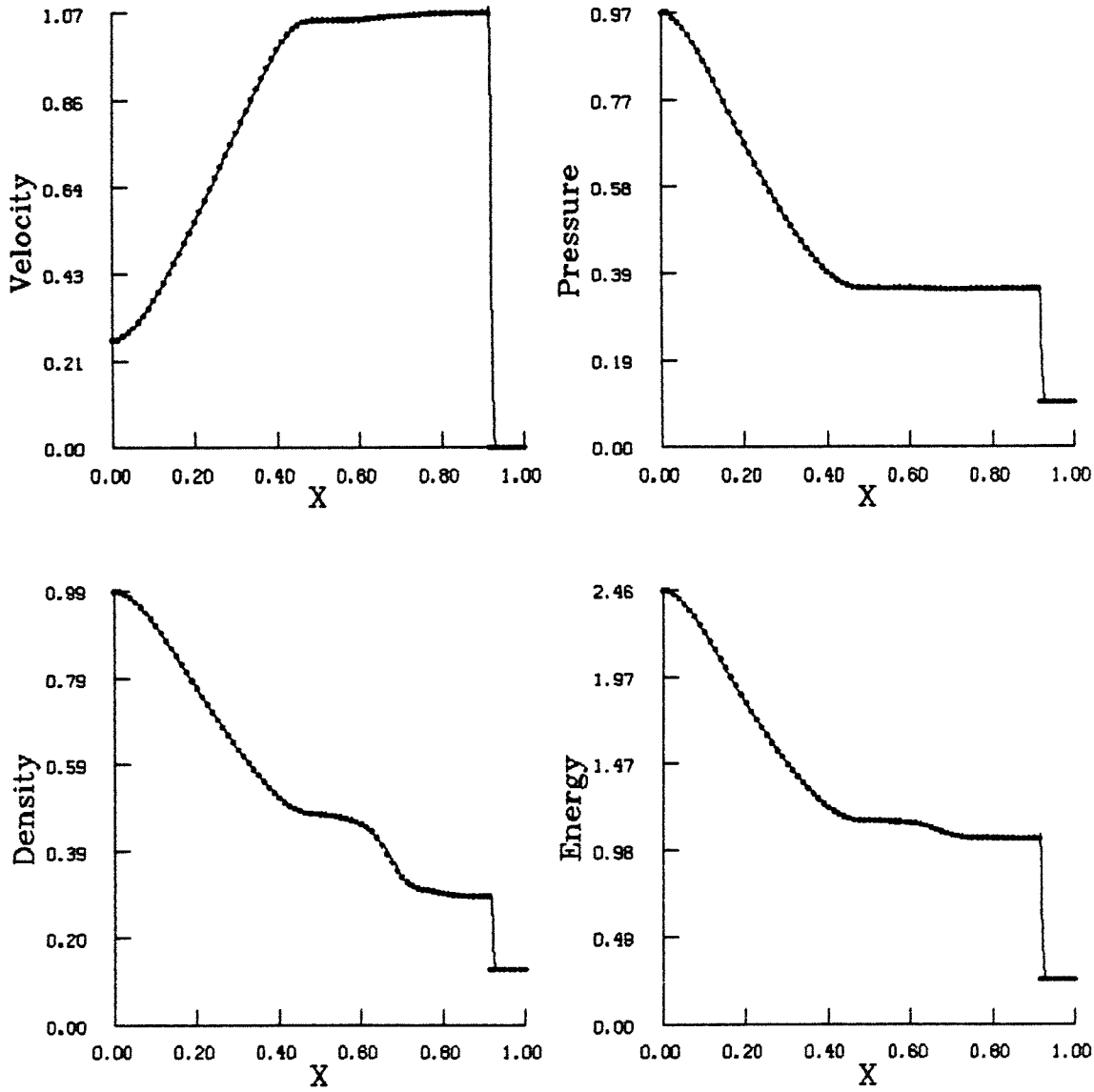


Figure 3.9 Shock Formation - Comparison to LW Solution

be helpful to follow the numerical solution of figure 3.11.

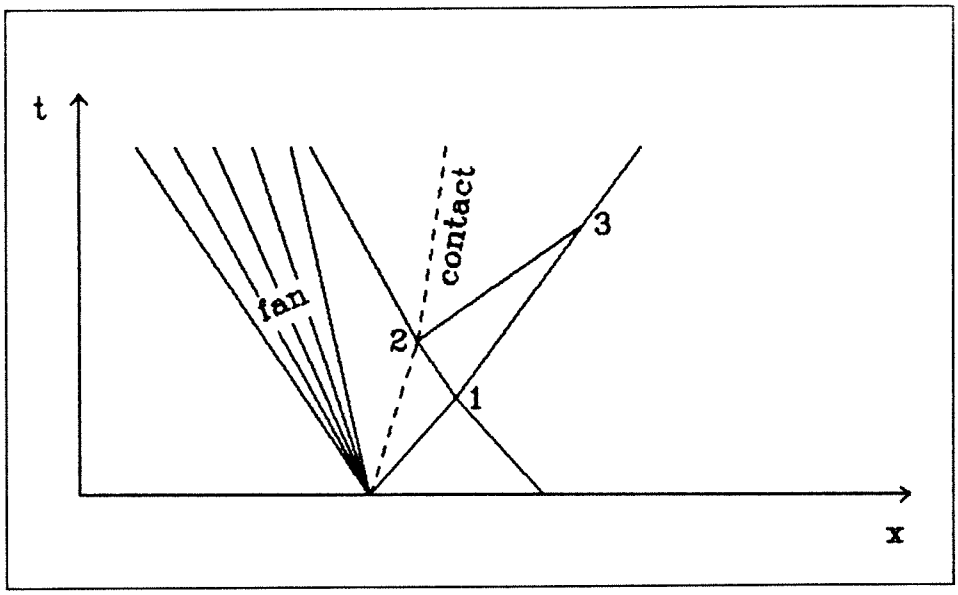


Figure 3.10 Shock Interactions

At the point marked (1) on the diagram the isolated shock which is moving in from the right side of the diagram hits the shock which began at $x = .5$ and is moving to the right. These shocks are of equal strength. They pass through each other after being refracted. The one shock continues to move to the left until it hits the contact discontinuity at (2) on the diagram. Note that in the figure 3.11 the contact discontinuity only appears in the density plot. When the shock and contact hit they pass through each other with some refraction. There is also a weak reflected shock which is generated from this collision. This reflected shock moves to the right and catches up with the other shock at (3).

Again the solution is compared to the result using Lax Wendroff and a large number of points, figures 3.12 and 3.13.

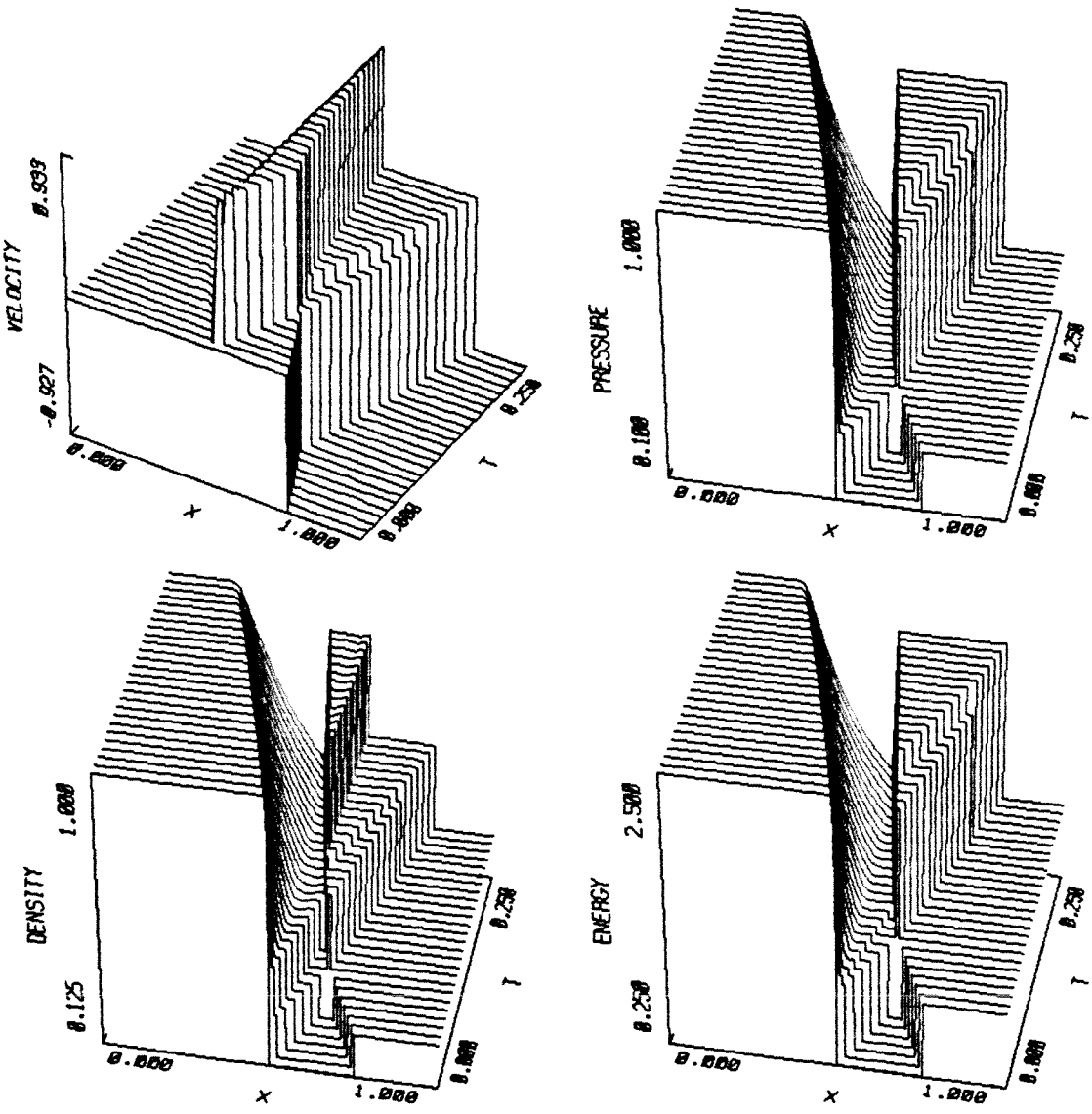


Figure 3.11 Shock Interactions - Time Evolution

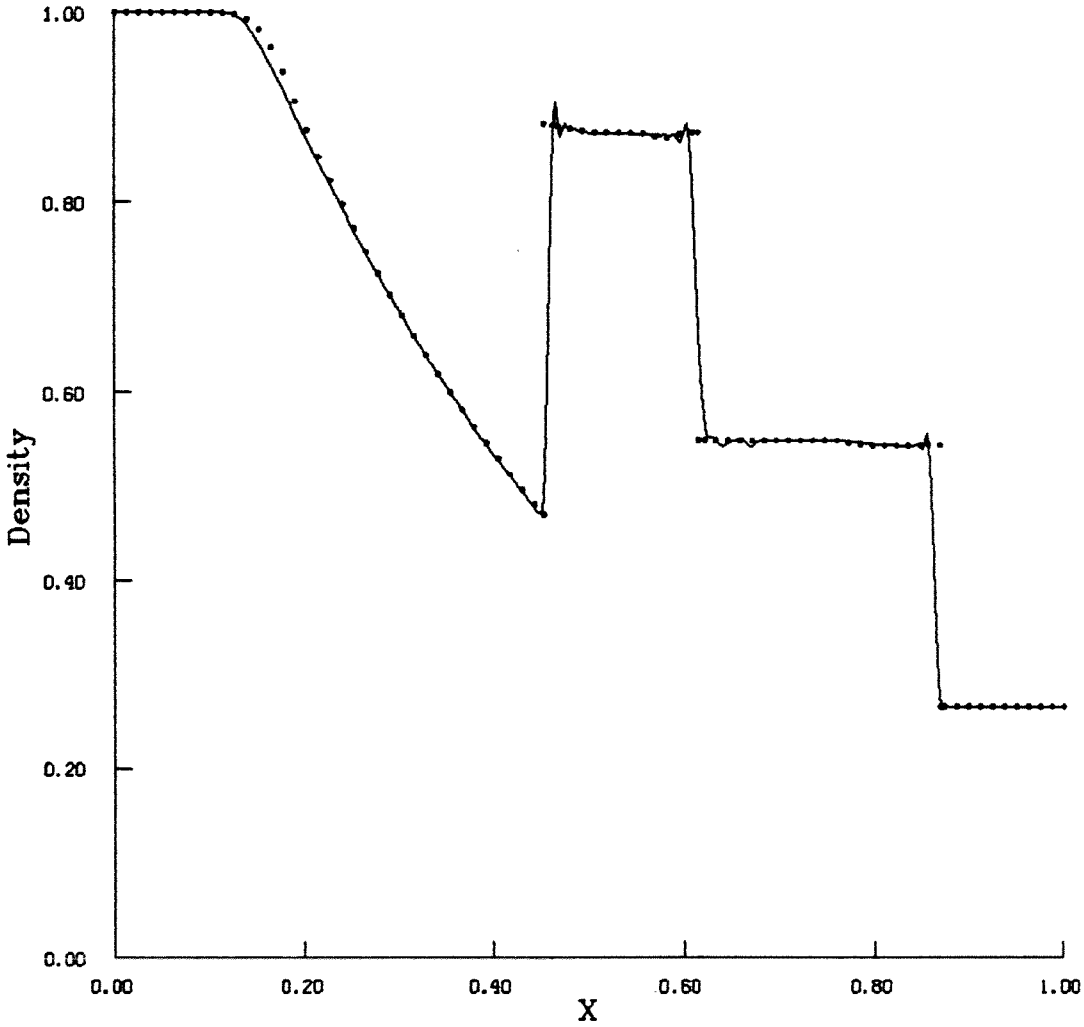


Figure 3.12 Shock Interactions - Comparison to LW Solution

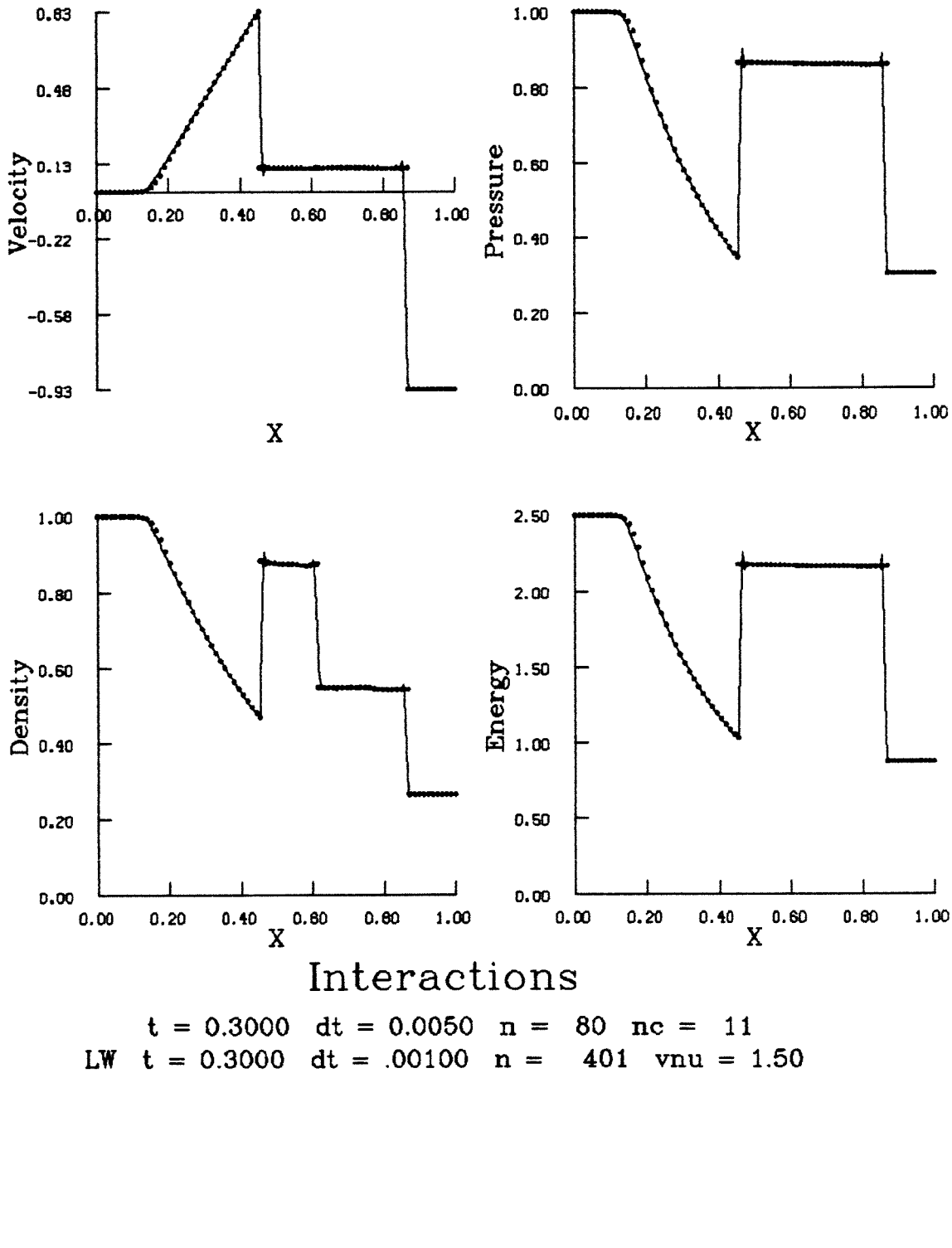


Figure 3.13 Shock Interactions - Comparison to LW Solution

References

M.J. Berger, *Adaptive Mesh Refinements for Hyperbolic Partial Differential Equations*, Ph.D.dissertation, Department of Computer Science, Stanford University, Report No. STAN-CS-82-924, 1982.

J.P. Boris and D.L. Book, *Flux Corrected Transport. I. SHASTA, A Fluid Transport Algorithm That Works*, J. Comput. Phys., **11**, p. 38, 1973.

J.U. Brackbill and J.S. Saltzman, *Adaptive Zoning for Singular Problems in Two Dimensions*, J. Comp. Phys. **46**, pp. 342-368, 1982.

D.L. Brown, *Solution Adaptive Mesh Procedures for the Numerical Solution of Singular Perturbation Problems*, Ph.D. thesis, California Institute of Technology, 1982.

A.J. Chorin, *Random Choice Solution of Hyperbolic Systems*, J. Comp. Phys. **22**, pp 517-533, 1976.

R. Courant and K.O. Freidrichs, *Supersonic Flow and Shock Waves*, Interscience Publishers, New York, 1948.

P. Colella and H.M. Glaz, *Numerical Modelling of Inviscid Shocked Flows of Real Gases*, Eight International Conference on Numerical Fluid Dynamics, Edited by E. Krause, Lecture Notes in Physics **170**, Springer-Verlag 1982.

P. Colella and P.R. Woodward, *The Piecewise-Parabolic Method (PPM) for Gas-Dynamical Simulations*, LBL report #14661, July 1982.

B. Engquist and S. Osher, *Upwind Difference Schemes for Systems of Conservation Laws, Potential Flow Equations*, Preprint, 1980.

J. Glimm, *Solutions in the Large for Nonlinear Hyperbolic Systems of Equations*, Comm. Pure Appl. Math., **18**, pp. 697-715, 1965.

S.K. Godunov, *A Finite Difference Method for the Numerical Computation of Discontinuous Solutions of the Equations of Fluid Dynamics*, Mat. Sb., **47**, pp. 271-290, 1959.

A. Harten, *The Artificial Compression Method for Computing Shocks and Discontinuities. I. Single Conservation Laws*, Comm. Pure Appl. Math., **30**, pp.611-638, 1977.

A. Harten and P.D. Lax, *A Random Choice Finite Difference Scheme for Hyperbolic Conservation Laws*, New York University Report DOE/ER/03077-167, May 1980.

A. Harten, P.D. Lax and Bram van Leer, *On Upstream Differencing and Godunov Type Schemes for Hyperbolic Conservation Laws*, Siam Review **25**, no. 1, January 1983.

A. Harten, J.M. Hyman and P.D. Lax, *On Finite-Difference Approximations and Entropy Conditions for Shocks*, Comm. on Pure and Appl. Math., vol. XXIX, pp 297-322, 1976.

J.M. Hyman, *Adaptive Moving Mesh Methods for Partial Differential Equations*, Los Alamos National Laboratory report LA-UR-82-3690, 1982.

J.M. Hyman, *Numerical Methods for Tracking Interfaces*, Los Alamos National Laboratory Report LA-9917-MS, March 1984.

S.N. Kruzkov, *First Order Quasi-linear Equations in Several Independent Variables*, Math. USSR Sb., **10**, pp. 217-243, 1970.

Lapidus, *A Detached Shock Calculation by Second Order Finite Differences*, J. Comp. Phys. **2**, pp.154-177, 1967.

P.D. Lax, *Hyperbolic Systems of Conservation Laws and the Mathematical Theory of Shock Waves*, SIAM Regional Conf. Series Lectures in Appl. Math., **11**, 1972.

P.D. Lax and B. Wendroff, *Systems of Conservation Laws*, Comm. Pure Appl. Math., **13**, pp. 217-237, 1960.

P. Lötstedt, *A Front Tracking Method Applied to Burger's Equation and Two-Phase Porous Flow*, J. Comp. Phys, **47**, pp. 211-228, 1982.

Ni Lin-an and Wu Xiong-hua, *The Singularity-Separating Method*, The Proceedings of the Fourth International Symposium on Finite Element Methods in Flow Problems, July 26-29, 1982.

O.A. Oleinik, *Discontinuous Solutions of Nonlinear Differential Equations*, Uspekhi Mat. Nauk, **12**, pp. 3-73, 1957. (Amer. Math. Soc. Transl. Ser. 2, 26, pp. 95-172)

S. Osher and S. Chakravarthy, *High Resolution Schemes and the Entropy Condition*, SIAM J. Numer. Anal. **21**, no. 5, October 1984

S. Osher and F. Solomon, *Upwind Difference Schemes for Systems of Conservation Laws*, Preprint 1980.

B. Plohr, J. Glimm, O. McBryan, *Application of Front Tracking to Two Dimensional Gas Dynamics Calculations*, Courant Institute Report, 1983.

R.D. Richtmeyer and K.W. Morton, *Difference Methods for Initial Value Problems*, 2nd Edition, Interscience, New York, 1967.

P.L. Roe, *The Use of the Riemann Problem in Finite Difference Schemes*, Royal Aircraft Establishment, Bedford, England, 1980.

P.L. Roe, *Approximate Riemann Solvers, Parameter Vectors and Difference Schemes*, J. Comp. Phys., **43**, pp. 357-372, 1981.

G. Sod, *A Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws*, J. Comp. Physics, **27**, 1978.

B. van Leer, *Towards the Ultimate Conservative Difference Scheme, V. A Second-order Sequel to Godunov's Method*, J. Comp. Phys., **32**, pp. 101-136, 1979.

B. van Leer, *Towards the Ultimate Conservative Difference Scheme, II. Monotonicity and Conservation Combined in a Second-order Scheme*, J. Comp. Phys., **14**, pp. 361-370, 1974.

J. Von Neumann and R.D. Richtmeyer, *A Method for the Numerical Calculations of Hydrodynamic Shocks*, J. Appl. Phys. **21**, p. 232, 1950.

G.B. Whitham, *Linear and Nonlinear Waves*, John Wiley and Sons, Inc., 1974.

P. Woodward and P. Colella, *The Numerical Simulation of Two Dimensional Fluid Flow with Strong Shocks*, J. Comp. Phys. **54**, pp. 115-173, 1984.

S.T. Zalesak, *Fully Multidimensional Flux-Corrected Transport Algorithms for Fluids*, J. Comp. Phys., **31**, pp. 335-362, 1979.

Part II: Composite Overlapping Grid Techniques

Chapter 1

Introduction

Composite overlapping meshes are very useful in the numerical solution of partial differential equations. This is especially true on regions of nontrivial shape. The basis of the technique is to cover the computational domain with a number of meshes. Where the meshes meet they overlap a few grid lines, rather than joining up exactly along their boundaries. Solution values on these overlapping mesh boundaries are obtained by interpolation from the solution values of the mesh which is overlapped. For example, figure 1.1 shows a region Ω which has been covered by two grids.

A curvilinear grid follows the boundary and a rectangular grid covers the interior. The *component* meshes are also shown . There are a number of advantages to using the composite mesh procedure :

(1) Accurate representation of boundaries

(2) Ease of applying boundary conditions

(3) Mesh refinement to resolve certain parts of the region is made easiser

(4) Flexibility in the grid construction. The curvilinear grid in figure 1.1 can be constructed in a manner essentially independent of the inner rectangular grid. This is in contrast to many methods which construct a single grid for the entire region.

(5) The method lends itself naturally to parallel processing.

Composite overlapping mesh methods are illustrated in the numerical solution of an oceanographic problem. This problem will illustrate methods for the numerical solution of time dependent and elliptic equations on composite meshes. The

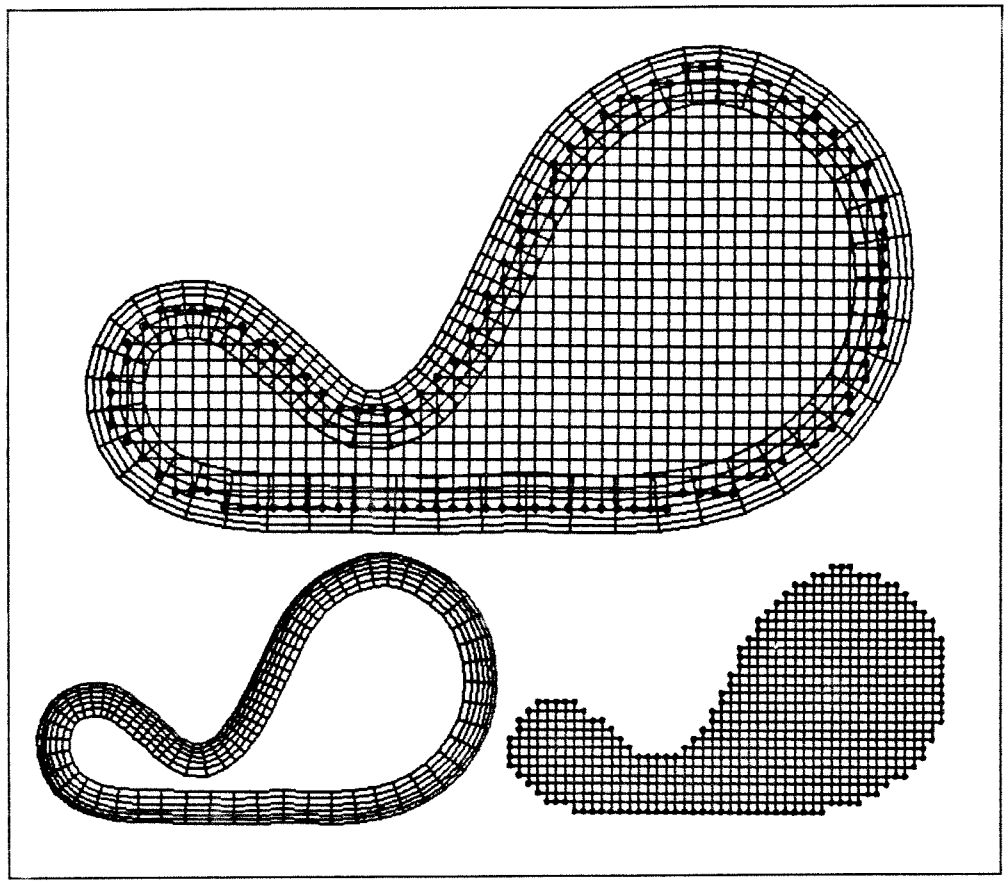


Figure 1.1 Overlapping Grids

following set of partial differential equations is to be solved on a region Ω. Ω can be thought of as an ocean basin.

$$\zeta_t = -R_0[u\zeta_x + v\zeta_y] - \beta v + E_k \nabla^2 \zeta + f \tag{1.1a}$$

$$\nabla^2 \psi = \psi_{xx} + \psi_{yy} = \zeta \tag{1.1b}$$

$$(u, v) = (-\psi_y, \psi_x) \tag{1.1c}$$

$$\psi = 0 \quad \text{and} \quad \frac{\partial \psi}{\partial n} = 0 \quad \text{on} \quad \partial\Omega$$

$\partial\Omega$ denotes the boundary of Ω. Partial differentiation with respect to time and space are denoted by subscripts.

$$\zeta_t = \frac{\partial \zeta}{\partial t}$$

$$\zeta_x = \frac{\partial \zeta}{\partial x}$$

These are the quasigeostrophic barotropic vorticity equations for flow on a beta plane. They are a model of large scale ocean flows. The parameters R_0 and E_k will be called the Rossby number and Ekman number, respectively. We shall refer to these equations as the *ocean equations* or the *vorticity stream function* equations. When the system is nondimensionalized so that the size of the region is $O(1)$ and the size of ψ is $O(1)$ typical values for R_0, E_k, β, and f are

$$R_0 = 10^{-2} \quad \text{to} \quad 10^{-3}$$

$$E_k = 10^{-4} \quad \text{to} \quad 10^{-6}$$

$$\beta = 1$$

$$f = 1$$

Of interest will be the effect of boundary geometry upon flows of small Ekman number. For small E_k strong boundary layers develop. It thus seems essential to have a grid which smoothly follows the boundary.

We now give a short outline of the contents of the chapters to follow. In chapter two composite meshes are described. This includes introducing the notation and the concepts of composite mesh functions and composite mesh operators. In chapter three the ocean equations are discussed. The form of the expected solution in boundary layers is derived. This indicates the appropriate stretching to use for the grids which lie next to the boundary. In addition, the time marching scheme that is used to solve the ocean equations is described. The fourth chapter is concerned with the solution of Poisson's equation on composite overlapping meshes. Both direct and iterative methods are discussed. A major section of chapter four deals with the multigrid algorithm as it applies to composite meshes. Chapter five presents the analysis of some model problems which relate to other parts of this work. The first model problem examines a one dimensional overlapping grid. The accuracy of the solution to an elliptic problem is examined. In particular we see how the amount of overlap between the grids and the order of interpolation at the

overlapping boundaries affects the error. The second model analysis examines the numerical boundary conditions which are used for the vorticity stream function equations. A reason is given to explain why higher order boundary conditions are needed. The final chapter, six, gives some numerical results from the multigrid solver and the solution of the ocean equations. As many of the ocean model codes approximate the boundary in a somewhat more crude fashion it is of interest to see what differences there are when composite meshes are used. Such a comparison is made in this final chapter.

Chapter 2

Composite Meshes

The composite overlapping grid technique has been in use for some time. For example, Starius [1977] looked at the convergence of elliptic problems on two overlapping meshes using the Schwartz alternating procedure. (This procedure is explained later.) In a later paper Starius [1980] considered the numerical solution of hyperbolic problems. The stability of the Lax Wendroff method was shown for a model problem on a one dimensional overlapping mesh. In his Ph.D thesis, Reyna [1982] obtained further stability results for the method of lines, again for a model hyperbolic problem on a one dimensional overlapping mesh. Reyna also gave examples and accuracy tests of the method. A method of mesh construction for two overlapping grids was described by B. Kreiss, [1983]. We use a similar method to the one described there. A more general program for generating composite meshes has been developed by G. Chesshire, [1985]. This program allows the interactive description of the problem from which a composite mesh is generated. Also on interest is a paper by M. Berger [1984] where she indicates how to obtain conservative difference approximations at grid interfaces.

2.1 Notation

In this section the notation used to described composite meshes is introduced. A good notation makes it much easier to describe algorithms on composite meshes. This is especially true when we need to describe the multigrid algorithm. A composite mesh consists of a union of (usually) simpler meshes. These simpler meshes will often be called *component* meshes. One can define *composite mesh functions* in the same way that mesh functions are defined on other types of grids. Similarly *composite mesh operators* can be defined. An *overlapping* composite mesh has com-

ponent meshes which overlap where they meet. For such composite grids we often deal with mesh functions which are related at the boundaries of the overlap to the mesh function values on the component grid which is being overlapped (through an interpolation formula say).

2.1.1 Composite Meshes

A composite mesh $\mathbf{M}$ is composed of component meshes M_p.

$$\mathbf{M} = M_1 \cup M_2 \cup \ldots$$

In general a component mesh need not be simple in structure, and could in principle be a composite mesh itself. For our purposes, however, we will assume that each component mesh consists of a set of mesh points.

$$M_p = \{ \underline{x}_p(i,j) \mid (i,j) \in N_p \}$$

$N_p \subset \mathbf{Z} \times \mathbf{Z}$ is a set of index pairs which specify the points on the component mesh. A component mesh will consist of boundary points and interior points. A point on the boundary, ∂M_p, of a component mesh will either be on that part of the boundary which is interpolated from another grid or else on the boundary of the computational region. The set of interpolation boundary points of M_p will be denoted by ιM_p. (The *iota* (ι) prefix will be used when an interpolation boundary is referenced.)

$$\iota M_p = \{ \underline{x}_p(i,j) \mid (i,j) \in \iota N_p \}$$

The set of all interpolation boundary points on $\mathbf{M}$ is $\iota\mathbf{M}$.

2.1.2 Composite Mesh Functions

A composite mesh function $\mathbf{f}$ is a function defined on the composite mesh $\mathbf{M}$. It is a union of component mesh functions $\underline{f}_p$.

$$\mathbf{f} = \underline{f}_1 \cup \underline{f}_2 \cup \ldots$$

Each component mesh function can be represented as a set of values defined at the mesh points of M_p,

$$\underline{f}_p = \{ f_p(i,j) \,|\, (i,j) \in N_p \}$$

Those values which appear on the interpolation boundary ιM_p will be denoted by $\iota \underline{f}_p$. The space of all composite mesh functions will be called **Mf** and the space of component mesh functions Mf_p

$$\mathbf{Mf} = \{ \mathbf{f} \,|\, \underline{f}_p \in \mathrm{Mf}_p \}$$

$$\mathrm{Mf}_p = \{ \underline{f}_p \,|\, f_p(i,j) \in \mathbb{C} \}$$

2.1.3 Composite Mesh Operators

A composite mesh operator **S** maps one or more composite mesh functions into a composite mesh function.

$$\mathbf{S} : \mathbf{Mf}^n \rightarrow \mathbf{Mf}$$

Composite mesh operators which act only on those values of the mesh function which correspond to the points on the interpolation boundaries will be denoted with a preceding ι, for example $\iota\mathbf{S}$. A composite mesh operator consists of component mesh operators. A component mesh operator maps one or more component mesh functions into another component mesh function.

$$S_p : \mathrm{Mf}_p{}^n \rightarrow \mathrm{Mf}_p$$

The subscript indicates which component mesh the operator acts on. Composite mesh operators will be written in terms of their component mesh operators as follows

$$\mathbf{S} = S_1 \cdot T_2 \cdot S_1 \ldots \cdot S_p$$

The order in which the component mesh operators appear may be important. The operators act from right to left.

2.2 A Two Component Composite Mesh

We now look at the solution of a simple hyperbolic problem on a region which has been covered by composite meshes. Consider for example the composite overlapping mesh of figure 1.1. A rectangular grid M_1 covers an interior portion of domain. A curvilinear mesh M_2 lies on an annulus next to the boundary of Ω. The rectangular grid and curvilinear mesh overlap. Functions defined on the composite mesh such as ψ and ς are connected between M_1 and M_2 by interpolation. The discussion here will mainly deal with the case of two overlapping meshes, although the extension to any number of meshes is not difficult. The partial differential equations (PDEs) to be solved must be discretized on the meshes M_p, $p = 1$ or 2. One way to do this is to first map each component mesh onto a simpler region such as a unit square. The PDEs can then be transformed to the coordinates of this square. The physical coordinates will usually be denoted by (x, y) and the transformed coordinates by (r, s)

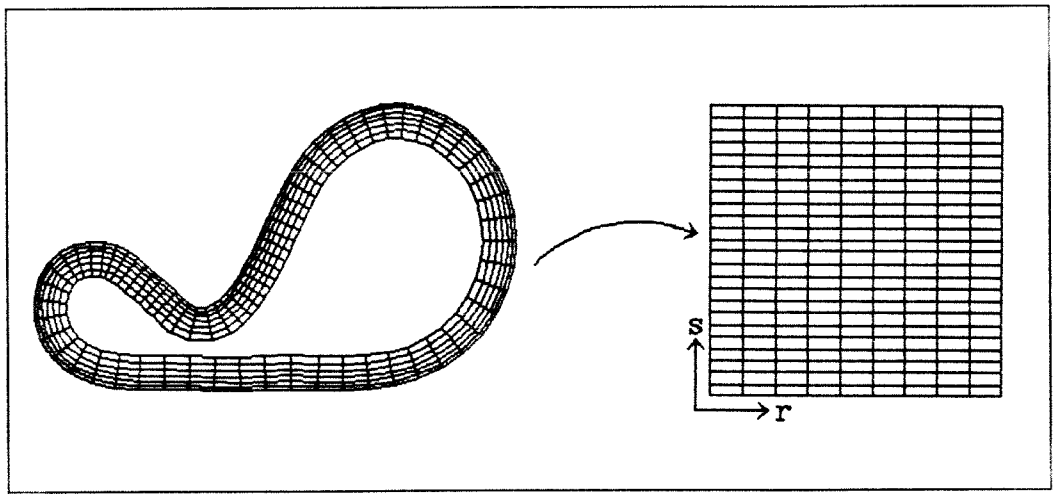


Figure 2.1 Transformation

As an illustration of the technique for solving time dependent problems on

composite meshes consider trying to solve

$$u_t = au_x + bu_y \tag{1}$$

where for simplicity boundary conditions will be ignored. For each mesh M_p the equation is written in (r, s) coordinates

$$\begin{aligned} u_t &= \tilde{a}_p u_r + \tilde{b}_p u_s \\ \tilde{a}_p &= a_p r_x + b_p r_y \\ \tilde{b}_p &= a_p s_x + b_p s_y \end{aligned} \tag{2}$$

Let the composite mesh function $\mathbf{v}(n)$ be an approximation to the solution u at time $t = n\Delta t$.

$$\mathbf{v}(n) = \underline{v}_1(n) \cup \underline{v}_2(n)$$

$$\underline{v}_p(n) = \{ v_p(i,j,n) \}$$

$$v_p(i,j,n) \approx u(x_p(i,j), y_p(i,j), t_n) \quad \text{for} \quad (x_p(i,j), y_p(i,j)) \in M_p \quad p = 1,2$$

The procedure to advance from time t_n to time $t_{n+1} = t_n + \Delta t$ is (using leap frog in time and central differences in space)

$$\begin{aligned} v_p(i,j,n+1) &= v_p(i,j,n-1) \\ &\quad + 2\Delta t \left[\tilde{a}_p D_{0r} + \tilde{b}_p D_{0s} \right] v_p(i,j,n) \quad (i,j) \in N_p \quad p = 1,2 \end{aligned}$$

$$v_1(i,j,n+1) = \sum_{k,l} \alpha_1(i,j,k,l) v_2(k,l,n+1) \qquad (i,j) \in \iota N_1 \tag{3}$$

$$v_2(i,j,n+1) = \sum_{k,l} \alpha_2(i,j,k,l) v_1(k,l,n+1) \qquad (i,j) \in \iota N_2$$

where the central difference operators D_{0s} and D_{0r} are given by

$$\begin{aligned} D_{0s} v(i,j) &= \frac{v(i+1,j) - v(i-1,j)}{2h_s} \\ D_{0r} v(i,j) &= \frac{v(i,j+1) - v(i,j-1)}{2h_r} \end{aligned}$$

As indicated, values on the boundary of M_1 are obtained by interpolation from values of M_2 and vice versa. The $\alpha_p(i,j,k,l)$ are the interpolation weights. If there is sufficient overlap so that boundary points are interpolated entirely from interior points of the opposite grid then the interpolation formula can be evaluated directly using the calculated values from the interior. Otherwise interpolation values are linked together and a system of equations must be solved (i.e. the interpolation equations) in order to determine those values on the interpolation boundaries. This is discussed in further detail in chapter 5. Equations (3) define a composite mesh operator for the leap frog scheme, $\mathbf{LF} : \mathbf{Mf} \times \mathbf{Mf} \rightarrow \mathbf{Mf}$,

$$\mathbf{v}(n+1) = \mathbf{LF}(\mathbf{v}(n), \mathbf{v}(n-1))$$

Chapter 3

The Ocean Equations and Time Marching

In this chapter we discuss the ocean equations and time marching. The ocean equations are first nondimensionalized and scaled so the expected size of the solution is $O(1)$. This makes it easier to see the relative sizes of each of the terms in the equation. For purposes of grid stretching the behaviour of the solution in boundary layers is determined. Explicit and implicit time marching schemes are outlined in the final section of this chapter.

3.1 Scaling

The quasigeostrophic barotropic potential vorticity equations are a simple homogeneous model of wind driven ocean circulation. See, for example, Pedlosky [1982] for a derivation. In dimensional form these equations can be written as

$$\zeta_t = \psi_y \zeta_x - \psi_x \zeta_y - \beta(y)\psi_x + A_H \nabla^2 \zeta + f(x,y,t) \tag{1}$$

A_H is the ***horizontal turbulent viscosity coefficient.*** We introduce nondimensional variables which will be denoted with a superscript ***star.*** The dimensional variables will be scaled by combinations of the three parameters L, F and B. L is a length scale on the order of the size of the ocean basin. F measures the size of the forcing term and B the size of $\beta(y)$.

$$F = ||f|| = \max_{x,y,t} |f(x,y,t)|$$

$$B = ||\beta||$$

For the moment a scaling Ψ for the stream function and T for the time will be used. These parameters will later be chosen in terms of L, F and B.

$$x^* = \frac{x}{L} \qquad y^* = \frac{y}{L}$$

$$t^* = \frac{t}{T} \qquad f^* = \frac{f}{F}$$

$$\psi^* = \frac{\psi}{\Psi} \qquad \beta^* = \frac{\beta}{B}$$

$$\varsigma^* = \psi^*_{x^*x^*} + \psi^*_{y^*y^*}$$

Substituting these expressions into (1) gives

$$\begin{aligned}\frac{\Psi}{TL^2}\varsigma^*_{t^*} = \frac{\Psi^2}{L^4}(\psi^*_{y^*}\varsigma^*_{x^*} - \psi^*_{x^*}\varsigma^*_{y^*}) - \frac{B\Psi}{L}\beta^*\psi^*_{x^*} \\ + \frac{A_H\Psi}{L^4}(\varsigma^*_{x^*x^*} + \varsigma^*_{y^*y^*}) + Ff^*\end{aligned} \tag{2}$$

The parameters Ψ and T will now be chosen so that the Coriolis term and the forcing term are the same order of magnitude. The reason for this choice being that in the problems we are interested in the main balance in the equations will tend to be between these two terms (at least away from boundaries). Dividing through by F and setting the coefficients of the forcing term and Coriolis term to 1 leads to the choice for T and Ψ.

$$T = \frac{1}{BL} \qquad \Psi = \frac{FL}{B}$$

Define the quantities R_0 and E_k to be the coefficients of the convection and dissipation terms respectively.

$$R_0 = \frac{F}{(BL)^2}$$

$$E_k = \frac{A_H}{L^3B}$$

These nondimensional numbers will be called the Rossby and Ekman numbers, respectively. The Rossby number measures the relative importance of the convective (advective) terms. The Ekman number (which is usually small) measures the importance of dissipation and will determine the size of boundary layers. Dropping the *stars* from the nondimensional variables results in the nondimensional equation.

$$\varsigma_t = R_0(\psi_y\varsigma_x - \psi_x\varsigma_y) - \beta(y)\psi_x + E_k\nabla^2\varsigma + f(x,y,t) \tag{3}$$

Some typical choices for L, F, B and A_H are

$$L = 10^6 m$$

$$F = 10^{-11}s^{-2}$$

$$B = 2\times 10^{-11}m^{-1}s^{-1}$$

$$A_H = 2000m^2s^{-1}$$

which gives values for the Rossby number and Ekman number of

$$R_0 = 2.5 \times 10^{-2} \qquad E_k = 10^{-4}$$

3.2 Approximate Solutions and Coordinate Stretching

From the typical values for R_0 and E_k given in the previous section it is not hard to see that much of the flow will be determined by a balance between the Coriolis term and the forcing. However, since the Ekman number is small one can expect that narrow boundary layers will be present. It is useful when solving a problem numerically to know the form of the solution in the boundary layer so that the grid can be stretched to place more points in the regions of rapid variation. Ideally the solution on the stretched grid should be everywhere smooth. One can use asymptotic analysis to determine the form of the solution in boundary layers. For small R_0 it is also possible to obtain an approximate (steady) global solution. This solution was first discussed by Munk [1950]. It possesses some of the main characteristics of ocean flows including the intense currents on the western shores. A more detailed discussion can be found in Pedlosky [1982]. For our purposes we start from the following model problem. This equation models the behaviour of steady, low Rossby number flows on regions of the ocean where the flow is mainly one dimensional with boundaries running essentially north and south.

$$\begin{array}{cc} -\beta\psi_x + \nu\psi_{xxxx} + \tau = 0 & -1 < x < 1 \\ \psi = \psi_x = 0 & x = \pm 1 \end{array} \tag{1}$$

The parameter ν is positive and assumed to be small, $\nu \ll 1$. The constant β is $O(1)$. τ is also O(1) and for simplicity is assumed to be constant. The problem is linear with constant coefficients and so can be solved explicitly. The solution is

$$\begin{aligned} \psi = \frac{4\tau}{\sqrt{3}\beta} e^{-\frac{1}{2}\lambda(x+1)} \{\cos(\frac{\sqrt{3}}{2}\lambda(x+1) - \frac{\pi}{6}) - \frac{\sqrt{3}}{2\lambda}\cos(\frac{\sqrt{3}}{2}\lambda(x+1) - \frac{\pi}{3})\} \\ - \frac{\tau}{\beta\lambda} e^{-\lambda(1-x)} + \frac{\tau}{\beta}x - \frac{\tau}{\beta}(1 - \frac{1}{\lambda}) + e.s.t. \end{aligned} \tag{2}$$

The notation *e.s.t.* stands for terms which are exponentially small, that is of the form λ to some power times $e^{-\lambda}$. λ is the principal root of the characteristic equation $\lambda^3 - \beta/\nu = 0$.

$$\lambda = \left(\frac{\beta}{\nu}\right)^{\frac{1}{3}}$$

The ***vorticity*** $\varsigma = \psi_{xx}$ is given by

$$\begin{aligned}\varsigma = \frac{4\tau}{\sqrt{3}\beta}\lambda^2 e^{-\frac{1}{2}\lambda(x+1)}\{\cos(\frac{\sqrt{3}}{2}\lambda(x+1) - \frac{5\pi}{6}) \\ + \frac{\sqrt{3}}{2\lambda}\cos(\frac{\sqrt{3}}{2}\lambda(x+1))\} - \frac{\tau}{\beta}\lambda e^{-\lambda(1-x)} + e.s.t.\end{aligned} \tag{3}$$

λ is large being $O(\nu^{-\frac{1}{3}})$. From the above expression for the vorticity (3) it can be seen that there are boundary layers at both $+1$ and -1. The boundary layer at $+1$ has strength (i.e. coefficient o the exponential) of $\lambda = O(\nu^{-\frac{1}{3}})$ with exponential decay rate of λ. The boundary layer at -1 is much stronger having strength $\lambda^2 = O(\nu^{-\frac{2}{3}})$. It oscillates rapidly and decays with rate $\lambda/2$. The solution away from boundaries looks like $\psi \approx (\tau/\beta)x + constant$. This is a solution to $\beta\psi_x + \tau = 0$ and is known as the Sverdrup relation. The form of the boundary layer given here will be used to determine the stretching of the grid. The expression for the vorticity (3) will be compared to some numerical results in the final chapter.

Coordinate Stretching

From the form of the boundary layer one can decide how to stretch the computational grid. The most important results are the exponential character of the layer and its decay factor. Consider the problem of stretching the annular shaped component mesh shown in figure (1.1) in order to resolve a boundary layer. Recall that the mesh is first mapped onto the unit square. Assume that the coordinate r of the unit square corresponds to the radial direction on the annular strip with $r = 0$ being the outer boundary and $r = 1$ the inner boundary. The grids are stretched

using a transformation of the form

$$\tilde{r} = \alpha r + (1-\alpha)\frac{(1-e^{-dr})}{(1-e^{-d})}$$
$$= g(r)$$

Note that $g : [0,1] \rightarrow [0,1]$ and $g(0) = 0, g(1) = 1$. If d is large and positive then away from $r = 0$ $\tilde{r}$ is related r by $\tilde{r} \approx \alpha r + c$. Thus if the grid spacing in r in this region is uniform the grid spacing in $\tilde{r}$ will also be uniform. Near $r = 0$

$$\tilde{r} \approx c_1 + c_2 e^{-dr}$$

or

$$r \approx -\frac{1}{d} ln(\frac{1}{c_2}(\tilde{r} - c_1))$$

Thus if $\varsigma(r) \approx e^{-\beta r}$ for $r \approx 0$ then

$$\varsigma(\tilde{r}) \approx (\frac{1}{c_2}(\tilde{r} - c_1))^{\beta/d}$$

Given β and taking into account the transformation to (r, s) coordinates, the constant d can be chosen so that an exponential boundary layer in r will behave as a low order polynomial in $\tilde{r}$.

Hence the finite difference equations applied in the $\tilde{r}$ coordinates need only be accurate for low order polynomials.

3.3 Time Marching

Two types of time marching methods will be described in this section. One will be an explicit scheme which uses leap frog on equation (1.1a) to march the vorticity and then solves the Poisson equation (1.1b) to obtain the stream function. This procedure must be done in a way so that both boundary conditions are satisfied. The second method will be partially implicit, using a Crank-Nicolson implicit scheme on the boundary grid but an explicit scheme in the interior.

The leap frog time marching scheme illustrated for the model problem of the section 2.2 is easily generalized to give an explicit method for the full vorticity equation.

$$\varsigma_t = R_0(\psi_y \varsigma_x - \psi_x \varsigma_y) + \beta\psi_x + E_k \nabla^2 \varsigma + f$$

To simplify the notation, the coordinate transformation from (x, y) to (r, s) will be not be taken into account. In addition the subscript which usually denotes the component mesh will be suppressed. The subscript will instead denote the point (i, j) on the mesh. Application of leap frog scheme then gives

$$\begin{aligned}\frac{\varsigma_{ij}^{n+1} - \varsigma_{ij}^{n-1}}{2\Delta t} =& R_0(D_{0y}\psi_{ij}^n D_{0x}\varsigma_{ij}^n - D_{0x}\psi_{ij}^n D_{0y}\varsigma_{ij}^n) \\ &+ \beta D_{0x}\psi_{ij}^n + E_k D^2 \varsigma_{ij}^{n-1} + f_{ij}^n\end{aligned}$$

where D^2 is the central difference approximation to the Laplacian operator.

$$D^2 = D_{+x}D_{-x} + D_{+y}D_{-y}$$

$$D_{+x}u_{ij} = \frac{u_{i+1j} - u_{ij}}{h_x}$$

$$D_{-x}u_{ij} = \frac{u_{ij} - u_{i-1j}}{h_x}$$

For stability reasons the diffusion term in not evaluated at the time level n but rather at the time level $n-1$. Away from boundaries the flow is mainly hyperbolic in nature, assuming that the Ekman number E_k is small and there are no internal layers. Leap frog is then a good procedure, with time differencing errors $O(\Delta t^2 + E_k \Delta t)$. The ***explicit*** time marching scheme consists of the three steps. Advance the vorticity for all nonboundary points, determine the stream function from the Poisson equation with $\psi_{ij}^{n+1} = 0$ on the boundary and finally compute the vorticity on the boundary using the zero normal derivative boundary condition.

$$\begin{cases} \text{Determine } \varsigma_{ij}^{n+1} & \text{for } (i,j) \text{ in the interior} \\ \text{Compute } \psi_{ij}^{n+1} \text{ from } D^2\psi_{ij}^{n+1} = \varsigma_{ij}^{n+1} & \text{for } (i,j) \text{ in the interior} \\ \text{Determine } \varsigma_{ij}^{n+1} & \text{for } (i,j) \text{ on the boundary} \end{cases} \tag{1}$$

The vorticity on the boundary is obtained by using

$$\zeta_{ij}^{n+1} = D^2 \psi_{ij}^{n+1} \tag{2}$$

where the stream function on the line outside the boundary is obtained from a discrete approximation to the normal derivative condition. This approximation is of the form

$$D_b \psi_{i0}^{n+1} = c_{-1}\psi_{i,-1}^{n+1} + c_0 \psi_{i0}^{n+1} + c_1 \psi_{i1}^{n+1} + c_2 \psi_{i2}^{n+1} + \ldots = 0 \tag{3}$$

The boundary is assumed to be the line $j = 0$ in the above formula, with interior values $j = 1, 2, \ldots$. The coefficients in (3) are taken so the formula is third order accurate. For a discussion of boundary conditions see the second section in chapter 5. The explicit scheme described here is second order accurate and numerically is found to be stable.

For the curvilinear grid which lies next to the boundary there are advantages in using an implicit method. (It is also of interest to see if there are any troubles in using different time marching methods on different component meshes.) To resolve viscous boundary layers the mesh may be highly stretched in the normal direction next to the boundary $\partial\Omega$. In these stretched coordinates the coefficient of $\partial^2/\partial n^2$ in the Laplacian will be $O(1)$ and the flow will be parabolic in nature. Implicit methods are often a more efficient way to solve parabolic problems. The stretching of the grid close to the boundary leads to a very small grid spacing and so explicit time marching techniques such as leap frog will require a very small time step Δt .The stability condition is of the form

$$\Delta t \leq constant \frac{h^2}{E_k}$$

To overcome this restriction arising from the diffusive term one can use an implicit scheme. With such a scheme the time step can be primarily chosen by accuracy considerations. There are some difficulties in implementing an implicit scheme. These

problems arise from the manner in which the vorticity stream function equations are being solved. The vorticity equation is used to march the vorticity, but there are no boundary conditions on the vorticity. The implicit scheme requires the vorticity on the boundary at the next time level. One way to solve this problem is to iterate on the value of the vorticity on the boundary attempting to converge to that value for which both boundary conditions on the stream function are satisfied.

An implicit Crank Nicolson (or Peaceman Rachford) type difference approximation to the vorticity equation is

$$\begin{aligned}\frac{\zeta_{ij}^{n+1}-\zeta_{ij}^{n}}{\Delta t} =& R_0[D_{0y}(\frac{\psi_{ij}^{n+1}+\psi_{ij}^{n}}{2})D_{0x}(\frac{\zeta_{ij}^{n+1}+\zeta_{ij}^{n}}{2}) \\ & - D_{0x}(\frac{\psi_{ij}^{n+1}+\psi_{ij}^{n}}{2})D_{0y}(\frac{\zeta_{ij}^{n+1}+\zeta_{ij}^{n}}{2})] \\ & + \beta D_{0x}(\frac{\psi_{ij}^{n+1}+\psi_{ij}^{n}}{2}) + E_k D^2(\frac{\zeta_{ij}^{n+1}+\zeta_{ij}^{n}}{2}) + \quad f_{ij}^{n+\frac{1}{2}}\end{aligned} \tag{4}$$

with boundary conditions

$$\psi_{i0} = 0$$

$$D_b\psi_{i0} = 0$$

The y direction has again been taken as the direction normal to the boundary with the line $j = 0$ being the boundary. The choice of the discrete approximation D_b to $\partial\psi/\partial n = 0$ is of the form (3). The implicit scheme is only applied on the mesh which follows the boundary, M_2. The solution on the inner mesh M_1 is first advanced using leap frog. The values of the vorticity at the next time level on the interpolation boundary of M_2 are then known. This assumes that the interpolation equations are not coupled. The system of equations (4) is solved by iteration, the values of the vorticity on the line $j = 0$ being unknown. Let w_{ij}^k and p_{ij}^k denote approximations to ζ_{ij}^{n+1} and ψ_{ij}^{n+1} respectively. The k+1 'st iterate is obtained from the k'th iterate by solving a sequence of tridiagonal systems given by the following

equations.

$$\frac{w_{ij}^{k+1}-\zeta_{ij}^{n}}{\Delta t} = R_0[D_{0y}(\frac{p_{ij}^{k}+\psi_{ij}^{n}}{2})D_{0x}(\frac{w_{ij}^{k}+\zeta_{ij}^{n}}{2}) - D_{0x}(\frac{p_{ij}^{k}+\psi_{ij}^{n}}{2})D_{0y}(\frac{w_{ij}^{k+1}+\zeta_{ij}^{n}}{2})] + \beta D_{0x}(\frac{p_{ij}^{k}+\psi_{ij}^{n}}{2}) + E_k[\frac{w_{ij+1}^{k+1}-2w_{ij}^{k+1}+w_{ij-1}^{k+1}}{2h_y^2} + \frac{w_{i+1j}^{k}-2w_{ij}^{k+1}+w_{i-1j}^{k}}{2h_x^2} + D^2\zeta_{ij}^{n}] + f_{ij}^{n+\frac{1}{2}}$$

We must re-solve for the stream function (on all meshes) from the new values for the vorticity.

$$D^2 p_{ij}^{k+1} = w_{ij}^{k+1}$$

The new guess for the vorticity on the boundary is obtained by the method given in Israeli [1970]. This method changes the current guess of vorticity on the boundary by a constant times the current estimate for $\partial\psi/\partial n$.

$$w_{ij}^{k+1} = w_{ij}^{k} + K D_b p_{ij}^{k+1}$$

The constant K is determined from a one dimensional analysis of the equations near the boundary. K depends on the approximation to D_b.

Chapter 4

Poisson Solver

In this chapter we consider the numerical solution of Poisson's equation with Dirichlet boundary conditions.

$$\begin{aligned} \nabla^2 u &= f \qquad \text{in } \Omega \\ u &= u_0 \qquad \text{on } \partial\Omega \end{aligned} \tag{1}$$

The problem is discretized to be solved on the composite overlapping grids. The composite mesh function **v** will be the numerical approximation to the solution u. The right hand side f will have corresponding mesh function **f**. The Poisson equation is discretized on each component mesh.

$$D_p^2 \underline{v}_p = \underline{f}_p \qquad p = 1, 2 \tag{2}$$

Points on the interpolation boundaries are obtained by interpolation from the mesh which they overlap.

$$\underline{v}_1(i,j) = \sum_{k,l} \alpha_1(i,j,k,l)\underline{v}_2(k,l) \qquad (i,j) \in \iota N_1 \tag{3}$$

$$\underline{v}_2(i,j) = \sum_{k,l} \alpha_2(i,j,k,l)\underline{v}_1(k,l) \qquad (i,j) \in \iota N_2 \tag{4}$$

This system of equations (2),(3) and (4) can be written as a single composite mesh equation for the composite mesh function **v**.

$$\mathbf{Av} = \mathbf{f} \tag{5}$$

These equations will be called the *mesh equations*.

4.1 Direct Solution of the Mesh Equations

When computer storage is available the mesh equations (4.5) can be solved directly. The Yale sparse matrix routines, Eisenstat et al. [1977], have been used to solve this system. These programs initially reorder the equations so as to reduce the fill-in that occurs during the Gaussian elimination. The matrix is factored once and the equations can be solved for various right hand sides by using this factorization. When the time to perform the reordering and factorization is ignored and the system is small enough to fit into the main computer memory then this is probably the fastest way to solve the Poisson equation of all the techniques that were considered.

4.1.1 Interpolation Equations

The interpolation equations will be said to *decouple* if the right hand sides of the interpolation equations do not involve any points on the interpolation boundaries. That is to say that boundary points are interpolated entirely from interior points. If the interpolation equations are coupled then a system of equations must be solved for the interpolation boundary values. This system becomes singular as the overlap of the grids goes to zero. In that limit the interpolation equations become independent of the interior values. To get an idea of how this coupling affects the solution one can study a one dimensional overlapping mesh problem. This problem is studied in detail in chapter 5.

4.2 Iterative Solution of the Mesh equations

For very large systems it may be necessary to use an iterative method to solve the mesh equations (4.5). It is important to consider solving all the the equations simultaneously. Trying to solve one part of the grid and then the other in the manner of the Schwartz alternating procedure tends to result in much slower convergence rates. In the context of two overlapping meshes the Schwartz alternating procedure takes the form

(1) Fix the boundary values of M_1 and solve the grid equations for M_1 exactly.

(2) Update the boundary values of M_2. (equation (4.4))

(3) Fix the boundary values of M_2 and solve the grid equations for M_2 exactly.

(4) Update the boundary values of M_1. (equation (4.3))

This process is repeated, and under certain conditions the iteration will converge. The procedure is probably more useful for theoretical than practical purposes. As already mentioned, Starius was able to show the convergence of the Schwartz alternating procedure, applied as an iteration, to the solution of a general class of elliptic problems on composite meshes, Starius [1977]. There seems little reason, however, to devote a lot of effort to accurately solve the equations on one component mesh when the boundary values are obtained from inaccurate solutions on the other component meshes. Studies by J. Linden reported in Hackbusch and Trottenburg [1982] indicate that the Schwartz procedure is much slower and also sensitive to the amount of overlap between the grids. A better approach is to update all grid points in each iteration sweep. Let $\mathbf{v}(n)$ denote the n'th iterate. One way to obtain the $n+1^{st}$ iterate is as follows.

$$L_p \underline{v}_p(n+1) = -U_p \underline{v}_p(n) + \underline{f}_p \qquad p = 1 \qquad (1)$$

$$\underline{v}_2(i,j,n+1) = \sum_{k,l} \alpha_2(i,j,k,l)\underline{v}_1(k,l,n) \qquad (i,j) \in \iota N_2 \qquad (2)$$

$$L_p \underline{v}_p(n+1) = -U_p \underline{v}_p(n) + \underline{f}_p \qquad p = 2 \qquad (3)$$

$$\underline{v}_1(i,j,n+1) = \sum_{k,l} \alpha_1(i,j,k,l)\underline{v}_2(k,l,n) \qquad (i,j) \in \iota N_1 \qquad (4)$$

Equations for one iteration sweep

The component mesh operators L_p and U_p are determined by the particular iteration technique that is used. For example one might want to use Gauss Seidel with over-relaxation (SOR) or perhaps line SOR. Note that the values on the

boundaries are updated at every sweep. In addition the most recently calculated boundary values should be used in equation (3) for Gauss Seidel type iterations. If the interpolation equations are explicit then all values appearing in the right hand sides of the interpolation equations can be taken from the iterate $n + 1$. It is a good idea to keep the convergence rates on both grids about the same or else one will encounter the same problems that are associated with the Schwartz alternating procedure. To keep the errors on the two grids nearly the same it may be necessary to make more than one relaxation sweep on the one grid for each sweep on the other grid.

The steps (1)-(4) define the order in which the grid point values are updated during the iteration sweep. In this sense the two mesh iteration is no different from a one mesh iteration since even on a single mesh the points are updated in a particular order. This iteration can be denoted symbolically by the composite mesh operator $\mathbf{S} : \mathbf{Mf} \rightarrow \mathbf{Mf}$. This operator consists of four component operators. $\mathbf{S} = I_1 \cdot S_2 \cdot I_2 \cdot S_1$. (To be read from right to left). S_1 indicates step (1), sweeping over the points on the mesh M_1. I_2 is the interpolation operator corresponding to the interpolation equations (2), assigning the interpolation boundary points of M_1. S_2 and I_2 are defined similarly. $\mathbf{S}$ will be called the *composite iteration operator*, or in the case of multigrid it will be the *composite smoothing operator.*

$$\mathbf{S} = \begin{cases} S_1 & : \text{Sweep } M_1 \\ I_2 & : \text{Interpolate } \iota M_2 \\ S_2 & : \text{Sweep } M_2 \\ I_1 & : \text{Interpolate } \iota M_1 \end{cases}$$

In this composite iteration the interpolation boundary points are updated immediately after the sweep over the mesh from which they interpolate. Note that in the sweep over the points of M_2 the newest values on the boundaries can be used. Another possible iteration operator is $\tilde{\mathbf{S}} = I_2 \cdot I_1 \cdot S_2 \cdot S_1$. This is the iteration

$$\tilde{\mathbf{S}} = \begin{cases} S_1 & : \text{Sweep } M_1 \\ S_2 & : \text{Sweep } M_2 \\ I_1 & : \text{Interpolate } \iota M_1 \\ I_2 & : \text{Interpolate } \iota M_2 \end{cases}$$

Experience seems to indicate that $\tilde{\mathbf{S}}$ is usually an inferior composite iteration to $\mathbf{S}$. Another question of interest is whether one should try to accelerate the convergence of the interpolation equations (2) and (4) by introducing a parameter. One possibility is to introduce a parameter as in successive over-relaxation.

$$\underline{v}_1{}^{(n+1)} = (1-\omega)\underline{v}_1{}^{(n)} + \omega\Big(\sum_{k,l} \alpha_1(i,j,k,l)\underline{v}_2{}^{(n)}(k,l)\Big) \tag{5}$$

Some improvement in convergence rates can be obtained by using this procedure.

4.3 Multigrid Solution of the Mesh equations

The composite mesh equations (4.5) can be solved using the multigrid algorithm. Although it is assumed that the reader has some familiarity with the multigrid procedure a short description will now be given. For further details see, for example, Hackbusch and Trottenberg [1982] or Brandt [1977]. Multigrid is an iterative method which utilizes a sequence of coarser and coarser grids to accelerate the convergence of the solution on the finest grid. The basic principle rests on the fact that it is possible to obtain iterative procedures (*smoothers*) for which the high frequency components of the solution converge rapidly. This means that after a few smoothing iterations the part of the solution yet to converge is smooth and hence can be accurately solved for on a coarser grid. Many of the standard iterative procedures such as Gauss-Seidel or SOR possess the property that the number of iterations required to solve a system of N equations to a given accuracy is proportional to N or worse. Multigrid convergence rates, however, are independent of N and hence their attractiveness for solving large problems.

4.3.1 Notation

Now consider the application of the multigrid algorithm to the solution of the composite mesh equations. The notation introduced for composite meshes will have to be extended since multigrid utilizes a sequence of composite meshes. A superscript q, $q = 1, 2, 3, \ldots$ will be used to denote a particular composite mesh. This superscript will appear on the component meshes, composite mesh functions etc. Hence a sequence of composite meshes $\mathbf{M}^q \quad q = 1, 2, 3, \ldots$ will be considered. Each composite mesh is made up of a union of component meshes.

$$\mathbf{M}^q = \bigcup_p M_p^q$$

Composite mesh functions will be written as $\mathbf{f}^q$. They are a union of component mesh functions $\underline{f}_p^q$. The composite mesh operator is generalized to be a mapping of mesh functions defined on one composite mesh to a mesh function defined on a possibly different composite mesh.

$$\mathbf{S}^{q_1 \to q_2} : \mathbf{M}\mathbf{f}^{q_1} \to \mathbf{M}\mathbf{f}^{q_2}$$

If $q_1 = q_2 = q$ the operator will usually be written in the simpler form $\mathbf{S}^q$.

4.3.2 Multigrid Algorithm on Composite Meshes

Suppose the region Ω has been covered by a composite overlapping mesh. Denote this composite mesh by $\mathbf{M}^1$. Associated with $\mathbf{M}^1$ will be a system of mesh equations of the form (4.5).

$$\mathbf{A}^1\mathbf{v}^1 = \mathbf{f}^1$$

These will be the *fine grid* equations.

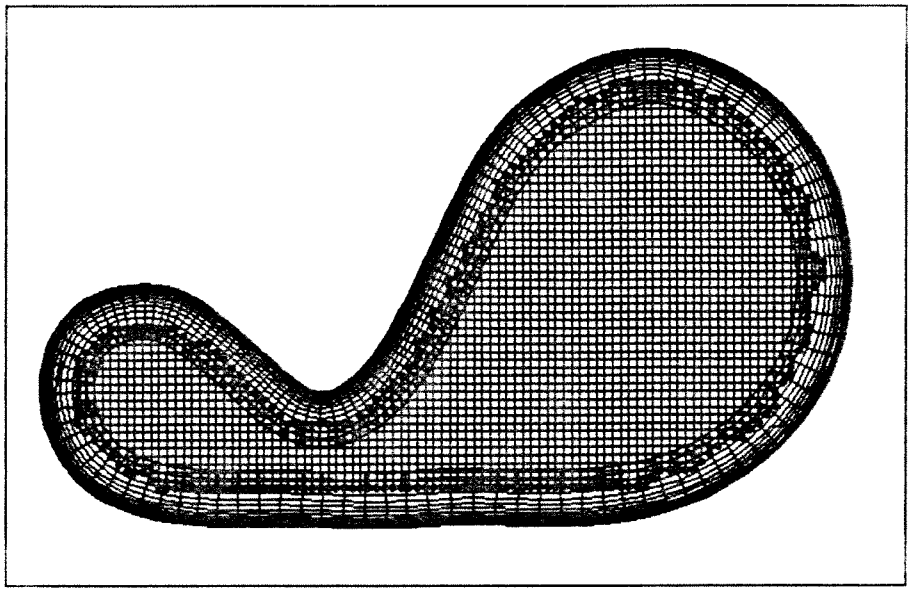


Figure 4.1 Fine grid $\mathbf{M}^1$

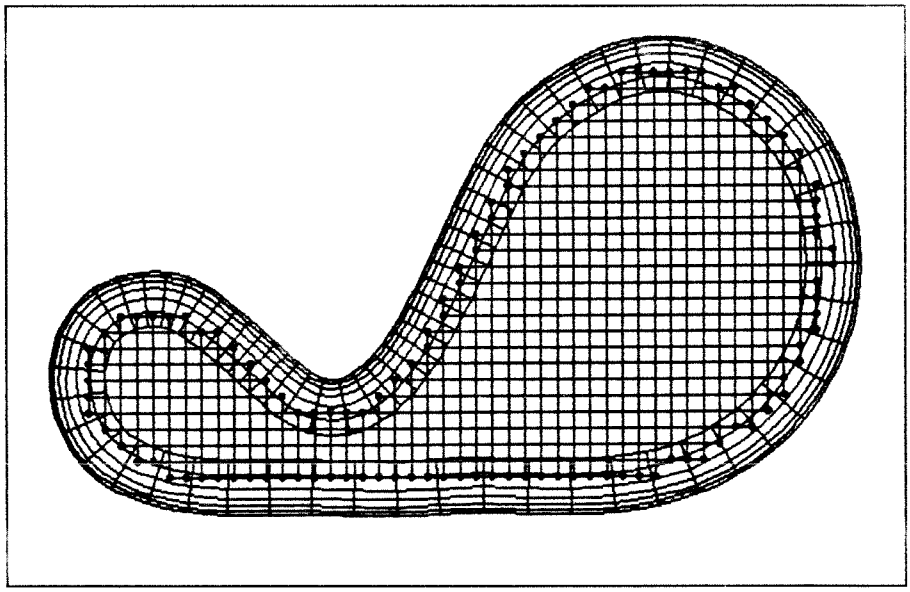


Figure 4.2 Coarse grid $\mathbf{M}^2$

A coarser grid $\mathbf{M}^2$ is constructed from $\mathbf{M}^1$ by removing every second line in the two coordinate directions from each of the component meshes of the fine grid. $\mathbf{M}^1$ can be constructed so that this coarser grid is a valid overlapping grid. The grid equations for $\mathbf{M}^2$ will again be of the form (4.2)-(4.4) and will be denoted by

$$\mathbf{A}^2\mathbf{v}^2 = \mathbf{f}^2 \tag{2}$$

This process of generating coarser grids and coarser grid equations can be repeated. In practice the coarsest grid is generated first and lines are added to form the finer grids. If the most coarse grid has sufficient overlap between its component composite meshes then the finer grids will also have enough overlap. In the previous section the method to perform relaxation sweeps was indicated. Multigrid smoothing steps will be performed in a similar manner since a smoother is just an iteration technique which is efficient at reducing the high frequency error components. The algorithm will first be outlined. The objective is to solve the fine grid equations . Let $\mathbf{v}^1(k)$ denote the k'th iterate in this iteration. Then the multigrid algorithm can be written as follows.

Multigrid Algorithm

while not converged **do**

smooth ν_1 times

$$\mathbf{v}^1(*) \leftarrow (\mathbf{S})^{\nu_1}\mathbf{v}^1(k)$$

compute the defect and transfer to coarser grid

$$\mathbf{f}^2 \leftarrow \mathbf{R}^{1\rightarrow 2}(\mathbf{f}^1 - \mathbf{A}^1\mathbf{v}^1(*))$$

solve the defect equation on the coarser grid

$$\mathbf{v}^2 \leftarrow (\mathbf{A}^2)^{-1}\mathbf{f}^2$$

correct the fine grid solution from coarse grid solution

$$\tilde{\mathbf{v}^1} \leftarrow \mathbf{v}^1(*) + \mathbf{P}^{2\rightarrow 1}\mathbf{v}^2$$

smooth ν_2 times

$$\mathbf{v}^1(k+1) \leftarrow (\mathbf{S})^{\nu_2}\tilde{\mathbf{v}^1}$$

end while

This algorithm is essentially the standard one for linear problems, with the difference that the various operations indicated should be applied to functions de-

fined on the composite overlapping grids. **S** stands for the composite smoothing operator. This composite smooth involves the four steps as outlined in section 4.2. The defect in a given composite mesh function $\mathbf{v}^1$ is defined to be $\mathbf{d}^1 = \mathbf{f}^1 - \mathbf{A}^1\mathbf{v}^1$. This is the amount by which the equations are not satisfied. $\mathbf{R}^{1\rightarrow 2}$ is the restriction operator which maps the defect on the fine mesh $\mathbf{M}^1$ to the forcing function for the mesh equations of the coarse mesh $\mathbf{M}^2$. $\mathbf{P}^{2\rightarrow 1}$ is the prolongation operator which maps the correction computed on the coarse grid on to the fine grid. These mesh operators are now discussed in more detail.

4.3.3 Composite Smoothers, Restrictions and Prolongations

Smoothers

As mentioned earlier a smoother is just an iterative method which is efficient at reducing the high frequency error components. Methods which are optimized to give a good overall convergence rate are usually not the best smoothers.

For example, optimal SOR is not as good a smoother as Gauss Seidel. The relaxation parameter, ω in SOR is chosen to minimize the spectral radius of the iteration matrix. A good smoother on the other hand only requires that a certain fraction of the higher frequencies converge rapidly. A good discussion of smoothers and their smoothing properties can be found in Stuben and Trottenburg [1981]. Here we simply outline a few of the smoothers that have been used.

Jacobi ω Relaxation

This is a Jacobi iteration with a relaxation parameter included in the fashion of Gauss-Seidel. As a pure iteration procedure the convergence rate of Jacobi cannot be improved by adding such a parameter. However, smoothing properties can be improved and the parameter ω is chosen less than one, ***under relaxation***, with $\omega = 4/5$ a good value.

Red Black Jacobi

This is a Jacobi type iteration in which the points on a square grid are updated in the so called red-black ordering. The points on the grid are divided into two groups in the manner of a checkerboard. A Jacobi sweep of all red points is performed followed by a Jacobi sweep of all black points. On the sweep over the black points the new red point values are used.

Line Zebra

This is a line Gauss-Seidel method in which all even numbered lines in a given direction are updated before all odd numbered lines.

Restriction Operators (Fine to Coarse Grid Transfer)

$\mathbf{R}^{1\to 2}$ is the projection or *restriction* operator which maps mesh functions defined on the mesh $\mathbf{M}^1$ to mesh functions defined on $\mathbf{M}^2$. One typical choice for this restriction operator in the interior of a composite mesh is the so called *full weighting*, which can be written symbolically as

$$\mathbf{R}^{1\to 2} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

This notation is to indicate that a point value on the coarse grid $\underline{v}_p^2(k,l)$ is an average of the 9 surrounding points of the fine grid

$$\begin{aligned}\underline{v}_p^2(k,l) = \frac{1}{16}(&\underline{v}_p^1(i-1,j+1) + 2\underline{v}_p^1(i,j+1) + \underline{v}_p^1(i+1,j+1) \\ &+2\underline{v}_p^1(i-1,j) + 4\underline{v}_p^1(i,j) + 2\underline{v}_p^1(i+1,j) \\ &1\underline{v}_p^1(i-1,j-1) + 2\underline{v}_p^1(i,j-1) + 1\underline{v}_p^1(i+1,j-1))\end{aligned}$$

Along the interpolation boundary a one dimensional restriction operator is used.

$$\iota\mathbf{R}^{1\to 2} = \frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

That is to say the value of a point on the interpolation boundary of the coarse grid is taken as an average with weights 1/4 , 1/2 and 1/4 of the three corresponding

points on the interpolation boundary of the finer grid. The composite restriction operator $\mathbf{R}^{1\to 2}$ then can be written as $\mathbf{R}^{1\to 2} = \iota R_2^{1\to 2} \cdot R_2^{1\to 2} \cdot \iota R_1^{1\to 2} \cdot R_1^{1\to 2}$.

$$\mathbf{R}^{1\to 2} = \begin{cases} R_1^{1\to 2} & : \text{Restrict } M_1^1 \text{ to } M_1^2 \\ \iota R_1^{1\to 2} & : \text{Restrict the boundary } \iota M_1^1 \text{ to } \iota M_1^2 \\ R_2^{1\to 2} & : \text{Restrict } M_2^1 \text{ to } M_2^2 \\ \iota R_2^{1\to 2} & : \text{Restrict the boundary } \iota M_2^1 \text{ to } \iota M_2^2 \end{cases}$$

Prolongation Operators (Coarse to Fine Grid Transfer)

$\mathbf{P}^{2\to 1}$ is the *prolongation operator* mapping grid functions on $\mathbf{M}^2$ to grid functions on $\mathbf{M}^1$. One form for this interpolation operator as applied to interior grid points is

$$\mathbf{P}^{2\to 1} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

This operator notation is to indicate how a value on the coarse grid is distributed to the 9 surrounding points on the fine grid. The interpolation can take one of four forms depending on whether the fine grid point coincides exactly with a coarse grid point, or lies between two coarse grid points (in either coordinate direction) or lies at the centre of 4 coarse grid points. Thus a point on the fine grid $\underline{v}_p^1(i,j)$ will be given by one of the following four equations.

$$\begin{aligned} \underline{v}_p^1(i,j) &= \underline{v}_p^2(k,l) \\ \underline{v}_p^1(i,j) &= \frac{1}{2}\underline{v}_p^2(k,l) + \frac{1}{2}\underline{v}_p^2(k+1,l) \\ \underline{v}_p^1(i,j) &= \frac{1}{2}\underline{v}_p^2(k,l) + \frac{1}{2}\underline{v}_p^2(k,l+1) \\ \underline{v}_p^1(i,j) &= \frac{1}{4}(\underline{v}_p^2(k,l) + \underline{v}_p^2(k+1,l) + \underline{v}_p^2(k,l+1) + \underline{v}_p^2(k+1,l+1)) \end{aligned}$$

The coarse grid values on the interpolation boundaries are transferred using a one dimensional prolongation operator.

$$\iota\mathbf{P}^{2\to 1} = \frac{1}{2}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

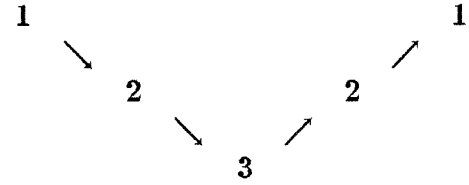
Hence the composite prolongation operator can take the form $\mathbf{P}^{2\to 1} = \iota P_2^{2\to 1} \cdot P_2^{2\to 1} \cdot \iota P_1^{2\to 1} \cdot P_1^{2\to 1}$.

$$\mathbf{P}^{2\to 1} = \begin{cases} P_1^{2\to 1} & : \text{prolongate } M_1^2 \text{ to } M_1^1 \\ \iota P_1^{2\to 1} & : \text{prolongate } \iota M_1^2 \text{ to } \iota M_1^1 \\ P_2^{2\to 1} & : \text{prolongate } M_2^2 \text{ to } M_2^1 \\ \iota P_2^{2\to 1} & : \text{prolongate } \iota M_2^2 \text{ to } \iota M_2^1 \end{cases}$$

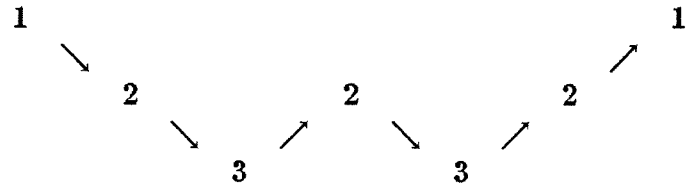
The restricted defect equation need not be solved exactly on the coarser grid, an approximate solution can be used. This approximate solution can of course be obtained by the multigrid procedure. The algorithm then becomes recursive, although at some level the defect equations are solved exactly or approximately by another method. The number of equations to solve in the defect equation at this last level is usually so small compared to the number of fine grid equations that it is not too crucial how efficiently the equations are solved. Note that the defect involves not only the residuals of the equations approximating the Laplacian but also the residuals of the interpolation equations. Depending on the composite smooth that was used, such as $\mathbf{S}$, the residuals in some of the interpolation equations can be nonzero. This means that in the solution of the defect equations the corresponding interpolation equations will be inhomogeneous.

4.3.4 Choosing the Cycle and Parameters

An important part of the multigrid algorithm that has not yet been touched upon is the choice of the various parameters ν_1, ν_2, etc. and the choice of cycle. *Cycle* is the term to denote the sequence in which the different levels of grids are traversed. The standard V cycle for example consists of moving from finest down to the coarsest and then back up to the finest. This is represented schematically for 3 levels as



The finest grid is level 1 and the coarsest grid level 3. An alternative is the W cycle



Brandt [1977] outlines the principles for dynamically determining the type of cycle and the values for ν_1 and ν_2. This procedure requires that one keep track of the residuals. In this context the residual refers to some norm of the defect.

$$r^q = ||\mathbf{f}^q - \mathbf{A}^q\mathbf{v}^q||$$

The smoothing rate (based on residuals) is the ratio of the residual after smoothing to the residual before smoothing. The basic approach to the dynamic determination of parameters is

(1) Continue to smooth at a given level as long as the smoothing rate is less than some number η, $\eta \approx .5$. When the smoothing rate becomes worse than η move to a coarser grid.

(2) Move to a finer grid once the residual at this level has been reduced by a total factor of δ, $\delta \approx .2$ Otherwise perform another multigrid iteration at this level.

Additional parameters are introduced since multigrid is being performed on composite meshes. Within the composite smooth one has decide how many interations to perform in each of the component smoothers. This choice can be made by trying to keep the *component residuals* (normalized appropriately) about the same size. It is important to do this or the overall convergence rates may be much worse. The technique of dynamically choosing parameters may not be the most efficient since there is some extra work required to compute residuals. However, it is an instructive approach to take when optimizing the parameters for a particular problem.

Chapter 5

Model Problem Analyses

5.1 One Dimensional Overlapping Grid

In order to gain some insight into the behaviour of the solution of the mesh equations we consider the solution of a one dimensional elliptic problem.

$$\begin{cases} u_{xx} = f & x \in (0,1) \\ u(0) = 0 \quad u(1) = 0 \end{cases}$$

This model problem can be solved on a composite overlapping grid. In this simple case one can determine the form of the errors. It is then easy to see how the accuracy of the discrete solution depends on the order of accuracy of the difference formulae, the interpolation formulae and the amount of overlap between the grids.

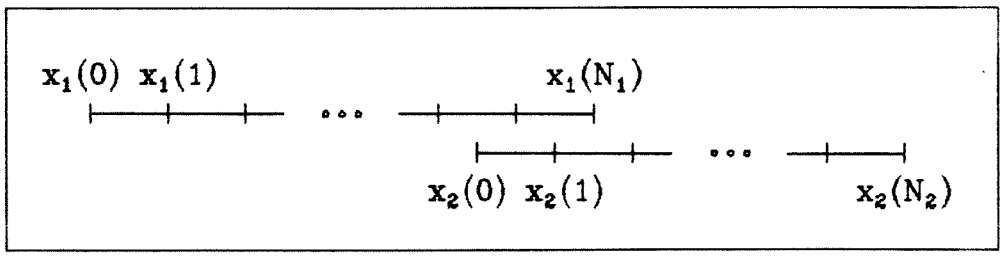


Figure 5.1 Composite Mesh for the Model 1D Problem

The composite mesh consists of two overlapping component meshes, M_1 and M_2

$$M_1 = \{\, x_1(i) \,|\, x_1(i) = x_1(0) + ih_1 \quad i = 0, 1, 2, \ldots, N_1 \,\}$$

$$M_2 = \{\, x_2(i) \,|\, x_2(i) = x_2(0) + ih_2 \quad i = 0, 1, 2, \ldots, N_2 \,\}$$

The meshes are assumed to overlap by a positive amount d.

$$d = x_1(N_1) - x_2(0) > 0$$

The mesh equations take the form

$$D_{+p}D_{-p}v_p(i) = f(x_p(i)) \qquad i = 1,2,\ldots,N_p - 1 \quad p = 1,2 \tag{1a}$$

$$v_1(0) = 0 \tag{1b}$$

$$v_2(N_2) = 0 \tag{1c}$$

$$v_1(N_1) = \sum_l \alpha_1(l)v_2(l) \qquad l \in [0, N_2] \tag{1d}$$

$$v_2(0) = \sum_k \alpha_2(k)v_1(k) \qquad k \in [0, N_1] \tag{1e}$$

where the difference operator $D_{+p}D_{-p}$ is defined as

$$D_{+p}D_{-p}v_p(i) = \frac{v_p(i+1) - 2v_p(i) + v_p(i-1)}{h_p^2}$$

Let $h = \max(h_1, h_2)$. An acceptable solution to the mesh equations (1) will be second order accurate since the difference approximation to u_{xx} is second order. The following proposition indicates the sufficient conditions under which the solution will be second order. For certain f or special arrangements of the composite meshes it is not necessary to satisfy these conditions. However, in general the conditions are necessary as well as being sufficient.

Proposition.

(1) If as the meshes are refined and $h \to 0$ the amount of overlap d is greater than some positive constant independent of h and the interpolation equations are at least second order accurate then the solution to the mesh equations (1) will be second order accurate. (The definition of order of accuracy follows shortly.)

(2) If as the meshes are refined and $h \to 0$ the amount of overlap satisfies $d \geq ch$ where c is a positive constant independent of h and the interpolation equations are at least third order accurate then the solution to the mesh equations (1) will be second order accurate.

As a remark it is to be noted that in practice one is almost always in the situation of case (2). The amount of overlap will be any where from a few mesh widths down to a fraction of a mesh width. To study the accuracy of the mesh equations we look at the error equations. Let **e** be the composite mesh error function. **e** is the difference between the exact solution u and the computed solution, normalized in a fashion shown shortly. This error function will have components e_1 and e_2 corresponding to the errors on M_1 and M_2. Each component is normalized by the square of the mesh spacing of the corresponding component mesh.

$$h_p^2 e_p(i) = u(x_p(i)) - v_p(i) \qquad p = 1, 2$$

The component errors satisfy the equations

$$\begin{aligned} & D_{+p}D_{-p}e_p(i) = \tau_p(i) \qquad i = 1, 2, \ldots, N_p - 1 \qquad p = 1, 2 \\ & e_1(0) = 0 \\ & e_2(N_2) = 0 \\ & \gamma e_1(N_1) = \sum \alpha_1(l) e_2(l) + \iota\tau_1 \\ & \frac{1}{\gamma} e_2(0) = \sum \alpha_2(k) e_1(k) + \iota\tau_2 \end{aligned} \tag{2}$$

where γ is the square of the ratio of the mesh widths.

$$\gamma = \left(\frac{h_1}{h_2}\right)^2$$

τ_p, $\iota\tau_1$ and $\iota\tau_2$ are the truncation errors in the finite difference formulae and interpolation formulae, respectively. The central difference operator $D_{+p}D_{-p}$ is a second order accurate approximation to d^2/dx^2. The interpolation formulae are assumed to be accurate to order μ and ν.

$$\begin{aligned} \tau_p &= \tau(x_p(i)) + O(h_p^2) \\ \tau(x) &\equiv \frac{2}{4!}\frac{d^4u}{dx^4}(x) \\ \iota\tau_1 &= h_2^{\mu-2} C_\mu \frac{d^\mu u}{dx^\mu}(x_1(N_1)) + O(h_2^{\mu-1}) \\ \iota\tau_2 &= h_1^{\nu-2} C_\nu \frac{d^\nu u}{dx^\nu}(x_2(0)) + O(h_1^{\nu-1}) \end{aligned}$$

There are two types of forcing terms in the error equations (2). The τ_p forcing in the first equations will turn out not to cause any difficulties. The forcing terms in the interpolation equations are more critical. To isolate the effects of these terms we split the problem into two parts. The error will be written as the sum $\mathbf{e} = \mathbf{e}^F + \mathbf{e}^I$. e_p^F satisfies the equations with no inhomogeneous terms in the interpolation equations

$$\begin{aligned} D_{+p}D_{-p}e_p^F(i) &= \tau_p(i) \qquad i = 1, 2, \ldots, N_p - 1 \quad p = 1, 2 \\ e_1^F(0) &= 0 \\ e_2^F(N_2) &= 0 \\ \gamma e_1^F(N_1) &= \sum \alpha_1(l) e_1^F(l) \\ \frac{1}{\gamma} e_2^F(0) &= \sum \alpha_2(k) e_1^F(k) \end{aligned} \tag{3}$$

and e_p^I is the solution to

$$\begin{aligned} D_{+p}D_{-p}e_p^I(i) &= 0 \qquad i = 1, 2, \ldots, N_p - 1 \quad p = 1, 2 \\ e_1^I(0) &= 0 \\ e_2^I(N_2) &= 0 \\ \gamma e_1^I(N_1) &= \sum \alpha_1(l) e_2^I(l) + \iota\tau_1 \\ \frac{1}{\gamma e_2^I(0)} &= \sum \alpha_2(k) e_1^I(k) + \iota\tau_2 \end{aligned} \tag{4}$$

The solution to the equations (4) for e_p^I will now be obtained. The first equation and the boundary conditions at 0 and 1 are satisfied by functions of the form

$$\begin{aligned} e_1^I(i) &= i e_1^I(1) \\ e_2^I(i) &= (N_2 - i) e_2^I(N_2 - 1) \end{aligned}$$

The values for $e_1^I(1)$ and $e_2^I(N_2 - 1)$ are determined by the interpolation equations of (4). This leads to the following system of equations

$$\begin{bmatrix} -\gamma N_1 & N_2 - d/h_2 \\ N_1 - d/h_1 & \frac{1}{\gamma} N_2 \end{bmatrix} \begin{bmatrix} e_1^I(1) \\ e_2^I(N_2 - 1) \end{bmatrix} = \begin{bmatrix} -\iota\tau_1 \\ \iota\tau_2 \end{bmatrix} \tag{5}$$

with solution

$$\begin{bmatrix} e_1^I(1) \\ e_2^I(N_2-1) \end{bmatrix} = \frac{h_1 h_2}{d} \begin{bmatrix} \frac{1}{\gamma} N_2 \iota\tau_1 + (N_2 - d/h_2)\iota\tau_2 \\ (N_1 - d/h_1)\iota\tau_1 + N_1 \gamma \iota\tau_2 \end{bmatrix}$$

In obtaining (5) we assumed that the interpolation equations were at least second order. The determinant of the matrix appearing in (5) is $-d/(h_1 h_2)$. It is apparent from this result that one may have difficulty if d is small. Expressions for the errors e_1^I and e_2^I are

$$\begin{aligned} h_1^2 e_1^I(i) &= h_1^2 (\frac{i}{N_1}) N_1 e_1^I(1) \\ &= (\frac{i}{N_1}) h_1^2 \frac{(N_1 h_1)(N_2 h_2)}{d\gamma} (\iota\tau_1 + \gamma(1 - \frac{d}{N_2 h_2})\iota\tau_2) \\ &= (\frac{i}{N_1}) \frac{\Delta_1 \Delta_2}{d} (h_1^2 (1 - \frac{d}{\Delta_2})\iota\tau_2 + h_2^2 \iota\tau_1) \end{aligned} \tag{6}$$

$$\begin{aligned} h_2^2 e_2^I(N_2) &= h_2^2 (\frac{N_2 - i}{N_2}) N_2 e_2^I(N_2 - 1) \\ &= (\frac{N_2 - i}{N_2}) \frac{\Delta_1 \Delta_2}{d} (h_1^2 \iota\tau_2 + h_2^2 (1 - \frac{d}{\Delta_1})\iota\tau_1) \end{aligned} \tag{7}$$

where Δ_1 and Δ_2 are the lengths of the component meshes M_1 and M_2 respectively.

$$\Delta_1 = N_1 h_1 \qquad \Delta_2 = N_2 h_2 \qquad \Delta_1 + \Delta_2 = 1 + d$$

Note that the product $\Delta_1 \Delta_2$ is always $\geq 1/4$.

Consider now equation (3) which determines e_p^F. It is not hard to show that

$$e_p^F(i) = T(x_p(i)) + O(h)$$

where T is the solution to

$$T_{xx} = \tau$$

$$T(0) = T(1) = 0.$$

Specifically

$$T(x) = \int_0^x \int_0^t \tau(t) dt dx - x \int_0^1 \int_0^t \tau(t) dt dx. \tag{8}$$

This result requires that the interpolation equations are second order.

We can now write down expressions for the components of the total error.

$$h_1{}^2 e_1(i) = \left(\frac{i}{N_1}\right)\frac{\Delta_1\Delta_2}{d}\left(h_1^2\left(1 - \frac{d}{\Delta_2}\right)\iota\tau_2 + h_2^2\iota\tau_1\right) + h_1{}^2 T(x_1(i)) + O(h^3) \tag{9}$$

$$h_2{}^2 e_2(i) = \left(\frac{N_2 - i}{N_2}\right)\frac{\Delta_1\Delta_2}{d}\left(h_1^2\iota\tau_2 + h_2^2\left(1 - \frac{d}{\Delta_1}\right)\iota\tau_1\right) + h_2{}^2 T(x_2(i)) + O(h^3) \tag{10}$$

From these last two equations we obtain sufficient conditions for second order accuracy. These conditions are

$$\iota\tau_1 = O(d) \text{ and } \iota\tau_2 = O(d) \qquad \text{as } h \to 0$$

Note that for special composite meshes the interpolation truncation errors $\iota\tau_1$ and $\iota\tau_2$ could be identically zero. For example, the interpolation points could happen to fall exactly on a mesh point of the opposite component mesh. Neglecting these cases the conclusion in general is that if the distance between the meshes goes to zero as h goes to zero then the interpolation equations must be more accurate. If, however, the amount of overlap d is greater than or equal to some constant as h goes to zero then the interpolation equations need only be second order accurate.

As a check of the above results the mesh equations (1) for the model problem were solved numerically for the following problem.

$$N_1 = 10 \qquad N_2 = 10$$

$$x_1(N_1) = .6 \qquad x_2(N_2) = .575$$

$$d = .025 = \frac{5}{12}h_1 = \frac{10}{17}h_2$$

$$f = -\pi^2 \sin(\pi x)$$

Order of Interpolation	$h_1^2\|\|e_1\|\|_\infty$		$h_2^2\|\|e_2\|\|_\infty$	
	Calculated	Estimated	Calculated	Estimated
2 (linear)	.592e-1	.586e-1	.608e-1	.616e-1
3 (quadratic)	.265e-2	.307e-2	.257e-2	.165e-2

Table 5.1 Error Comparison

The true solution is $u = \sin(\pi x)$. The maximum errors were computed for second and third order accurate interpolation. These results were compared to the estimates obtained using (9) and (10). The results, which are summarized in table 5.1, suggest the correctness of the formulae that have been derived for the errors.

5.2 Boundary Conditions for the Stream Function Vorticity Equations †

5.2.1 Introduction

One of the difficulties associated with the stream function vorticity equations is the numerical implementation of the no slip boundary conditions $\psi = 0$ and $\partial\psi/\partial n = 0$. These boundary conditions place constraints on the stream function and its normal derivative at the boundary. There is, however, no explicit boundary condition for the vorticity. This lack of a vorticity boundary condition seems to be at the root of the problems that appear. Some reviews of numerical methods for the stream function vorticity equations can be found in Peyret and Taylor [1983], Orszag and Israeli [1974] and Roache [1972].

There are (at least) two ways to look at the numerical approximation of the boundary conditions. The first and more common approach is to use the normal derivative condition to determine an expression for the vorticity on the wall in terms of interior and boundary values of the stream function. The alternative approach, which we prefer, is to think of approximating not the vorticity at the wall but rather approximating $\partial\psi/\partial n$. The vorticity on the wall is defined as it is in the interior. The discrete approximation to $\partial\psi/\partial n$ will determine the values of ψ_{ij} needed to apply the formula for ζ on the boundary.

To achieve accurate answers many investigators advocate the use of higher order approximations to the boundary conditions. When an implicit time marching method is used or when a steady state solution is required, the stream function vorticity equations may have to be solved by iteration. When such iterations are required the higher order boundary conditions are often found to be ***unstable*** and abandoned in preference to lower order schemes. We have found in implicit time stepping calculations that higher order methods are stable provided the appropriate iteration scheme is used, such as the one developed by Israeli [1970].

† The work in this section was performed together with Michael Naughton.

There has been some work performed at trying to obtain accurate and stable boundary conditions and to try and understand the difficulties present in this problem including the work of Briley [1971], Bontoux, Gilly and Roux [1980], Israeli [1970] and Orszag and Israeli [1974]. It appears that most results are heuristic or only qualitative in nature, although Orszag and Israeli study a model problem similar to the one we look at here.

Reduction to a Model Problem

Numerical experience shows that the difficulties seem to be related to boundary layers in the vorticity. In such boundary layers the flow is often one dimensional in character, varying in the normal direction to the boundary. Tangential derivatives of the stream function and vorticity are small. A reasonable model problem to study thus seems to be the following two point boundary value problem.

$$\varsigma_t = \nu \varsigma_{xx} + f(x,t)$$

$$\varsigma = \psi_{xx}$$

$$\psi = \psi_x = 0 \text{ at } x = 0,1$$

The terms that have been neglected are assumed to be small or to vary smoothly in which case one can argue that they can be absorbed into the forcing term f. Discretization of this system in time using a second order centred finite difference scheme gives

$$\frac{\varsigma^{n+1} - \varsigma^n}{\Delta t} = \nu \frac{\varsigma_{xx}^{n+1} + \varsigma_{xx}^n}{2} + f(x, t_n + \frac{\Delta t}{2})$$

$$\varsigma^n = \psi_{xx}^n$$

$$\psi^n = \psi_x^n = 0 \quad \text{at } x = 0,1$$

where $\psi^n(x)$ is an approximation to $\psi(x, n\Delta t)$. Given a solution at time $t = n\Delta t$ these equations define the solution at the next time step $t = (n+1)\Delta t$. By letting $\varsigma = \varsigma^{n+1}$ and $\psi = \psi^{n+1}$ a single step of the difference equations can be written in

the form

$$\varsigma - \epsilon\varsigma_{xx} = F$$

$$\varsigma = \psi_{xx} \tag{1}$$

$$\psi = \psi_x = 0 \text{ at } x = 0, 1$$

where $\epsilon = \nu\Delta t/2$ and $F = F(x)$ depends on f, Δt and the solution at the previous time step

$$F = \varsigma^n + \epsilon\varsigma^n_{xx} + \Delta t f(x, t + \frac{\Delta t}{2})$$

We shall refer to (1) as the ***single time step model problem.*** Discretization of (1) in space leads to the ***discrete single time step model.***

$$\begin{aligned} z_\nu - \epsilon D_+ D_- z_\nu &= F_\nu \qquad \nu = 1, 2, \ldots, N-1 \\ z_\nu &= D_+ D_- \phi_\nu \qquad \nu = 0, 1, \ldots, N \\ \phi_0 &= \phi_N = 0 \\ D_{l,q}\phi_0 &= D_{r,q}\phi_N = 0 \end{aligned} \tag{2}$$

ϕ_ν, which is defined for $\nu = -1, 0, 1, \ldots, N-1, N, N+1$, is an approximation to $\psi_\nu := \psi(x_\nu)$, $x_\nu = \nu h$, $\nu = -1, 0, \ldots, N, N+1$ with $Nh = 1$. Second order centred finite differences are used to approximate d^2/dx^2. The derivative boundary conditions are differenced with approximations of order q. $D_{l,q}$ will be the approximation to d/dx at the left boundary and $D_{r,q}$ the approximation at the right boundary. This system represents one time step of the Crank-Nicholson method applied to the space discrete time dependent problem. The boundary condition approximations are for now left unspecified. It is convenient to introduce a shorthand notation for the model problems. Define the continuous and discrete operators L and L_h by

$$L\psi(x) := \psi_{xx} - \epsilon\psi_{xxxx}$$

$$L_h\phi_\nu := D_+ D_- \phi_\nu - \epsilon(D_+ D_-)^2\phi_\nu$$

and the boundary operators B and B_h by

$$B\psi = \begin{bmatrix} \psi(0) \\ \psi(1) \\ \psi_x(0) \\ \psi_x(1) \end{bmatrix} \quad , \qquad B_h\phi = \begin{bmatrix} \phi_0 \\ \phi_N \\ D_{l,q}\phi_0 \\ D_{r,q}\phi_N \end{bmatrix}$$

With this notation the continuous and discrete single time step model problems are written as

$$L\psi = F \qquad B\psi = 0 \tag{3}$$

and

$$L_h\phi = F^h \qquad B_h\phi = 0 \tag{4}$$

respectively. Proceeding one step further we will write (3) and (4) as

$$\mathbf{L}\psi = \mathbf{F} \tag{5}$$

$$\mathbf{L}_h\phi = \mathbf{F}^h \tag{6}$$

We investigate some of the properties of the discrete solution (ϕ_ν, z_ν) of (2) as an approximation of the continuous solution $(\psi(x), \varsigma(x))$ of (1). An equation for the normalized error in the stream function $e^h_\nu := (\psi_\nu - \phi_\nu)/h^2$ can be written down as

$$L_h e^h = G \qquad B_h e^h = g = \begin{bmatrix} 0 \\ 0 \\ g_0 \\ g_1 \end{bmatrix} \tag{7}$$

or

$$\mathbf{L}_h e^h = \mathbf{G}^h$$

The functions G, g_0 and g_1 are related to the truncation errors. Since these functions are known as continuous functions we can form a continuous error equation as an approximation to the exact discrete error equation (2). The continuous error equation is

$$Le = G \qquad Be = g = \begin{bmatrix} 0 \\ 0 \\ g_0 \\ g_1 \end{bmatrix} \tag{8}$$

We emphasize that it is the exact error e^h which is of importance and that e is only an approximation to e^h. We initially choose to consider the second order form of the discrete boundary conditions with

$$D_{l,2} = D_0 \text{ and } D_{r,2} = D_0 \tag{9}$$

This corresponds to an approximation to the vorticity on the left hand boundary of

$$z_0 = \frac{2\phi_1}{h^2} \tag{10}$$

This approximation is alternatively referred to as the *conventional approximation* (Gupta and Manohar [1979]) or Thom's formula. The truncation error in formula (10) is formally only $O(h)$. We shall also be interested in using higher order approximations for the no slip condition. The second order approximation was centred but the higher order schemes are one sided; there are more values of the stream function used from within the interval but still only a single point outside the boundary (i.e. at x_{-1} or x_{N+1}). The third order approximations can be derived from Taylor expansions of ψ and are

$$D_{l,3}\phi_0 := \frac{-2\phi_{-1} - 3\phi_0 + 6\phi_1 - \phi_2}{6h} = 0$$
$$D_{r,3}\phi_N := \frac{\phi_{N-2} - 6\phi_{N-1} + 3\phi_N + 2\phi_{N+1}}{6h} = 0$$

The fourth order approximations are

$$D_{l,4}\phi_0 := \frac{-3\phi_{-1} - 10\phi_0 + 18\phi_1 - 6\phi_2 + \phi_3}{12h} = 0$$
$$D_{r,4}\phi_N := \frac{-\phi_{N-3} + 6\phi_{N-2} - 18\phi_{N-1} + 10\phi_N + 3\phi_{N+1}}{12h} = 0$$

The truncation errors in approximation (2) with the conventional approximation for boundary conditions are all $O(h^2)$. The discrete stream function computed from (2) will thus be second order accurate provided that the normalized error satisfies $\|e^h\| = \|\mathbf{L}_h{}^{-1}\mathbf{G}^h\| = O(1)$. The discrete vorticity $z_\nu = D_+D_-\phi_\nu$ will also be second order provided that the error e^h_ν is sufficiently smooth. This can been seen as follows. The computed stream function ϕ_ν is related to the true solution $\psi(x_\nu)$ by $\phi_\nu = \psi(x_\nu) - h^2 e^h_\nu$. The discrete vorticity is thus given by

$$z_\nu \equiv D_+D_-\phi_\nu = D_+D_-\psi_\nu - h^2 D_+D_- e^h_\nu$$
$$= \psi_{xx}(x_\nu) + O(h^2) - h^2 D_+D_- e^h_\nu$$

If e_ν^h is sufficiently smooth then $D_+D_-e_\nu^h$ looks as if a second derivative and $\|D_+D_-e_\nu^h\|$ will be $O(1)$. The approximation to the vorticity will then be second order.

It seems useful to consider the problem of obtaining an accurate answer to the vorticity in the manner outlined above. That is, to think of obtaining an approximation to the stream function with a smooth error. Divided difference approximations to higher derivatives of the stream function will then have the same accuracy as the stream function itself. This is one reason why we consider approximating $\partial\psi/\partial n = 0$ rather than the vorticity on the wall. If ϵ is not small compared to h^2 we will show that the vorticity computed from (2) with the boundary conditions (9) will be second order. This is despite the fact that the approximation to the vorticity on the boundary is formally only *first order*.

The parameter ϵ in (1) and (2) results from the application of an implicit time stepping procedure and is related to the time step by $\epsilon = \nu\Delta t/2$. In general the solutions to (1) or (2) will have boundary layers which depend on ϵ. In the context of time stepping, however, we expect (hope?) that as $\Delta t \to 0$ and $\epsilon \to 0$ the solution will only have boundary layers that depend on ν and not Δt. The ϵ dependent boundary layers will hopefully be suppressed by the form of the forcing and the boundary conditions. Even if the forcing satisfies compatibility conditions we will see that the discrete approximations to the boundary conditions can create ϵ dependent boundary layers so that as $\epsilon \to 0$ the error in the solution to (2) is not smooth and the vorticity is only first order accurate.

Synopsis

We now outline the contents of the rest of this section and state some propositions which give the basic results which are obtained. We first look at the continuous single time step model (1). We find that the solution is composed of a smooth part with derivatives bounded independently of ϵ and a boundary layer part. The bound-

ary layer can be suppressed to a given order in ϵ provided the forcing F satisfies certain compatibility conditions. In the propositions to follow we state our results only to first order. However, in later sections more detailed results are usually obtained. The full set of compatibilty conditions is given in section 5.2.2. For presenting the propositions we will need the following definition.

Definition. *A function F is said to satisfy the compatibility conditions to $O(\epsilon)$ if*

$$\int_0^1 F(x)dx = O(\epsilon) \text{ and } \int_0^1 \int_0^{x'} F(x')dx'dx = O(\epsilon)$$

Then in section 5.2.2 we show the following.

Proposition 1. *If F satisfies the compatibility conditions to $O(\epsilon)$ then the solution to (1) satisfies*

$$\zeta = \psi_{xx} = F + O(\epsilon)$$

We next look at the discrete single time step model (2) and the corresponding error equation (7). This discrete error equation is approximated by the continuous error equation (8). Using the results obtained from the continuous single time step model the behaviour of the solution to the continuous error equation is found. The important difference between the continuous single time step model (3) and the continuous error equation is that the normal derivative boundary conditions in (8) are inhomogeneous. We find that even when F satisfies the compatibilty conditions the error in the vorticity is $O(\max(h^2, h^q/\sqrt{\epsilon}))$, where q is the order of accuracy of the boundary conditions. In particular we prove

Proposition 2. *If F satisfies the compatibility conditions to $O(\epsilon)$ and e is the solution to the continuous error equation (8) then*

$$e_{xx} = G(x) - \frac{g_0}{\sqrt{\epsilon}} e^{-x/\sqrt{\epsilon}} + \frac{g_1}{\sqrt{\epsilon}} e^{-(1-x)/\sqrt{\epsilon}} + O(\epsilon)$$

(For boundary conditions of order q, g_0 and g_1 are $O(h^{q-2})$.)

For typical time steps $\sqrt{\epsilon} = O(h)$ and thus if the boundary conditions are only second order it looks like the vorticity is only first order. Of course if there are strong boundary layers the continuous error equation may not be a good approximation to its discrete counterpart. In such a case it will be necessary to consider the discrete error equation.

The analysis of the discrete error equation is performed next. The basic result is contained in the following proposition.

Proposition 3. *If F satisfies the compatibility conditions to $O(\epsilon)$ and e^h is the solution to the discrete error equations (7) then*

$$D_+D_-(e^h)_\nu = \bar{d}(\kappa)\left(g_0\kappa^{-\nu} - g_1\kappa^{\nu-N}\right) + O(h)$$

where

$$\kappa = 1 + \frac{h^2}{2\epsilon} + \sqrt{\frac{h^2}{2\epsilon}\left(2 + \frac{h^2}{2\epsilon}\right)}$$

and where $\bar{d}(\kappa)$ depends on the type of boundary condition. For the second, third and fourth boundary conditions we considered

$$\bar{d}(\kappa) = \begin{cases} O(1/\sqrt{\epsilon}) & h \ll \sqrt{\epsilon} \\ O(1/h) & \sqrt{\epsilon} \ll h \end{cases}$$

and $\bar{d}(\kappa)$ varies smoothly between these two limiting cases.

Thus if higher order boundary conditions are used and $\epsilon = O(h^2)$ then the vorticity will be second order. In this case the single time step model continues to apply for future steps (since the compatibility conditions are satisfied by the forcing at the next time step to $O(\max(h^2, \epsilon))$). However, if only second order boundary conditions are used the analysis breaks down since the forcing for the next time step will not satisfy the compatibilty conditions to even $O(\epsilon)$.

5.2.2 Asymptotic Expansion of the Single Time Step Model

In this section we derive the form of the solution to the continuous single time step

model.

$$\zeta - \epsilon\zeta_{xx} = F$$

$$\zeta = \psi_{xx} \tag{1}$$

$$\psi = \psi_x = 0 \quad \text{at } x = 0, 1$$

We construct an asymptotic expansion (for $\epsilon \to 0$) of the solution to (1). The solution will be found to consist of a smooth part and a boundary layer part. We derive compatibilty conditions that F must satisfy in order to suppress the boundary layer to a given order in ϵ. The system (1) can be solved by variation of parameters

$$\zeta(x) = \frac{1}{2\sqrt{\epsilon}} \left\{ \int_0^x F(\xi) e^{(\xi - x)/\sqrt{\epsilon}} d\xi + \int_x^1 F(\xi) e^{(x-\xi)/\sqrt{\epsilon}} d\xi \right\} + a e^{-x/\sqrt{\epsilon}} + b e^{-(1-x)/\sqrt{\epsilon}}$$

where a and b are constants which are determined by the boundary conditions at $x = 0$ and 1.

The particular solution (the two integral terms) can be developed into an asymptotic expansion for $\epsilon \to 0$ using integration by parts, assuming that $F \in C^{(p)}[0, 1]$. The first integral becomes

$$\begin{aligned}
\frac{1}{2\sqrt{\epsilon}} \int_0^x F(\xi) e^{(\xi - x)/\sqrt{\epsilon}} d\xi &= \frac{1}{2} \left[F(\xi) e^{(\xi - x)/\sqrt{\epsilon}} \right]_0^x - \frac{1}{2} \int_0^x F'(\xi) e^{(\xi - x)/\sqrt{\epsilon}} d\xi \\
&= \ldots = \frac{1}{2} \sum_{m=0}^{p-1} \left[(-1)^m \epsilon^{m/2} F^{(m)}(\xi) e^{(\xi - x)/\sqrt{\epsilon}} \right]_0^x \\
&\quad + \frac{1}{2} (-1)^p \epsilon^{(p-1)/2} \int_0^x F^{(p)}(\xi) e^{(\xi - x)/\sqrt{\epsilon}} d\xi \\
&= \frac{1}{2} \sum_{m=0}^{p-1} (-1)^m \epsilon^{m/2} \left\{ F^{(m)}(x) - e^{-x/\sqrt{\epsilon}} F^{(m)}(0) \right\} \\
&\quad + \text{remainder integral}
\end{aligned}$$

and similarly for the second integral

$$\begin{aligned}
\frac{1}{2\sqrt{\epsilon}} \int_x^1 F(\xi) e^{(x - \xi)/\sqrt{\epsilon}} d\xi &= \frac{1}{2} \sum_{m=0}^{p-1} \epsilon^{m/2} \left\{ F^{(m)}(x) - e^{-(1-x)/\sqrt{\epsilon}} F^{(m)}(1) \right\} \\
&\quad + \frac{1}{2} \epsilon^{(p-1)/2} \int_x^1 F^{(p)}(\xi) e^{(\xi - x)/\sqrt{\epsilon}} d\xi
\end{aligned}$$

These results can be combined to give

$$\begin{aligned}\zeta(x) =& \frac{1}{2}\sum_{m=0}^{p-1}[1+(-1)^m]\epsilon^{m/2}F^{(m)}(x)\\ &+ e^{-x/\sqrt{\epsilon}}\left\{a - 1/2\sum_{m=0}^{p-1}(-1)^m\epsilon^{m/2}F^{(m)}(0)\right\}\\ &+ e^{-(1-x)/\sqrt{\epsilon}}\left\{b - 1/2\sum_{m=0}^{p-1}\epsilon^{m/2}F^{(m)}(1)\right\} \quad + O(\epsilon^{p/2})\end{aligned}$$

In the first sum every second term drops out so that the solution for the vorticity is of the form

$$\begin{aligned}\zeta(x) = F(x) + \epsilon F''(x) + \epsilon^2 F^{(4)} + \ldots + \epsilon^p F^{(2p)} + O(\epsilon^{p+1})\\ + \frac{A}{\sqrt{\epsilon}}e^{-x/\sqrt{\epsilon}} + \frac{B}{\sqrt{\epsilon}}e^{-(1-x)/\sqrt{\epsilon}}\end{aligned} \tag{2}$$

Hence, asymptotically the solution for ζ consists of a smooth part plus boundary layers at $x = 0, 1$. The boundary layers arise from the singular nature of the problem in the limit $\epsilon \to 0$. The boundary layers are needed to match the boundary conditions and the coefficients of the boundary layers depend on the forcing function F. We are interested in the case where the solution is smooth up to the boundary; this is the case only if F satisfies appropriate compatibility conditions. These conditions will now be derived.

The solution derived for ζ can be integrated twice to give the general solution for ψ.

$$\begin{aligned}\psi(x) = &\int_0^x\int_0^{x'}\zeta_s(x'')dx''dx'\\ &- \sqrt{\epsilon}A(1 - e^{-x/\sqrt{\epsilon}} - x(1 - e^{-1/\sqrt{\epsilon}}))\\ &- \sqrt{\epsilon}B(1 - e^{-(1-x)/\sqrt{\epsilon}} - (1-x)(1 - e^{-1/\sqrt{\epsilon}}))\\ &+ C + Dx\end{aligned} \tag{3}$$

The terms which A and B multiply in (3) have been chosen to be zero at the two boundaries. ζ_s is the smooth part of the vorticity which asymptotically is given by

$$\zeta_s(x) \sim F(x) + \epsilon F''(x) + \epsilon^2 F^{(4)}(x) + \ldots$$

The constants A, B, C and D are chosen to satisfy the boundary conditions $\psi(0) = \psi(1) = \psi_x(0) = \psi_x(1) = 0$. Initially we need only look at A and B which are determined from the following two conditions on the vorticity.

$$\int_0^1 \varsigma(x)dx = [\psi_x]_0^1 = 0$$

$$\int_0^1 \int_0^x \varsigma(x')dx'dx = [\psi]_0^1 - \psi_x(0) = 0$$

Whence

$$(1 - e^{-1/\sqrt{\epsilon}})(A + B) + \int_0^1 \varsigma_s(x)dx = 0$$

$$A[1 - \sqrt{\epsilon}(1 - e^{-1/\sqrt{\epsilon}})] + B[\sqrt{\epsilon}(1 - e^{-1/\sqrt{\epsilon}}) - e^{-1/\sqrt{\epsilon}}] + \int_0^1 \int_0^x \varsigma_s(x')dx'dx = 0$$

Define the single and double integrals I_1 and I_2 by

$$I_1 := \int_0^1 \varsigma_s(x)dx$$

$$I_2 := \int_0^1 \int_0^x \varsigma_s(x')dx'dx$$

A and B satisfy the matrix equation

$$\begin{bmatrix} 1 - e^{-1/\sqrt{\epsilon}} & 1 - e^{-1/\sqrt{\epsilon}} \\ 1 - \sqrt{\epsilon}(1 - e^{-1/\sqrt{\epsilon}}) & \sqrt{\epsilon}(1 - e^{-1/\sqrt{\epsilon}}) - e^{-1/\sqrt{\epsilon}} \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \end{bmatrix}$$

The coefficients of the boundary layers will vanish if and only if the integrals I_1 and I_2 vanish. Ignoring the exponentially small terms, $e^{-1/\sqrt{\epsilon}}$, gives

$$\begin{bmatrix} 1 & 1 \\ 1 - \sqrt{\epsilon} & \sqrt{\epsilon} \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \end{bmatrix}$$

which is solved for A and B

$$\begin{bmatrix} A \\ B \end{bmatrix} = \frac{1}{1 - 2\sqrt{\epsilon}} \begin{bmatrix} -\sqrt{\epsilon} & 1 \\ 1 - \sqrt{\epsilon} & -1 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \end{bmatrix}$$

To leading order in ϵ the coefficients of the boundary layer terms in $\varsigma = \psi_{xx}$ are thus

$$A \sim I_2$$

$$B \sim (I_1 - I_2)$$

From the asymptotic expansion for ζ_s we can easily relate I_1 and I_2 to F.

$$\zeta_s(x) \sim F(x) + \epsilon F''(x) + \epsilon^2 F^{(4)}(x) + \dots$$

$$I_1 = \int_0^1 F(x)dx + \epsilon \int_0^1 F''(x)dx + \epsilon^2 \int_0^1 F^{(4)}(x)dx + \dots$$

$$I_2 = \int_0^1 \int_0^x F(x)dx'dx + \epsilon \int_0^1 \int_0^x F''(x)dx'dx + \dots$$

Thus we see that the strength of the boundary layer in ζ (and also in ψ) is related to the vanishing of a number of integrals of F.

Definition 1. *$F \in C^\infty[0,1]$ is said to satisfy the compatibilty conditions if*

$$\begin{aligned} \int_0^1 F^{(2m)}(x)dx &= 0 \\ \int_0^1 \int_0^x F^{(2m)}(x')dx'dx &= 0 \end{aligned} \tag{4}$$

for $m = 0, 1, 2, \dots$. These conditions imply that that the single and double integrals of F are zero and that the derivatives of F evaluated at the end points satisfy

$$\begin{aligned} \left[F^{(2m+1)}(x)\right]_0^1 &= 0 \\ \left[F^{(2m)}(x)\right]_0^1 &= F^{(2m+1)}(0) \end{aligned} \tag{5}$$

for $m = 0, 1, 2, \dots$

If F satisfies all the compatibilty conditions then the solution to (1) will consist entirely of the smooth solution ζ_s. If F only satisfies the first M conditions then the coefficient of the boundary layer terms in solution for ζ will be $O(\epsilon^{M-\frac{1}{2}})$.

To complete the calculation we need to determine the coefficients C and D in the equation for the stream function (3). These constants are determined from $\psi(0) = 0$ and $\psi(1) = 0$.

$$0 = \psi(0) = C$$

$$0 = \psi(1) = I_2 + C + D$$

Thus $C = 0$ and $D = -I_2$. Hence if F satisfies the compatibilty conditions D will be zero.

5.2.3 Discrete Approximation of the Single Time Step Model

We now consider the discrete single time step model equations. These equations were introduced previously, where they were written as

$$L_h\phi = F^h \qquad B_h\phi = 0 \tag{1}$$

To analyze the error in the stream function we form the equation for the error. Recall that the normalized error in the stream function was defined as

$$e_\nu^h := (\psi_\nu - \phi_\nu)/h^2$$

and satisfies the discrete error equations given by

$$\begin{aligned} D_+D_-e_\nu^h - \epsilon(D_+D_-)^2 e_\nu^h &= G_\nu \qquad \nu = 1,2,\ldots,N-1 \\ e_0^h = e_N^h &= 0 \\ D_{l,q}e_0^h = g_0 \qquad D_{r,q}e_N^h &= g_1 \end{aligned} \tag{2}$$

or

$$L_he^h = G \qquad B_he^h = g = \begin{bmatrix} 0 \\ 0 \\ g_0 \\ g_1 \end{bmatrix}$$

G_ν, g_0, and g_1 are the truncation errors of the space differences divided by h^2. Explicitly, G_ν is given by

$$\begin{aligned} G_\nu &= G(x_\nu) \\ G(x) &:= \frac{1}{h^2}(L_h - L)\psi(x) \\ &= \left[(D_+D_- - \frac{d^2}{dx^2}) - \epsilon((D_+D_-)^2 - \frac{d^4}{dx^4})\right]\psi(x) \\ &= \frac{1}{12}\psi^{(4)} + \frac{2h^2}{6!}\psi^{(6)} + O(h^4) - \epsilon\left[\frac{56}{6!}\psi^{(6)} + O(h^2)\right] \end{aligned}$$

The truncation errors in the boundary terms will be of the form

$$\begin{aligned} g_0 &:= \frac{1}{h^2}(D_{l,q} - \frac{d}{dx})\psi(0) = C_{0q}h^{q-2}\frac{d^{q+1}\psi}{dx^{q+1}}(0) + O(h^{q-1}) \\ g_1 &:= \frac{1}{h^2}(D_{r,q} - \frac{d}{dx})\psi(1) = C_{1q}h^{q-2}\frac{d^{q+1}\psi}{dx^{q+1}}(1) + O(h^{q-1}) \end{aligned}$$

The approximations are accurate to order q. In particular for the conventional approximation $D_{l,q} = D_0$ and $D_{r,q} = D_0$ we have

$$g_0 := \frac{1}{h^2}(D_0 - \frac{d}{dx})\psi(0) = \frac{1}{6}\psi_{xxx}(0) + \frac{h^2}{5!}\psi^{(5)}(0) + O(h^4)$$
$$g_1 := \frac{1}{h^2}(D_0 - \frac{d}{dx})\psi(1) = \frac{1}{6}\psi_{xxx}(1) + \frac{h^2}{5!}\psi^{(5)}(1) + O(h^4)$$

Since $G(x)$ is known as a continuous function we can form a continuous error equation as an approximation to the exact discrete error equation (2). The solution to this continuous error equation should be close to the solution to the discrete error equation provided that the solution we find is smooth. We consider

$$Le = G \qquad Be = g = \begin{bmatrix} 0 \\ 0 \\ g_0 \\ g_1 \end{bmatrix} \tag{3}$$

Since G involves only even derivatives of ψ and assuming F satisfies the compatibility conditions it is easily checked that G also satisfies these compatibility conditions. We decompose e into a *forced* and *boundary* part

$$e = e_f + e_b$$

where e_f and e_b satisfy

$$Le_f = G \qquad Be_f = 0$$

$$Le_b = 0 \qquad Be_b = g$$

The results on asymptotic expansions of the continuous problem obtained in the previous section allows us immediately to write down the solution e_f.

$$e_f \sim \int_0^x \int_0^{x'} G(x'') + \epsilon G_{xx} \ldots dx''dx'$$

Integrating gives

$$\begin{aligned} e_f \sim & \int_0^x \int_0^{x'} G(x'')dx''dx' + \epsilon G(x) + \epsilon^2 G_{xx}(x) + \ldots \\ & - \left[\epsilon G(0) + \epsilon^2 G_{xx}(0) + \ldots\right] \\ & - \left[\epsilon G_x(0) + \epsilon^2 G_{xxx}(0) + \ldots\right] x \end{aligned}$$

Hence the continuous approximation of the forced part of the error is smooth and $O(1)$. (If F only statisfies the compatibilty conditions to $O(\epsilon)$ then e_f will also contain boundary layer terms. These terms will be small, however, having strength $O(\epsilon^{3/2})$.)

Likewise we can solve for the the boundary part e_b of e as

$$e_b = g_0\sqrt{\epsilon}(1 - e^{-x/\sqrt{\epsilon}}) - g_1\sqrt{\epsilon}(1 - e^{-(1-x)/\sqrt{\epsilon}})$$
$$- \sqrt{\epsilon}(g_0 + g_1)x + \text{ smaller terms}$$

That is the first and second derivative of e_b are

$$(e_b)_x \sim g_0 e^{-x/\sqrt{\epsilon}} + g_1 e^{-(1-x)/\sqrt{\epsilon}}$$
$$(e_b)_{xx} \sim -\frac{g_0}{\sqrt{\epsilon}} e^{-x/\sqrt{\epsilon}} + \frac{g_1}{\sqrt{\epsilon}} e^{-(1-x)/\sqrt{\epsilon}}$$

Hence the continuous approximation of the boundary part of the vorticity error has boundary layers with thickness $O(\sqrt{\epsilon})$ and strength $O(g_0/\sqrt{\epsilon})$ at $x = 0$ and strength $O(g_1/\sqrt{\epsilon})$ at $x = 1$.

For completeness we include the full calculation of e_b. e_b is a solution of the homogeneous system and so it has the form

$$e_b = Ae^{-x/\sqrt{\epsilon}} + Be^{-(1-x)/\sqrt{\epsilon}} + C + Dx$$

The coefficients are determined from the boundary conditions. Substituting into the boundary conditions gives

$$A + Be^{-1/\sqrt{\epsilon}} + C = 0$$
$$Ae^{-1/\sqrt{\epsilon}} + B + C + D = 0$$
$$\frac{1}{\sqrt{\epsilon}}(-A + Be^{-1/\sqrt{\epsilon}}) + D = g_0$$
$$\frac{1}{\sqrt{\epsilon}}(-Ae^{-1/\sqrt{\epsilon}} + B) + D = g_1$$

Ignoring exponentially small terms $e^{-1/\sqrt{\epsilon}}$ gives $C = -A$, $D = A - B$ and

$$\begin{bmatrix} -(1-\sqrt{\epsilon}) & -\sqrt{\epsilon} \\ \sqrt{\epsilon} & 1-\sqrt{\epsilon} \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \sqrt{\epsilon} \begin{bmatrix} g_0 \\ g_1 \end{bmatrix}$$

Hence if we denote exponentially small terms by *e.s.t* the coefficients A and B are

$$A = -\frac{\sqrt{\epsilon}}{1-2\sqrt{\epsilon}}[g_0 + \sqrt{\epsilon}(g_1 - g_0)] + e.s.t = -\sqrt{\epsilon}g_0 + O(\epsilon)$$

$$B = \frac{\sqrt{\epsilon}}{1-2\sqrt{\epsilon}}[g_1 - \sqrt{\epsilon}(g_1 - g_0)] + e.s.t. = \sqrt{\epsilon}g_1 + O(\epsilon)$$

and

$$e_b = Ae^{-x/\sqrt{\epsilon}} + Be^{-(1-x)/\sqrt{\epsilon}} - A(1-x) - Bx$$

5.2.4 Asymptotic Expansion of the Error

Now we would like to have a decomposition of the discrete error e^h of the single time step model into a smooth forced part and a boundary layer part analogous to the splitting which we have demonstrated in the previous section for the continuous error approximation e. We write

$$e^h = e_f^h + e_b^h \tag{1}$$

where e_f^h and e_b^h satisfy

$$L_h e_f^h = G \qquad B_h e_f^h = 0 \tag{2}$$

$$L_h e_b^h = 0 \qquad B_h e_b^h = g \tag{3}$$

The behaviour of these two parts of the error is the subject of the following two propositions.

Proposition 1. e_b^h , *the boundary part of the error, satisfies*

$$D_+D_-(e_b^h)_\nu = \bar{d}(\kappa)\left(g_0\kappa^{-\nu} - g_1\kappa^{\nu-N}\right)$$

where

$$\kappa = 1 + \frac{h^2}{2\epsilon} + \sqrt{\frac{h^2}{2\epsilon}\left(2 + \frac{h^2}{2\epsilon}\right)}$$

and where $\bar{d}(\kappa)$ *depends on the type of the boundary conditions. For the second, third and fourth order boundary conditions considered here*

$$\bar{d}(\kappa) = \begin{cases} O(1/\sqrt{\epsilon}) & h \ll \sqrt{\epsilon} \\ O(1/h) & \sqrt{\epsilon} \ll h \end{cases}$$

Proposition 2. *If F satisfies the compatibility conditions then e_f^h, the forced part of the error, can be decomposed as*

$$e_f^h = e_{fs}^h + e_{fb}^h$$

where e_{fs}^h is the smooth part of the forced error and e_{fb}^h is the boundary part of the forced error. Then

$$D_+D_-(e_{fs}^h)_\nu = O(1)$$

$$D_+D_-(e_{fb}^h)_\nu = K_q h^2 \bar{d}(\kappa)\left[G'(0)\kappa^{-\nu} - G'(1)\kappa^{\nu-N}\right] + O(h^2)$$

where $K_q = O(1)$.

From these two propositions the conclusion of Proposition 3 of section 5.2.1 follows. Proposition 1 is the subject of the rest of this section and Proposition 2 is a consequence of the results of the next section.

We consider the solution of (3) for e_b^h. The homogeneous difference equation for e_b^h has the general solution

$$(e_b^h)_\nu = A\kappa^{-\nu} + B\kappa^{\nu-N} + C + Dx_\nu \tag{4}$$

where κ is the larger root of

$$\kappa - \frac{\epsilon}{h^2}(\kappa - 1)^2 = 0 \tag{5}$$

(Note that the two roots of (5) satisfy $\kappa_1\kappa_2 = 1$.) The larger root is

$$\kappa = 1 + \frac{h^2}{2\epsilon} + \sqrt{\frac{h^2}{2\epsilon}\left(2 + \frac{h^2}{2\epsilon}\right)} \tag{6}$$

There are two interesting limits.

I. $h \ll \sqrt{\epsilon} \ll 1$. This is the case where the possible boundary layer is well resolved by the mesh.

II. $\sqrt{\epsilon} \ll h \ll 1$. This is the case where the boundary layer is not well resolved because it is small compared with h.

In these limiting cases

$$\kappa \sim \begin{cases} 1 + h/\sqrt{\epsilon} & h \ll \sqrt{\epsilon} \\ h^2/\epsilon & \sqrt{\epsilon} \ll h \end{cases} \tag{7}$$

The coefficients A, B, C, D are determined from the boundary conditions. Substituting into the boundary conditions gives

$$\begin{aligned} A + B\kappa^{-N} + C &= 0 \\ A\kappa^{-N} + B + C + D &= 0 \\ d(\kappa)(A - B\kappa^{-N}) + D &= g_0 \\ d(\kappa)(A\kappa^{-N} - B) + D &= g_1 \end{aligned} \tag{8}$$

where

$$d(\kappa) := \frac{\kappa^{-1} - \kappa}{2h} = -\frac{1}{\sqrt{\epsilon}} \left(1 + \frac{h^2}{4\epsilon}\right)^{1/2} \tag{9}$$

for the second order boundary conditions. Clearly $d(\kappa) \gg 1$ for ϵ small; in the two limits of interest

$$d(\kappa) \sim \begin{cases} -1/\sqrt{\epsilon} & h \ll \sqrt{\epsilon} \\ -h/2\epsilon & \sqrt{\epsilon} \ll h \end{cases} \tag{10}$$

In either case κ^{-N} is negligibly small and so the coefficients are well approximated by

$$\begin{aligned} A + C &= 0 \\ B + C + D &= 0 \\ Ad(\kappa) + D &= g_0 \\ -Bd(\kappa) + D &= g_1 \end{aligned} \tag{11}$$

so

$$\begin{aligned} A &\sim g_0/d(\kappa) \\ B &\sim -g_1/d(\kappa) \\ C &\sim -A \\ D &\sim A - B \end{aligned} \tag{12}$$

The contribution of e_b^h to the vorticity error is $D_+D_-e_b^h$ which is given by

$$\begin{aligned} D_+D_-(e_b^h)_\nu &= A\frac{(\kappa-1)^2}{h^2\kappa}\kappa^{-\nu} + B\frac{(\kappa-1)^2}{h^2\kappa}\kappa^{\nu-N} \\ &\sim \bar{d}(\kappa)\left(g_0\kappa^{-\nu} - g_1\kappa^{\nu-N}\right) \end{aligned} \tag{13}$$

where

$$\begin{aligned} \bar{d}(\kappa) &:= \frac{1}{d(\kappa)}\frac{(\kappa-1)^2}{h^2\kappa} = \frac{1}{d(\kappa)}\frac{1}{\epsilon} \\ &= -\frac{1}{\sqrt{\epsilon}}\left(1+\frac{h^2}{4\epsilon}\right)^{-1/2} = \left(\epsilon+\frac{h^2}{4}\right)^{-1/2} \\ &\leq \min\left(\frac{1}{\sqrt{\epsilon}}, \frac{2}{h}\right) \end{aligned} \tag{14}$$

In the two limiting cases

$$\bar{d}(\kappa) \sim \begin{cases} -1/\sqrt{\epsilon} & h \ll \sqrt{\epsilon} \\ -2/h & \sqrt{\epsilon} \ll h \end{cases}$$

The case $h \ll \sqrt{\epsilon}$ follows the continuous case closely as expected. In the unresolved case $\sqrt{\epsilon} \ll h$ it is interesting that the vorticity error costs only a power of h at the boundary.

In the case of no slip boundary conditions of higher order the calculation of e_b^h remains the same with different formulae for $d(\kappa)$ and $\bar{d}(\kappa)$. For third order boundary conditions $d(\kappa)$ has the form

$$d(\kappa) = \frac{-2\kappa - 3 + 6\kappa^{-1} - \kappa^{-2}}{6h} \sim \begin{cases} -1/\sqrt{\epsilon} & h \ll \sqrt{\epsilon} \\ -h/3\epsilon & \sqrt{\epsilon} \ll h \end{cases} \tag{15}$$

and so

$$\bar{d}(\kappa) = \frac{1}{\epsilon}\frac{1}{d(\kappa)} \sim \begin{cases} -1/\sqrt{\epsilon} & h \ll \sqrt{\epsilon} \\ -3/h & \sqrt{\epsilon} \ll h \end{cases} \tag{16}$$

Similarly, for fourth order boundary conditions

$$d(\kappa) = \frac{-3\kappa - 10 + 18\kappa^{-1} - 6\kappa^{-2} + \kappa^{-3}}{12h} \sim \begin{cases} -1/\sqrt{\epsilon} & h \ll \sqrt{\epsilon} \\ -h/4\epsilon & \sqrt{\epsilon} \ll h \end{cases} \tag{17}$$

and

$$\bar{d}(\kappa) = \frac{1}{\epsilon}\frac{1}{d(\kappa)} \sim \begin{cases} -1/\sqrt{\epsilon} & h \ll \sqrt{\epsilon} \\ -4/h & \sqrt{\epsilon} \ll h \end{cases} \tag{18}$$

Thus we conclude that the error in the vorticity from e_b^h costs at most a power of h. That is, if a higher order boundary condition $D_{l,q}\phi_0 = D_{r,q}\phi_N = 0$ is used such that $h^2 g_0$, $h^2 g_1$ are $O(h^q)$ then the strength of the boundary layer in $h^2 D_+ D_- e_b^h$ is at most $O(h^{q-1})$.

5.2.5 Asymptotic Expansion of the Discrete Single Time Step Model

We now construct an asymptotic expansion of the discrete single time step model

$$L_h\phi = F^h \qquad B_h\phi = 0 \tag{1}$$

which is contained in the following result.

Proposition 1. *If F satisfies the compatibility conditions then ϕ can be decomposed into a smooth part and a boundary part as*

$$\phi = \phi^s + \phi^b$$

The smooth part of the solution has any number of bounded divided differences. In particular,

$$D_+ D_- \phi^s = O(1)$$

The boundary part satisfies

$$D_+ D_- \phi^b = M_q \bar{d}(\kappa) \left[F'(0)\kappa^{-\nu} - F'(1)\kappa^{\nu-N} \right] + O(h^2) \tag{2}$$

where

$$M_q = \begin{cases} -h^2/6 & \text{if } q = 2 \\ O(h^3) & \text{if } q \geq 3 \end{cases}$$

and κ, $\bar{d}(\kappa)$ have been defined in the previous section.

This result is obtained under the assumption that F satisfies the sequence of compatibility conditions identically. In the time dependent problem we expect that the compatibility conditions will only be satisfied to some order in ϵ. The analysis

below still holds as far as the compatibility conditions are satisfied. For example if the compatibility conditions are violated at order ϵ then (2) is modified by terms of order ϵ on the right hand side.

In the continuous case we used variation of parameters to write down the exact solution of the model problem and then obtained the asymptotic expansion by integration by parts. Similarly in the discrete case it is possible to write down the solution using variation of parameters and then use summation by parts to develop the asymptotic expansion of ϕ. Instead we shall proceed more formally by assuming the form of the asymptotic expansion and solving for the coefficients which appear.

We make an ansatz that ϕ has the form

$$\begin{aligned} \phi_\nu \sim & \left(\phi_\nu^{(0)} + \epsilon\phi_\nu^{(1)} + \epsilon^2\phi_\nu^{(2)} + \ldots\right) \\ & + A\kappa^{-\nu} + B\kappa^{\nu-N} + C + Dx_\nu \\ & = \phi_\nu^s + \phi_\nu^b \end{aligned} \tag{3}$$

where ϕ^s is the asymptotically smooth particular solution and ϕ^b is the homogeneous solution needed to satisfy the boundary conditions.

In establishing the above result we shall first determine the boundary conditions such that each of the $\phi^{(m)}$'s in the asymptotic series for ϕ^s is smooth independent of h. In obtaining this expansion we shall use the smoothness of F to make Taylor series expansions but we do not need any compatibility conditions to be satisfied. Next we use this series to calculate A and B, which depend on ϕ^s and hence on F through the boundary conditions. These calculations are the discrete forms of those which were made in 5.2.2 to obtain the boundary layer terms for the continuous single time step model.

We need to prescribe boundary conditions for ϕ^s and ϕ^b. First we concentrate on the smooth part of the solution. We choose the boundary conditions for ϕ^s to

make ϕ^s smooth up to the boundary. We take

$$\phi_0^s = \phi_N^s = 0$$

$$D_+D_-\phi_0^s = z_0^s \sim z_0^{(0)} + \epsilon z_0^{(1)} + \epsilon^2 z_0^{(2)} + \dots$$

$$D_+D_-\phi_N^s = z_N^s \sim z_N^{(0)} + \epsilon z_N^{(1)} + \epsilon^2 z_N^{(2)} + \dots$$

where z_0^s, z_N^s will be specified later.

Substituting this ansatz into the difference equations (1) and equating coefficients of powers of ϵ gives a sequence of equations defining the $\phi^{(m)}$'s

$$D_+D_-\phi_\nu^{(0)} = F_\nu$$

$$D_+D_-\phi_\nu^{(1)} = (D_+D_-)^2\,\phi_\nu^{(0)}$$

$$D_+D_-\phi_\nu^{(2)} = (D_+D_-)^2\,\phi_\nu^{(1)}$$

$$\vdots$$

for $\nu = 1, \dots, N-1$ with boundary conditions

$$\phi_0^{(m)} = \phi_N^{(m)} = 0$$

$$D_+D_-\phi_0^{(m)} = z_0^{(m)}$$

$$D_+D_-\phi_N^{(m)} = z_N^{(m)}$$

Note that each of these problems for the $\phi^{(m)}$ is well defined and that the $\phi^{(m)}$'s are independent of ϵ. We must choose the $z^{(m)}$'s in such a way that each of the $\phi^{(m)}$'s is smooth up to the boundary. We consider the first few terms in turn.

The first term $\phi^{(0)}$ satisfies

$$D_+D_-\phi_\nu^{(0)} = \begin{cases} z_0^{(0)} & \nu = 0 \\ F_\nu & \nu = 1, \dots, N-1 \\ z_N^{(0)} & \nu = N \end{cases}$$

and so $\phi^{(0)}$ has at least two bounded divided differences for any choice of $z_0^{(0)}$ and $z_N^{(0)}$.

At the next order $O(\epsilon)$

$$D_+D_-\phi_\nu^{(1)} = \begin{cases} z_0^{(1)} & \nu = 0 \\ (D_+D_-)^2\phi_\nu^{(0)} & \nu = 1, \dots, N-1 \\ z_N^{(1)} & \nu = N \end{cases}$$

and

$$(D_+D_-)^2\phi_\nu^{(0)} = \begin{cases} D_+D_-F_1 + \frac{1}{h^2}\left(z_0^{(0)} - F_0\right) & \nu = 1 \\ D_+D_-F_\nu & \nu = 2,\ldots,N-2 \\ D_+D_-F_{N-1} + \frac{1}{h^2}\left(z_N^{(0)} - F_N\right) & \nu = N-1 \end{cases}$$

We see that $(D_+D_-)^2\phi^{(0)}$ is bounded independently of h if and only if

$$z_0^{(0)} = F_0 + O(h^2)$$

$$z_N^{(0)} = F_N + O(h^2)$$

This also ensures that $D_+D_-\phi^{(1)}$ is bounded. At this level there is no constraint on $z_0^{(1)}$ and $z_N^{(1)}$.

At second order in ϵ

$$D_+D_-\phi_\nu^{(2)} = \begin{cases} z_0^{(2)} & \nu = 0 \\ (D_+D_-)^2\phi_\nu^{(1)} & \nu = 1,\ldots,N-1 \\ z_N^{(2)} & \nu = N \end{cases}$$

and

$$(D_+D_-)^2\phi_\nu^{(1)} = \begin{cases} (D_+D_-)^2F_\nu & \nu = 3,\ldots,N-3 \\ (D_+D_-)^2F_2 + \frac{1}{h^4}\left(z_0^{(0)} - F_0\right) & \nu = 2 \\ (D_+D_-)^2F_{N-2} + \frac{1}{h^4}\left(z_N^{(0)} - F_N\right) & \nu = N-2 \\ \frac{1}{h^2}\left[D_+D_-F_2 - 2D_+D_-F_1 + z_0^{(1)}\right] & \\ \quad -\frac{2}{h^4}(z_0^{(0)} - F_0) & \nu = 1 \\ \frac{1}{h^2}\left[D_+D_-F_{N-2} - 2D_+D_-F_{N-1} + z_N^{(1)}\right] & \\ \quad -\frac{2}{h^4}(z_N^{(0)} - F_N) & \nu = N-1 \end{cases}$$

At this level there are four conditions which must be satisfied for $(D_+D_-)^2\phi^{(1)}$ to be bounded. They are

$$z_0^{(0)} = F_0 + O(h^4)$$

$$z_N^{(0)} = F_N + O(h^4)$$

$$z_0^{(1)} = 2D_+D_-F_1 - D_+D_-F_2 + O(h^2) = F''(0) + O(h^2)$$

$$z_N^{(1)} = 2D_+D_-F_{N-1} - D_+D_-F_{N-2} + O(h^2) = F''(1) + O(h^2)$$

This process can be continued to higher orders in ϵ; at each order $z_0^{(m)}$ and $z_N^{(m)}$ are chosen such that $D_+D_-\phi^{(m+1)}$ is bounded independently of h. This requires that $(D_+D_-)^2\phi^{(m)}$ be bounded independent of h.

We continue the calculation one further order without showing all the details in order to show more clearly the trend which develops. There are six conditions which must be satisfied for $D_+D_-\phi^{(3)}$ to be bounded. The three conditions on $z_0^{(0)}$, $z_0^{(1)}$ and $z_0^{(2)}$ are

$$
\begin{aligned}
z_0^{(0)} &= F_0 + O(h^6) \\
z_0^{(1)} &= D_+D_-\left(2F_1 - F_2\right) + h^2(D_+D_-)^2\left(2F_2 - F_3\right) + O(h^4) \\
z_0^{(2)} &= (D_+D_-)^2\left(2F_2 - F_3\right) + O(h^2)
\end{aligned}
$$

and the other three conditions are of the same form at the other boundary. Substituting the Taylor series of F into these equations gives expressions for the $z^{(m)}$'s in terms of the F and its derivatives at the boundaries

$$
\begin{aligned}
z_0^{(0)} &= F(0) + O(h^6) \\
z_N^{(0)} &= F(1) + O(h^6) \\
z_0^{(1)} &= F''(0) + \frac{2}{4!}h^2F^{(4)}(0) + O(h^4) \\
z_N^{(1)} &= F''(1) + \frac{2}{4!}h^2F^{(4)}(1) + O(h^4) \\
z_0^{(2)} &= F^{(4)}(0) + O(h^2) \\
z_N^{(2)} &= F^{(4)}(1) + O(h^2)
\end{aligned}
\tag{4}
$$

Now we investigate the homogeneous part which contains the boundary layer terms. The homogeneous solution depends on the smooth particular solution ϕ^s through the boundary conditions. The boundary conditions for ϕ^b are

$$
\begin{aligned}
\phi_0^b &= \phi_N^b = 0 \\
D_{l,q}\phi_0^b &= -D_{l,q}\phi_0^s \\
D_{r,q}\phi_N^b &= -D_{r,q}\phi_N^s
\end{aligned}
$$

Just as in the continuous case there are two constraints which together determine the coefficients A and B of the boundary layer terms. We examine the situation first for second order boundary conditions; the modifications for higher order boundary conditions are then easily made. The conditions which determine A and B are obtained by expressing sums and double sums of $D_+D_-\phi^s$ over the interval $[0,1]$ in terms of the boundary conditions at the ends. They are

$$\begin{aligned} 0 &= D_0\phi_N - D_0\phi_0 \\ &= \left(D_0\phi_N^b - D_0\phi_0^b\right) + \left(D_0\phi_N^s - D_0\phi_0^s\right) \\ &= -(A+B)d(\kappa) + \left(\frac{h}{2}D_+D_-\phi_0^s + \sum_{\nu=1}^{N-1} hD_+D_-\phi_\nu^s + \frac{h}{2}D_+D_-\phi_N^s\right) \end{aligned} \tag{5}$$

and

$$\begin{aligned} 0 = \phi_N - \phi_0 &= \sum_{\nu=0}^{N-1} hD_+\phi_\nu^s \\ &= \sum_{\nu=0}^{N-1} h\left[D_+\phi_0^s + \sum_{\mu=1}^{\nu} hD_+D_-\phi_\mu^s\right] \\ &= D_0\phi_0^s + \sum_{\nu=0}^{N-1} h\left[\frac{h}{2}D_+D_-\phi_0^s + \sum_{\mu=1}^{\nu} hD_+D_-\phi_\mu^s\right] \\ &= -Ad(\kappa) + \sum(\ldots) \end{aligned} \tag{6}$$

We introduce a notation for the trapezoidal sum

$$\mathbf{S}_{\nu_1}^{\nu_2} f := \frac{h}{2}f_{\nu_1} + \sum_{\nu=\nu_1+1}^{\nu=\nu_2-1} hf_\nu + \frac{h}{2}f_{\nu_2}$$

for $0 \le \nu_1 < \nu_2 \le N$ and also for the double summation

$$\Delta f := \sum_{\nu=0}^{N-1} h\left[\frac{h}{2}f_0 + \sum_{\mu=1}^{\nu} hf_\mu\right]$$

Then the equations (5) and (6) become

$$(A+B)d(\kappa) = \mathbf{S}_0^N D_+D_-\phi^s \tag{7}$$

and

$$Ad(\kappa) = \Delta(D_+D_-\phi^s) \tag{8}$$

These relations together determine asymptotic expansions of A and B. We consider these two relations in turn.

First we examine (7). Substituting the asymptotic expansion for ϕ^s gives

$$\begin{aligned}(A+B)d(\kappa) &= \mathbf{S}_0^N D_+ D_- \phi^s \\ &\sim \mathbf{S}_0^N D_+ D_- \phi^{(0)} + \epsilon \mathbf{S}_0^N D_+ D_- \phi^{(1)} + \epsilon^2 \mathbf{S}_0^N D_+ D_- \phi^{(2)} + \ldots .\end{aligned}$$

We consider the first few terms of this asymptotic series. The first term is

$$\mathbf{S}_0^N D_+ D_- \phi^{(0)} = \frac{h}{2}\left(z_0^{(0)} + z_N^{(0)}\right) - \frac{h}{2}(F_0 + F_N) + \mathbf{S}_0^N F.$$

We prove below that $\mathbf{S}_0^N F = 0$ provided the compatibility conditions of F are satisfied and so from the forms of $z_0^{(0)}$ and $z_N^{(0)}$ in (4) it follows that

$$\mathbf{S}_0^N D_+ D_- \phi^{(0)} = O(h^7)$$

At first order

$$\begin{aligned}\mathbf{S}_0^N D_+ D_- \phi^{(1)} &= \frac{h}{2}\left(z_0^{(1)} + z_N^{(1)}\right) + \sum_{\nu=1}^{N-1} h D_+ D_- F_\nu \\ &\quad + \frac{1}{h}\left[(z_0^{(0)} - F_0) + (z_N^{(0)} - F_N)\right] \\ &= \frac{h}{2}\left(z_0^{(1)} + z_N^{(1)}\right) + D_- F_N - D_+ F_0 + O(h^5) \\ &= \frac{h}{2}\left(z_0^{(1)} + z_N^{(1)}\right) + \left[F'|_0^1 + \frac{h^2}{3!}F'''|_0^1 + \frac{h^4}{5!}F^{(5)}|_0^1 + \ldots\right] \\ &\quad - \frac{h}{2}\left[(F''(0) + F''(1)) + \frac{2h^2}{4!}(F^{(4)}(0) + F^{(4)}(1)) + \ldots\right] + O(h^5)\end{aligned}$$

The terms involving the odd derivatives vanish by the compatibility conditions on F. The terms involving the even derivatives cancel to fourth order in h by the forms of $z_0^{(1)}$ and $z_N^{(1)}$ in (4). Hence

$$\mathbf{S}_0^N D_+ D_- \phi^{(1)} = O(h^5)$$

At second order

$$\mathbf{S}_0^N D_+D_-\phi^{(2)} = \frac{h}{2}\left(z_0^{(2)} + z_N^{(2)}\right) + \frac{1}{h}\left[D_+D_-F_2 - 2D_+D_-F_1 + z_0^{(1)}\right] + \frac{1}{h}\left[D_+D_-F_{N-2} - 2D_+D_-F_{N-1} + z_N^{(1)}\right] + \frac{1}{h^3}\left[(z_0^{(0)} - F_0) + (z_N^{(2)} - F_N)\right] + \sum_{\nu=2}^{N-2} h(D_+D_-)^2F_\nu$$

We can reduce this expression by eliminating the sum

$$\frac{1}{h}\left[D_+D_-\left(F_2 - 2F_1\right) + D_+D_-\left(F_{N-2} - 2F_{N-1}\right)\right] + \sum_{\nu=2}^{N-2} h(D_+D_-)^2F_\nu = \frac{1}{h}\left(D_+D_-F_1 + D_+D_-F_{N-1}\right)$$

and then expand using Taylor series

$$\begin{aligned} D_+D_-F_1 + D_+D_-F_{N-1} &= \left[F_1'' + \frac{2}{4!}h^2F_1^{(4)} + \frac{2}{6!}h^4F_1^{(6)} + \ldots\right] \\ &\quad + \left[F_{N-1}'' + \frac{2}{4!}h^2F_{N-1}^{(4)} + \frac{2}{6!}h^4F_{N-1}^{(6)} + \ldots\right] \\ &= \left(F''(0) + hF'''(0) + \frac{h^2}{2}F^{(4)}(0) + \ldots\right) + \frac{2}{4!}h^2F^{(4)}(0) + \ldots \\ &\quad + \left(F''(1) - hF'''(1) + \frac{h^2}{2}F^{(4)}(1) + \ldots\right) + \frac{2}{4!}h^2F^{(4)}(1) + \ldots \\ &= \left(z_0^{(1)} + z_N^{(1)}\right) + \frac{h^2}{2}\left(z_0^{(2)} + z_N^{(2)}\right) + O(h^4) \end{aligned}$$

using the compatibility conditions on F as well as the forms (4) for the $z^{(m)}$'s. Hence

$$\mathbf{S}_0^N D_+D_-\phi^{(2)} = O(h^3)$$

We have shown that the first three terms of the asymptotic expansion (7) of $(A+B)d(\kappa)$ are $O(h^7)$, $O(\epsilon h^5)$ and $O(\epsilon^2 h^3)$ respectively. This expansion can be carried to higher orders with the result that

$$A + B = 0$$

to all orders.

Now we consider the relation (8) which determines A. Substituting the asymptotic expansion for ϕ^s gives

$$\begin{aligned} Ad(\kappa) &= \Delta(D_+D_-\phi^s) \\ &\sim \Delta(D_+D_-\phi^{(0)}) + \epsilon\Delta(D_+D_-\phi^{(1)}) + \epsilon^2\Delta(D_+D_-\phi^{(2)}) + \dots \end{aligned}$$

To determine the $\Delta(D_+D_-\phi^{(m)})$ terms we need to use the double integral compatibility conditions on F.

We begin with the first term

$$\Delta(D_+D_-\phi^{(0)}) = \Delta(F)$$

We use Taylor series to relate the sum of F to the double integral of F and the compatibility conditions on F to reduce the expression which results. This leads to the following result.

Lemma 1. *If F satisfies the compatibility conditions then*

$$\Delta(D_+D_-\phi^{(0)}) = -\frac{h^2}{3!}F'(0) + O(h^4)$$

Proof. By the compatibility conditions

$$-\Delta(D_+D_-\phi^{(0)}) = \int_0^1 \int_0^x F(x')dx'dx - \Delta(F)$$

We break up the integration domain $\{(x,x') : 0 < x < 1,\ 0 < x' < x\}$ into pieces by first making slices along the lines $x = x_\nu$ and then along the lines $x' = x_\mu + \frac{h}{2}$

$$\begin{aligned} -\Delta(D_+D_-\phi^{(0)}) &= \sum_{\nu=0}^{N-1} \int_{x_\nu}^{x_{\nu+1}} \left[\int_0^x F(x')dx' - \sum_{\mu=0}^{\nu} hF_\nu \right] \\ &= \int_0^h \int_0^x (F(x') - F_0)dx'dx \\ &\quad + \sum_{\nu=1}^{N-1} \int_{x_\nu}^{x_{\nu+1}} \left\{ \int_0^{h/2} (F(x') - F_0)dx' + \sum_{\mu=1}^{\nu-1} \int_{x_\mu - \frac{h}{2}}^{x_\mu + \frac{h}{2}} (F(x') - F_\mu)dx' \right. \\ &\qquad \left. + \int_{x_\nu - \frac{h}{2}}^{x} (F(x') - F_\nu)dx' \right\} dx \end{aligned}$$

Next we make Taylor series expansions about $x' = x_\mu$ and change variables to $y = x - x_\mu$

$$-\Delta(D_+D_-\phi^{(0)}) = \sum_{m=1}^{\infty}\frac{1}{m!}\int_0^h\left\{F_0^{(m)}\int_0^x y^m dy + \sum_{\nu=1}^{N-1}\left(F_0^{(m)}\int_0^{h/2} y^m dy + \sum_{\mu=1}^{\nu-1}F_\mu^{(m)}\int_{-h/2}^{h/2} y^m dy + F_\nu^{(m)}\int_{-h/2}^{x} y^m dy\right)\right\}dx$$

Evaluating the integrals gives

$$-\Delta(D_+D_-\phi^{(0)}) = \sum_{m=1}^{\infty}\frac{h^{m+2}}{(m+2)!}\sum_{\nu=0}^{N-1}F_\nu^{(m)} + \sum_{m=1}^{\infty}\frac{(h/2)^{m+1}}{(m+1)!}\sum_{\nu=1}^{N-1}\left\{hF_0^{(m)} + \sum_{\mu=1}^{\nu-1}[1-(-1)^{m+1}]hF_\mu^{(m)} - (-1)^{m+1}hF_\nu^{(m)}\right\}$$

Replacing the inner summations by trapezoidal sums we obtain

$$\begin{aligned}-\Delta(D_+D_-\phi^{(0)}) &= \sum_{m=1}^{\infty}\frac{h^{m+1}}{(m+2)!}\left(\mathbf{S}_0^N F^{(m)} - \frac{h}{2}F^{(m)}|_0^1\right)\\ &\quad + \sum_{\substack{m=1\\(m\text{ odd})}}^{\infty}\frac{(h/2)^{m+1}}{(m+1)!}\sum_{\nu=1}^{N-1}h(F_0^{(m)} - F_\nu^{(m)})\\ &\quad + \sum_{\substack{m=1\\(m\text{ even})}}^{\infty}\frac{(h/2)^{m}}{(m+1)!}\sum_{\nu=1}^{N-1}h\mathbf{S}_0^\nu F^{(m)}\\ &= <1> + <2> + <3>\end{aligned}$$

$<2>$ can be further simplified by replacing the sum over ν by boundary terms

$$\sum_{\nu=1}^{N-1}h(F_0^{(m)} - F_\nu^{(m)}) = F^{(m)}(0) - \mathbf{S}_0^N F^{(m)} + \frac{h}{2}F^{(m)}|_0^1$$

which is $O(h^2)$ by the compatibility conditions as shown in the appendix to this section. Similarly $<3>$ can be simplified by

$$\begin{aligned}\sum_{\nu=1}^{N-1}h\mathbf{S}_0^\nu F^{(m)} &= \sum_{\nu=1}^{N-1}h\left[\frac{h}{2}F_0^{(m)} + \sum_{\mu=1}^{\nu}hF_\mu^{(m)} - \frac{h}{2}F_\nu^{(m)}\right]\\ &= \sum_{\nu=0}^{N-1}h\left[\frac{h}{2}F_0^{(m)} + \sum_{\mu=1}^{\nu}hF_\mu^{(m)}\right] - \frac{h}{2}\sum_{\nu=0}^{N-1}hF_\nu^{(m)}\\ &= \Delta F^{(m)} - \frac{h}{2}\mathbf{S}_0^N F^{(m)} + \frac{h^2}{2}F^{(m)}|_0^1\\ &= \Delta F^{(m)} + \frac{h^2}{2}F^{(m+1)}(0)\end{aligned}$$

Hence

$$-\Delta(D_+D_-\phi^{(0)}) = \sum_{\substack{m=1 \\ (m \text{ odd})}}^{\infty} \left\{ \frac{h^{m+1}}{(m+2)!}\mathbf{S}_0^N F^{(m)} + \frac{(h/2)^{m+1}}{(m+1)!}(F^{(m)}(0) - \mathbf{S}_0^N F^{(m)}) \right\}$$

$$+ \sum_{\substack{m=1 \\ (m \text{ even})}}^{\infty} \left\{ \frac{h^{m+1}}{(m+2)!}(-\frac{h}{2}F^{(m+1)}(0)) + \frac{(h/2)^m}{(m+1)!}(\Delta F^{(m)} + \frac{h^2}{2}F^{(m+1)}(0)) \right\}$$

$$= \frac{h^2}{3!}F'(0) + \frac{(h/2)^2}{3!}\Delta(F'') + O(h^4)$$

But F'' satisfies all the same compatibility conditions as F and so

$$-\Delta(D_+D_-\phi^{(0)}) = \frac{h^2}{3!}F'(0) + O(h^4)$$

which proves the lemma.

For completeness we go to first order in ϵ in the expansion for $Ad(\kappa)$ by calculating $\Delta(D_+D_-\phi^{(1)})$.

$$\Delta(D_+D_-\phi^{(1)}) = \sum_{\nu=0}^{N-1} h \left\{ \frac{h}{2}z_0^{(1)} + \sum_{\mu=1}^{\nu} h(D_+D_-)^2\phi_\mu^{(0)} \right\}$$

$$= \sum_{\nu=0}^{N-1} h \left\{ \frac{h}{2}z_0^{(1)} + \sum_{\mu=1}^{\nu} hD_+D_-F_\mu \right\}$$

$$= \frac{h}{2}z_0^{(1)} + \sum_{\nu=0}^{N-1} h\{D_+F_\nu - D_+F_0\}$$

$$= \frac{h}{2}z_0^{(1)} + F_N - F_0 - D_+F_0$$

$$= \frac{h}{2}z_0^{(1)} + F'(0) - D_+F_0$$

$$= \frac{h}{2}\left\{ F''(0) + \frac{2}{4!}h^2F^{(4)}(0) + \frac{2}{6!}h^4F^{(6)}(0) + \ldots \right\}$$

$$- \left\{ \frac{h}{2!}F''(0) + \frac{h^2}{3!}F^{(3)}(0) + \frac{h^3}{4!}h^2F^{(4)}(0) + \ldots \right\}$$

$$= -\frac{h^2}{3!}F^{(3)}(0) - \frac{h^4}{5!}F^{(5)}(0) - \ldots$$

We have found the first two terms of the asymptotic expansions for the coefficients A and B

$$Ad(\kappa) \sim -\frac{h^2}{3!}F'(0) + O(h^4, \epsilon h^2)$$

$$Bd(\kappa) \sim \frac{h^2}{3!}F'(1) + O(h^4, \epsilon h^2)$$

Now we consider the modifications to determine the coefficients A and B when higher order boundary conditions are used. In this case (7) and (8) involve extra terms; they can be written as

$$(A+B)d(\kappa) = \mathbf{S}_0^N D_+ D_- \phi^s + H_1 \tag{7'}$$

and

$$Ad(\kappa) = \Delta(D_+ D_- \phi^s) + H_2 \tag{8'}$$

H_1 and H_2 are determined from the equations

$$\begin{aligned} 0 &= D_{r,q}\phi_N - D_{l,q}\phi_0 \\ &= (D_0 - D_{l,q})\,\phi_0^s - (D_0 - D_{r,q})\,\phi_N^s \\ &\quad + (D_0\phi_N^s - D_0\phi_0^s) + (D_{r,q}\phi_N^b - D_{l,q}\phi_0^b) \\ &= H_1 + \mathbf{S}_0^N D_+ D_- \phi^s - (A+B)d(\kappa) \end{aligned}$$

and

$$\begin{aligned} 0 = \phi_N^s - \phi_0^s &= \sum_{\nu=0}^{N-1} hD_+\phi_\nu^s \\ &= \cdots = D_0\phi_0^s + \Delta(D_+ D_- \phi^s) \\ &= (D_0 - D_{l,q})\,\phi_0^s + D_{l,q}\phi_0^s + \Delta(D_+ D_- \phi^s) \\ &= (D_0 - D_{l,q})\,\phi_0^s - D_{l,q}\phi_0^b + \Delta(D_+ D_- \phi^s) \\ &= H_2 - Ad(\kappa) + \Delta(D_+ D_- \phi^s) \end{aligned}$$

Hence

$$\begin{aligned} H_1 &= (D_0 - D_{l,q})\,\phi_0^s - (D_0 - D_{r,q})\,\phi_N^s \\ H_2 &= (D_0 - D_{l,q})\,\phi_0^s \end{aligned}$$

From the properties of ϕ^s which were found above it is easily deduced that

$$(D_0 - D_{l,q})\,\phi_0^s = \begin{cases} 0 & \text{if } q = 2 \\ \frac{h^2}{3!}F'(0) + O(h^3, \epsilon h^2) & \text{if } q = 3 \\ \frac{h^2}{3!}F'(0) + O(h^4, \epsilon h^2) & \text{if } q = 4 \end{cases}$$

and likewise for $(D_{r,q} - D_0)\,\phi_N^s$. Hence for $q \geq 3$ it follows that $H_1 = O(h^3)$ and

$$H_2 + \Delta(D_+ D_- \phi^s) = O(h^3)$$

That is, both $Ad(\kappa)$ and $Bd(\kappa)$ are $O(h^3)$, which is the final result of Proposition 1.

Appendix to Section 5.2.5

Lemma 1. *If F satisfies the compatibility conditions*

$$\int_0^1 F^{(2m)}(x)dx = 0 \qquad m = 0, 1, 2, \ldots$$

then

$$\mathbf{S}_0^N F := \frac{h}{2}F_0 + \sum_{\nu=0}^{N-1} hF_\nu + \frac{h}{2}F_N = O(h^{2q})$$

for all positive integers q.

Proof. The proof is by induction.

$$\begin{aligned}
\int_0^1 F(x)dx - \mathbf{S}_0^N F &= \left\{ \int_0^{h/2} + \sum_{\nu=1}^{N-1} \int_{x_\nu - \frac{h}{2}}^{x_\nu + \frac{h}{2}} + \int_{1-\frac{h}{2}}^1 \right\} (F(x) - F_\nu)dx \\
&= \{\ldots\} \sum_{m=1}^{\infty} \frac{1}{m!} F_\nu^{(m)} (x - x_\nu)^m dx \\
&= \sum_{m=1}^{\infty} \frac{1}{m!} \frac{(h/2)^{m+1}}{m+1} \left\{ F_0^{(m)} \right. \\
&\qquad \left. + \sum_{\nu=1}^{N-1} [1 - (-1)^{m+1}] F_\nu^{(m)} - (-1)^{m+1} F_N^{(m)} \right\} \\
&= \sum_{\substack{m=1 \\ (m \text{ odd})}}^{\infty} \frac{(h/2)^{m+1}}{(m+1)!} [-F^{(m)}]\Big|_0^1 \\
&\quad + \sum_{\substack{m=1 \\ (m \text{ even})}}^{\infty} \frac{(h/2)^m}{(m+1)!} \mathbf{S}_0^N F^{(m)} \\
&= \sum_{n=1}^{\infty} \frac{(h/2)^{2n}}{(2n+1)!} \mathbf{S}_0^N F^{(2n)}
\end{aligned}$$

Therefore

$$\mathbf{S}_0^N F = \sum_{n=1}^{\infty} a_n^{(1)} h^{2n} \mathbf{S}_0^N F^{(2n)}$$

where

$$a_n^{(1)} = -\frac{1}{2^{2n}(2n+1)!} \mathbf{S}_0^N F^{(2n)} \qquad n = 1, 2, \ldots$$

Now F'' satisfies the same compatibility conditions as F and so

$$\mathbf{S}_0^N F'' = \sum_{n=1}^{\infty} a_n^{(1)} h^{2n} \mathbf{S}_0^N F^{(2n+2)}$$

Hence

$$\mathbf{S}_0^N F = \sum_{n=2}^{\infty} a_n^{(2)} h^{2n} \mathbf{S}_0^N F^{(2n)}$$

where

$$a_n^{(2)} = a_n^{(1)} + a_1^{(1)} a_{n-1}^{(1)} \qquad n = 2, 3, \ldots$$

This process can be continued to any order in h with the result that

$$\mathbf{S}_0^N F = O(h^{2q})$$

for any positive integer q.

Lemma 2. *If F satisfies the compatibility conditions*

$$\int_0^1 F^{(2m+1)}(x)dx = F^{(2m+1)}(0) \qquad m = 0, 1, 2, \ldots$$

then

$$\mathbf{S}_0^N F^{(q)} = F^{(q)}(0) + O(h^2)$$

for any odd positive integer q.

Proof.

$$\int_0^1 F^{(q)}(x)dx - \mathbf{S}_0^N F^{(q)} = \sum_{\substack{m=1 \\ (m \text{ odd})}}^{\infty} \frac{(h/2)^{m+1}}{(m+1)!} [-F^{(q+m)}]\Big|_0^1 + \sum_{\substack{m=1 \\ (m \text{ even})}}^{\infty} \frac{(h/2)^m}{(m+1)!} \mathbf{S}_0^N F^{(q+m)}$$

Therefore

$$\begin{aligned}
\mathbf{S}_0^N F^{(q)} &= F^{(q)}(0) + \sum_{\substack{m=1 \\ (m \text{ odd})}}^{\infty} \frac{(h/2)^{m+1}}{(m+1)!} F^{(q+m+1)}(0) \\
&\quad - \sum_{\substack{m=1 \\ (m \text{ even})}}^{\infty} \frac{(h/2)^m}{(m+1)!} \mathbf{S}_0^N F^{(q+m)} \\
&= F^{(q)}(0) + \frac{1}{2!}\left(\frac{h}{2}\right)^2 \left[F^{(q+2)}(0) - \frac{1}{3}\mathbf{S}_0^N F^{(q+2)}\right] \\
&\quad + \frac{1}{4!}\left(\frac{h}{2}\right)^4 \left[F^{(q+4)}(0) - \frac{1}{5}\mathbf{S}_0^N F^{(q+4)}\right] + \ldots \\
&= F^{(q)}(0) + \sum_{\substack{m=1 \\ (m \text{ even})}}^{\infty} \frac{1}{m!}\left(\frac{h}{2}\right)^m \left[F^{(q+m)}(0) - \frac{1}{m+1}\mathbf{S}_0^N F^{(q+m)}\right]
\end{aligned}$$

The result follows.

5.2.6 Numerical Examples

In this section we look at some computational results to see how well our analysis predicts the observed numerical behaviour. Solutions are computed to the discrete single time step model when the forcing has been chosen so that an exact solution to the corresponding continous model is easily found. The errors are compared to the errors which are estimated from the results of previous sections. We consider two examples. In the first example the solution is of a form where there are no boundary layers in the error. The computed stream function and vorticity are then accurate to second order. In the second example, however, the error has boundary layers. In the two limits, $\epsilon \ll h^2$ and $\epsilon \gg h^2$, the errors are seen to be of the appropriate form. Accuracy is lost when the boundary conditions are only second order. When higher order boundary conditions are used the computed vorticity is seen to be second order.

Example 1

When the forcing to the continuous single time step model is chosen to be

$$F = 4\pi^2[1 - 4\pi^2\epsilon]\cos 2\pi x$$

The solution is given by

$$\psi = 1 - \cos 2\pi x.$$

We solved the discrete single time step model problem (5.2.1.2) with the forcing F_ν equal to $F(x_\nu)$. Second order boundary conditions were used. The error has no boundary layer component in this case. The terms which force the boundary layers in the error, g_0 and g_1, are both zero since all odd derivatives of ψ are zero at the boundaries (see section 5.2.3). The asymptotic expansions for the continuous and the discrete vorticity are

$$\varsigma \sim F + \epsilon F_{xx} + \ldots$$

$$z \sim F + \epsilon D_+ D_- F + \ldots.$$

Thus the normalized error in the vorticity $e_\varsigma^h := (\varsigma - z)/h^2$ is

$$\begin{aligned} e_\varsigma^h &\sim -\frac{\epsilon}{h^2}\left(D_+D_- - \frac{\partial^2}{\partial x^2}\right)F + O(\epsilon^2) \\ &= -\epsilon\frac{1}{12}F_{xxxx} + O(\epsilon^2, \epsilon h^2) \\ &= -\epsilon\frac{\pi^2}{3}(2\pi)^4 \cos 2\pi x + O(\epsilon^2, \epsilon h^2) \\ &\approx -\epsilon(5.127 \times 10^3)\cos 2\pi x + O(\epsilon^2, \epsilon h^2) \end{aligned}$$

The normalized error in the stream function $e_\psi^h := (\psi - \phi)/h^2$ is given by

$$\begin{aligned} e_\psi^h &\sim \int_0^x \int_0^{x'} G dx'' dx' + \epsilon G + \ldots \\ &= -\frac{(2\pi)^2}{12}(1 - cos 2\pi x) + O(\epsilon, h^2) \end{aligned}$$

Whence the leading order terms for the maximum norms of the errors are

$$||e_\varsigma^h||_\infty \sim 5.127 \times 10^3 \epsilon$$

$$||e_\psi^h||_\infty \sim 6.58.$$

The errors and their norms were calculated for $\epsilon = 10^{-4}$. Table 5.2 gives the results for some different values of h.

h	$\|e_\psi^h\|_\infty$	$\|e_\varsigma^h\|_\infty/\epsilon$
1/10	6.74	5.04×10^3
1/20	6.64	5.09×10^3
1/40	6.61	5.10×10^3
1/80	6.61	5.11×10^3

Table 5.2 Normalized Errors for Example 1, $\epsilon = 10^{-4}$

It is seen that the errors behave has expected. Both the stream function and vorticity are computed to second order. In fact, the error the vorticity is $O(\epsilon h^2)$.

Example 2

As a second example we consider the forcing function

$$F = -\omega^2 \sin\left[\omega(x - \frac{1}{2})\right]$$

where

$$\omega = 2\tan\frac{\omega}{2}, \qquad 2\pi < \omega < 3\pi$$

$$\omega \approx 8.9868189158181$$

The exact solution is

$$\psi = \sin\left[\omega(x - \frac{1}{2})\right] + (1 - 2x)\sin\frac{\omega}{2}$$

The asymptotic expansions for the continuous and the discrete vorticity are

$$\varsigma \sim F + \epsilon F_{xx} + \ldots$$

$$z \sim F + \epsilon D_+ D_- F + \ldots + \left(A\kappa^{-\nu} + B\kappa^{\nu-N}\right)\frac{(\kappa-1)^2}{h^2\kappa}$$

where

$$A \sim -g_0 h^2/d(\kappa), \qquad B \sim g_1 h^2/d(\kappa)$$

with

$$g_0 \sim \frac{1}{3!}\psi_{xxx}(0) \quad \approx 26.3$$

$$g_1 \sim \frac{1}{3!}\psi_{xxx}(1) \quad \approx 26.3.$$

h	$\|e_\psi^h\|_\infty$	$\|e_\varsigma^h\|_\infty$	$h\|e_\varsigma^h\|_\infty$	$h/\sqrt{\epsilon}$
1/10	9.58	539	53.9	31.6
1/20	9.29	1050	52.6	15.8
1/40	9.22	2050	51.2	7.9
1/80	9.21	3760	47.0	3.95
1/100	9.22	4460	44.6	3.16

Table 5.3 Errors for $\epsilon = 10^{-5}$

From these expressions we can write down the normalized error in the vorticity

$$\begin{aligned} e_\varsigma^h &\sim -\epsilon\frac{1}{12}F_{xxxx} + (-g_0\kappa^{-\nu} + g_1\kappa^{\nu-N})\bar{d}(\kappa) + O(\epsilon, h^2) \\ &\approx 4.4\times 10^4\epsilon\sin\omega(x-\frac{1}{2}) - 26.3\bar{d}(\kappa)(\kappa^{-\nu}+\kappa^{\nu-N}) + O(\epsilon, h^2) \end{aligned}$$

For second order boundary conditions the function $\bar{d}(\kappa)$ is given by

$$\bar{d}(\kappa) := \frac{1}{d(\kappa)}\frac{(\kappa-1)^2}{h^2\kappa} \sim \begin{cases} 1/\sqrt{\epsilon} & h \ll \sqrt{\epsilon} \ll 1 \\ 2/h & \sqrt{\epsilon} \ll h \ll 1 \end{cases}$$

The normalized error in the stream function in this example is given by

$$\begin{aligned} e_\psi^h &\sim \int_0^x \int_0^{x'} G dx'' dx' + \epsilon G + \ldots \\ &= -\frac{\omega^2}{12}\psi(x) + O(\epsilon, h^2) \\ &\approx -6.7\psi(x) \end{aligned}$$

Hence, the leading order terms for the maximum norms of the errors are

$$\|e_\varsigma^h\|_\infty \sim \max\left\{4.4\times 10^4\epsilon,\ 26.3\bar{d}(\kappa)\right\}$$

$$\|e_\psi^h\|_\infty \sim 6.7\|\psi\|_\infty \sim 9.18$$

In the two limits

$$26.3\bar{d}(\kappa) = \begin{cases} 26.3/\sqrt{\epsilon} & \text{for } h \ll \sqrt{\epsilon} \\ 52.6/h & \text{for } \sqrt{\epsilon} \ll h \end{cases}$$

When $h \gg \sqrt{\epsilon}$ the above results suggest that the computed vorticity should be only first order accurate with $h\|e_\varsigma^h\|_\infty \approx 52.6$. Table 5.3 shows the results for different values of h when $\epsilon = 10^{-5}$.

h	$\|\|e_\psi^h\|\|_\infty$	$\|\|e_\varsigma^h\|\|_\infty$	$\sqrt{\epsilon}\|\|e_\varsigma^h\|\|_\infty$	$h/\sqrt{\epsilon}$
1/10	150	2540	25.4	1
1/20	146	2530	25.3	.5
1/40	145	2550	25.5	.25
1/80	145	2550	25.5	.125
1/100	145	2560	25.6	.1

Table 5.4 Errors for $\epsilon = 10^{-2}$

In the other limit, $h \ll \sqrt{\epsilon}$, the predicted behaviour of the error in the vorticity is $\sqrt{\epsilon}\|e_\varsigma^h\|_\infty \approx 26.3$. Results in this limit are shown in table 5.4.

Now we show the behaviour of the error as $h/\sqrt{\epsilon}$ varies from large to small. The error shows a continuous transition between the two limiting cases.

h	$\|\|e_\psi^h\|\|_\infty$	$\|\|e_\varsigma^h\|\|_\infty$	$\sqrt{\epsilon}\|\|e_\varsigma^h\|\|_\infty$	$h\|\|e_\varsigma^h\|\|_\infty$	$h/\sqrt{\epsilon}$
1/10	9.67	528	5.28	52.8	10
1/20	9.39	986	9.86	49.2	5
1/40	9.35	1660	16.6	41.6	2.5
1/80	9.36	2260	22.6	28.3	1.25
1/100	9.36	2390	23.9	23.9	1.

Table 5.5 Errors for $\epsilon = 10^{-4}$

Higher Order Boundary Conditions

The boundary part of the vorticity error when higher order boundary conditions are used is given asymptotically by

$$\|e_{\varsigma_b}^h\|_\infty \sim |g\,\bar{d}(\kappa)|$$

Recall that $\bar{d}(\kappa)$ depends on the approximation to the boundary condition. In table 5.6 we show the expected form of the boundary part of the error.

Order of b.c.	g	$\|\|e^h_{\varsigma_b}\|\|_\infty$ $h \ll \sqrt{\epsilon}$	$\sqrt{\epsilon} \ll h$
2	$(h^2/3!)\psi_{xxx}(0) \sim 26.3h^2$	$26.3/\sqrt{\epsilon}$	$52.6/h$
3	$-(2h^3/4!)\psi^{(4)}(0) \sim 530h^3$	$530h/\sqrt{\epsilon}$	1600
4	$6(h^4/5!)\psi^{(5)}(0) \sim 637h^4$	$637h^2/\sqrt{\epsilon}$	$2600h$

Table 5.6 Form of Boundary Error for Higher Order B.C.'s

h	$\|\|e^h_\psi\|\|_\infty$	$\|\|e^h_\varsigma\|\|_\infty$	$\sqrt{\epsilon}\|\|e^h_\varsigma\|\|_\infty/h$	$h/\sqrt{\epsilon}$
1/10	9.59	1330	133	10
1/20	9.30	1340	268	5
1/40	9.24	1070	428	2.5
1/80	9.24	651	520	1.25
1/100	9.24	533	533	1.

Table 5.7 Errors for Third Order B.C.'s and $\epsilon = 10^{-4}$

h	$\|\|e^h_\psi\|\|_\infty$	$\|\|e^h_\varsigma\|\|_\infty$	$\|\|e^h_\varsigma\|\|_\infty/h$	$\sqrt{\epsilon}\|\|e^h_\varsigma\|\|_\infty/h^2$	$h/\sqrt{\epsilon}$
1/10	9.61	894	8940	894	10
1/20	9.34	278	5560	1110	5
1/40	9.28	72.7	2910	1160	2.5
1/80	9.26	18.8	1500	1200	1.25
1/100	9.26	13.1	1310	1310	1.

Table 5.8 Errors for Fourth Order B.C.'s and $\epsilon = 10^{-4}$

In the next two tables we give the errors for third and fourth order boundary conditions, respectively.

The vorticity looks to be even more accurate than second order. However, other parts of the (normalized) error of order ϵ. If a larger epsilon is taken the vorticity shows that it is second order. This is illustrated in the last table of this section. (When the boundary errors are small we expect $\|e^h_\varsigma\|$ to be $\approx 4.4 \times 10^4 \epsilon$.)

h	$\|e_\psi^h\|_\infty$	$\|e_\zeta^h\|_\infty$
1/10	167	660
1/20	173	505
1/40	174	483
1/80	174	484
1/100	174	483

Table 5.9 Errors for Fourth Order B.C.'s and $\epsilon = 10^{-1}$

Chapter 6

Numerical Examples

6.1 Multigrid

In this section we illustrate the solution of Poisson's equation on composite overlapping meshes using multigrid.

Multigrid Example 1

As a first example the region Ω is a circle which has been covered by a composite mesh consisting of two component meshes. Figure 6.1 shows this mesh along with the coarser meshes which are used by the multigrid algorithm. Boundary points on the inner component mesh are marked with small circles. Biquadratic interpolation is used at the interpolation boundaries of the component meshes. The composite smoother for the curvilinear grid consists of an alternating line SOR (*Line Zebra*) on lines in the radial direction. A Red-Black Jacobi smoother is used on the inner rectangular grid. A composite smooth of type S is employed. For the rectangular grid the restriction operator is taken as the *half weighting*.

$$\mathbf{R}^{1\to 2} = \frac{1}{8}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

A full weighting is used on the curvilinear grid. The mesh equations are solved directly on the coarsest grid. The interpolation equations are coupled at the lower multigrid levels and it is necessary to solve a small system of equations to update the interpolation points.

Tables 6.1 and 6.2 summarize the results when the number of multigrid levels is 2 and 3, respectively. The quantities appearing in the table are now explained. $r(k)$ is the residual on the finest grid after the k^{th} multigrid iteration and defined

by

$$r(k) = \|\mathbf{f}^1 - \mathbf{A}^1\mathbf{v}^1(k)\|$$

$WU(k)$ is the total number of Work Units used up to and including the k^{th} iteration. A work unit is the amount of work (number of multiplications say) to perform one iteration of SOR on the composite mesh. The effective convergence rate ECR is defined as

$$ECR(k) = \left(\frac{r(k)}{r(k-1)}\right)^{\rho}$$

$$\rho = \frac{1}{(WU(k) - WU(k-1))}$$

This effective convergence rate is useful to use when comparing the multigrid methods to other methods. It essentially indicates the convergence rate that an SOR type iteration would have to achieve in order to be doing as well as multigrid. We usually obtain effective convergence rates in the range .6 to .7. For comparison the (effective) convergence rate for optimum SOR on a unit square with $N \times N$ points, $N = 50$, is $(1 - sin(\pi/N)/(1 + \sin(\pi/N)) \approx .87$. The convergence rate for Gauss Seidel is $\cos^2(\pi/N) \approx .996$.

Iteration	r(k)	r(k)/r(k-1)	WU(k)	ECR(k)
k=2	.77e-3	.050	10.7	.48
k=3	.71e-4	.093	14.8	.56
k=4	.12e-4	.17	19.0	.65
k=5	.20e-5	.17	23.1	.65

Table 6.1 Convergence Rates for 2 Levels

Iteration	r(k)	r(k)/r(k-1)	WU(k)	ECR(k)
k=2	.99e-2	.14	11.5	.65
k=3	.14e-2	.14	16.0	.65
k=4	.20e-3	.14	20.6	.65
k=5	.29e-4	.14	25.1	.65

Table 6.2 Convergence Rates for 3 Levels

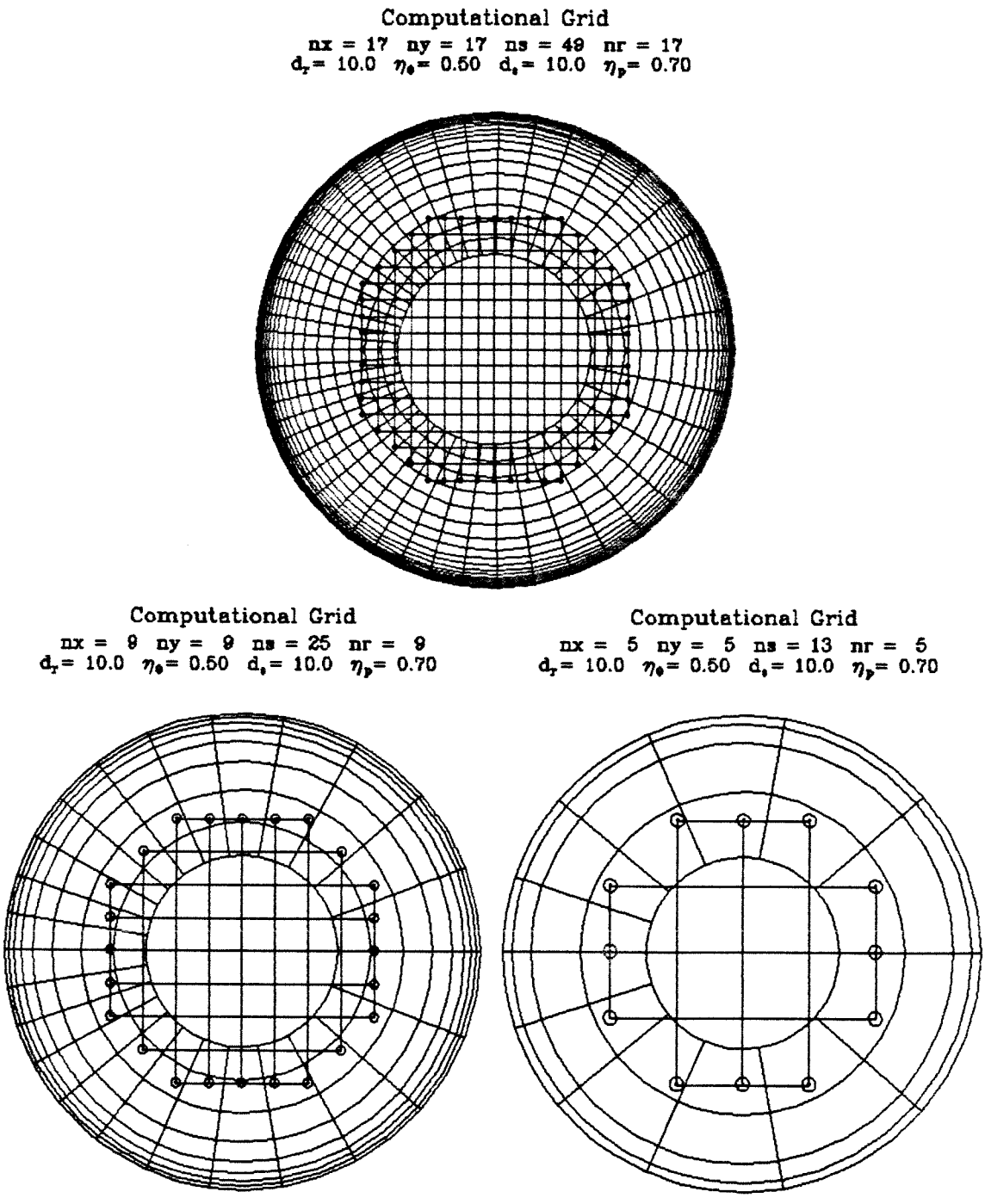


Figure 6.1 Composite Mesh for Multigrid Example 1

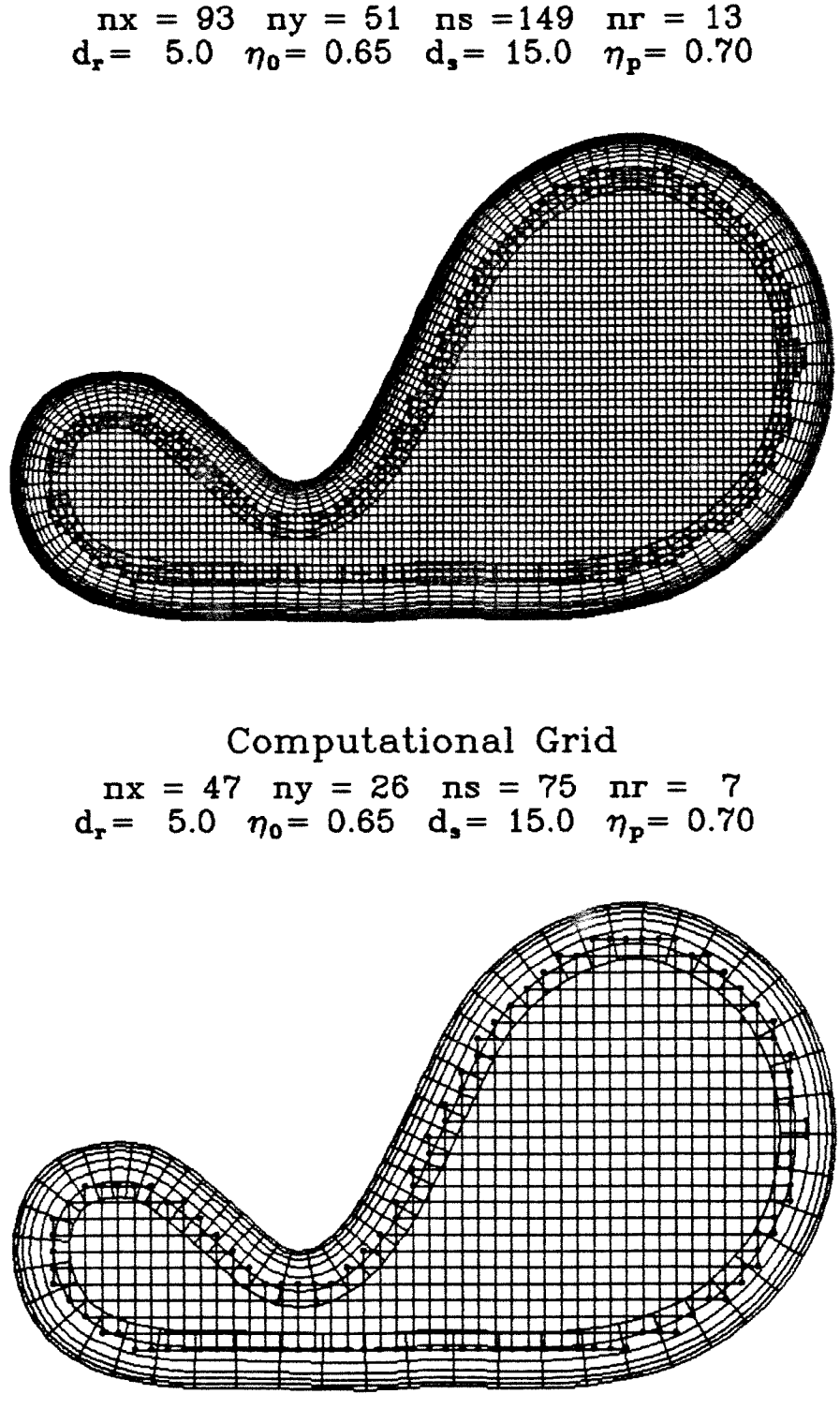


Figure 6.2 Composite Mesh for Multigrid Example 2

Multigrid Example 2

As a second example Poisson's equation is solved on the composite mesh shown in figure 6.2. This region is used for the *big ocean* run presented in the final section in this chapter. The actual grid shown here, however, is finer by about a factor of four than the one that is used in that section. The effective convergence rates for this large problem, presented in table 6.3, are seen to be just as good as those of the previous example. The total amount of work to solve this problem is of course more since there are a greater number of points on this mesh.

Iteration	r(k)	r(k)/r(k-1)	WU(k)	ECR(k)
k= 2	.34e-2	.11	10.9	.59
k= 3	.48e-3	.14	15.1	.63
k= 4	.64e-4	.13	19.3	.62
k= 5	.11e-4	.17	23.5	.66

Table 6.3 Convergence Rates for 2 Levels

Timings

Next we present some results which give an idea of the speed of the multigrid method. When the problem is small enough a direct solver can be used. The time for multigrid is compared with the time required to solve the same composite mesh equations using the direct solver (not including the factorization). The program has been run on a Cray-1 at the National Center for Atmospheric Research in Boulder Colorado, and on the Fluid Dynamics Vax 11/750 at Caltech. At the time the runs on the Cray were made the mesh equations at the lowest level were not solved exactly. Instead an SOR iteration was used. Experience shows that some saving in time is gained by using the direct solver.

The number of grid points indicated is an approximate number only. The multigrid programs are relatively efficient but they were not written to be as fast

Run	Machine	Number of Grid Points	Multigrid		Direct
			Per Cycle	Per WU	
1	Cray-1	2300	.03	.005	.073
2	Cray-1	4200	.04	.007	???
3	Vax 750	1100	3.8	1.1	.94
4	Vax 750	4300	9.2	2.1	???

Table 6.4 CPU times in seconds

as possible. The times shown should thus be considered with this in mind. Timings for multigrid are given in two ways. The first is the time for an average multigrid cycle. Dividing this value by the number of work units required by the cycle gives a time per work unit. Although times for a cycle and number of work units required per cycle may vary, the time per work unit for a given grid is fairly constant. The effective convergence rates for the runs in table 6.4 were all around .6 to .7. From the results of run 1 on the Cray one sees that approximately two multigrid iterations take the same time as the direct solver. Hence, if one starts with a good guess, two multigrid iterations may achieve the desired accuracy and the multigrid iteration could be just as fast as the direct solver. For larger problems multigrid becomes more efficient relative to the direct solver.

6.2 Comparison with a Rectangular Model

In this section we look at the solution of the vorticity stream function (*ocean*) equations on a circular basin. For comparison the same equations are solved on a mesh which consists of a subset of a uniform rectangular grid. This alternate method will be referred to as the *rectangular model*. The four different grids used in this comparison are shown in figure 6.3. There are two composite overlapping meshes and two rectangular grids. For reference the grids are labelled 1 through 4. Some of the points to consider when examining the results from this comparison are

(1) the accuracy of the composite overlapping mesh method versus the rect-

angular model.

(2) the effect of boundary fitted coordinates and the effect of stretching the grid to resolve boundary layers.

(3) the relative speeds of the methods.

We now proceed to briefly describe the rectangular model.

Rectangular Model

A disk or radius 1 is approximated by a subset of a rectangular grid. Points on the grid are given by

$$\mathbf{RG} = \{ (x_i, y_j) \,|\, (i,j) \in N_R \}$$

$$x_i = -1 + (i-1)h \qquad y_j = -1 + (j-1)h$$

$$N_R = \{ (i,j) \quad | \quad \sqrt{x_i^2 + y_j^2} \leq 1 \}$$

The vorticity stream function equations were discretized in space using central differences and in time using leap frog, in the same fashion as outlined in chapter 3. The Poisson equation for ψ was solved using a sparse matrix solver. Multigrid was used for a few runs and gave the same results. For problems of this size, however, the direct solver is faster. The no slip boundary conditions $\psi = 0$ and $\partial\psi/\partial n = 0$ are approximated by setting the discrete stream function to zero on the boundary of **RG** and on the points outside this boundary. The boundary of **RG** was chosen so that on average the circle of radius 1 lies midway between the boundary of **RG** and the next line out.

Accuracy Tests

As a test of the accuracy of the codes the equations were solved with a forcing that makes the following stream function and vorticity satisfy the equations exactly.

$$\psi^T(x,t) = \left(x^2 + y^2 - 1\right)^2 \sin(2\pi t)$$

$$\varsigma^T = \nabla^2 \psi^T$$

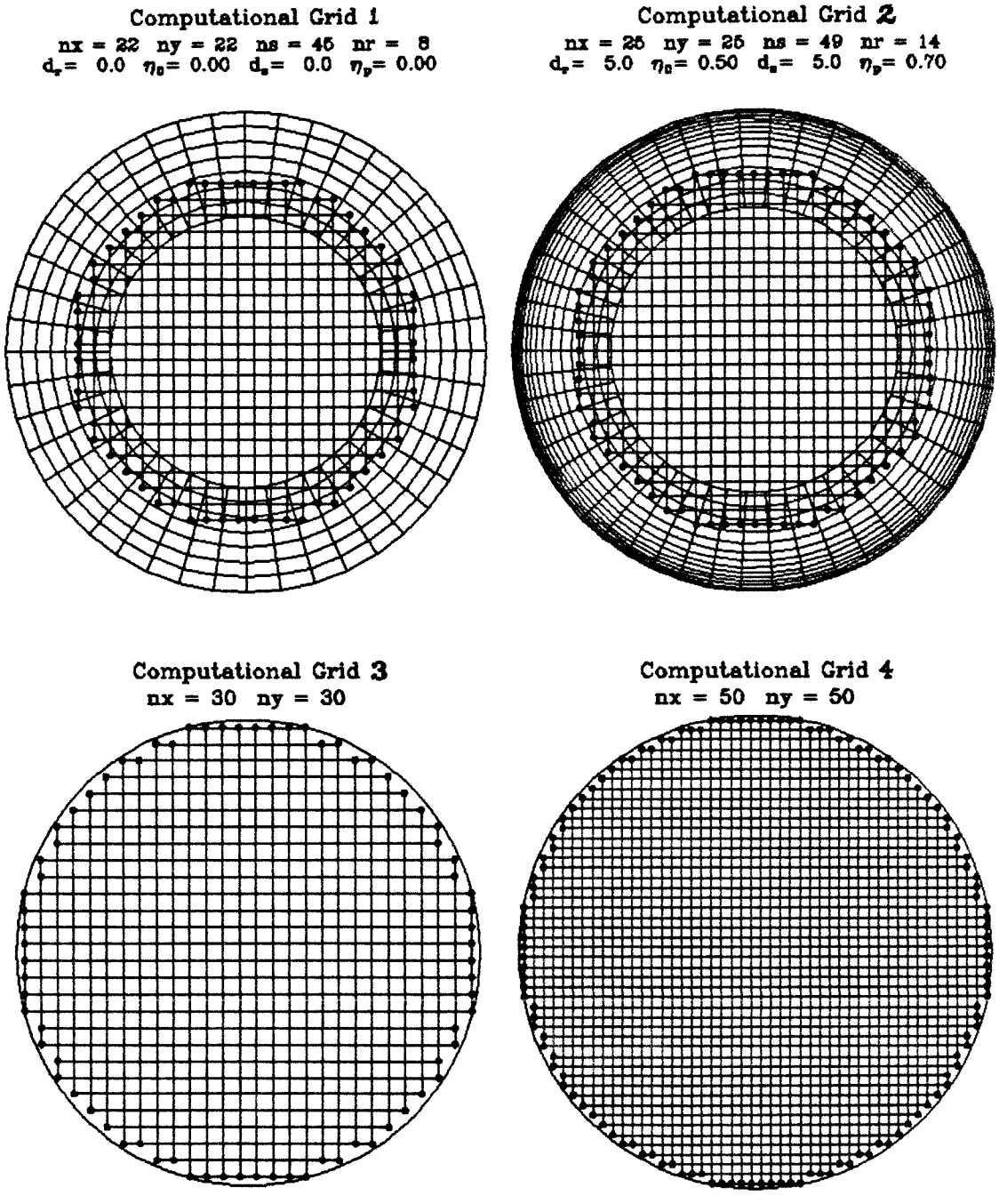


Figure 6.3 Meshes for the Comparison Runs

This stream function satisfies the no slip boundary conditions and so will be a solution of the equations (1) provided the forcing is chosen as

$$f(x,y,t) = \zeta_t^T - R_0(\psi_y^T \zeta_x^T - \psi_x^T \zeta_y^T) - \beta \psi_x^T - E_k \nabla^2 \zeta^T$$

This test example gives an indication of the accuracy of the codes when the solution is smooth. The test is somewhat artificial since the solution has no boundary layers. The other parameters in the equations are taken to be

$$R_0 = 10^{-1}$$

$$E_k = 10^{-2}$$

$$\beta = 1$$

Table 6.5 shows the results for runs on the 2 composite meshes. The errors in the computed vorticity and stream function on the two component meshes are given. The errors are given as normalized l_2 errors. The square of the l_2 error for a component mesh function $\underline{v}_p$ is defined to be the sum of the squares of the mesh function values divided by the number of values on the component mesh.

$$\|\underline{v}_p\|_2^2 = \frac{1}{\#N_p} \sum_{i,j} \underline{v}_p{}^2(i,j)$$

Note that the definition of the norm $\|\cdot\|_2$ depends on the component mesh function which appears as its argument. The infinity norm of a component mesh function is defined as

$$\|\underline{v}_p\|_\infty = \max_{N_p} \quad |\underline{v}_p(i,j)|$$

The errors in the vorticity and stream function given in the table are

$$e_p^\zeta = \frac{\|\underline{\zeta}_p - \underline{\zeta}_p^T\|_2}{\|\underline{\zeta}_p^T\|_\infty} \qquad p = 1,2$$

$$e_p^\psi = \frac{\|\underline{\psi}_p - \underline{\psi}_p^T\|_2}{\|\underline{\psi}_p^T\|_\infty} \qquad p = 1,2$$

Grid	Method	Time	e_1^{ς}	e_2^{ς}	e_1^{ψ}	e_2^{ψ}	time/step
1	Implicit	.25	.012	.00083	.0061	.0047	
		.75	.015	.0014	.0073	.0056	2.7
1	Explicit	.25	.013	.00085	.0065	.0049	
		.75	.016	.0016	.0078	.0058	1.9
2	Implicit	.25	.011	.00026	.0036	.0015	
		.75	.013	.00040	.0041	.0018	4.8

Table 6.5 Errors on Composite Meshes

$\underline{\varsigma}_p^T$ and $\underline{\psi}_p^T$ are the component mesh functions corresponding to the true solutions ς^T and ψ^T. The annular mesh is component mesh $p = 1$ and the interior square grid the component mesh $p = 2$.

Since grid 1 is not stretched near the boundary it is feasible to use the explicit time stepping procedure. Only the implicit method was run on grid 2. The table also shows the computer time, in seconds, required per step for each of the methods.

Table 6.6 shows the errors in the numerical solution for the rectangular grids. The errors e^{ς_R} and e^{ψ_R} can be defined in the same manner as e_p^{ς} and e_p^{ψ}, if one considers the rectangular grid as a component mesh. The errors in the vorticity on the rectangular grids are quite large near the boundary.

Grid	Method	Time	e^{ς_R}	e^{ψ_R}	time/step
3	Explicit	.25	.15	.0047	
		.75	.15	.0036	1.5
4	Explicit	.25	.13	.0036	
		.75	.13	.0022	4.7

Table 6.6 Errors on the Rectangular Grid

Figures 6.4 and 6.5 show plots of the numerical solutions on grids 1 and 4 respectively. The corresponding plot for grid 2 is much the same as the one for grid 1. Since there are no boundary layers in the solution no advantage is gained

from the stretched grid. The contour plots of the stream function and vorticity on figure 6.4 were made by calling a contour plotter twice, once for each component mesh. The contours match up across the interpolation region. This is a further indication of the accuracy (or inaccuracy) of the solution. Some of the parameters related to the run are given above each contour plot. This information includes the maximum and minimum solution values, along with the interval between contours, CI. Negative contours are plotted as dashed lines.

Comparison Run

To see how the codes and grids compare on a more realistic problem we solve with parameters and forcing given as

$$R_0 = 10^{-3} \qquad E_k = 10^{-4}$$

$$\beta = 1 \qquad f = -1$$

This is a similar problem to the one solved by Beardsley [1973] and Reyna [1983]. The forcing f (curl of the wind stress) causes the ocean to rotate in a clockwise direction. A strong boundary layer develops on the western coast. A time sequence of the solution is given in figure 6.6. The solutions at time $t = 50$ for the 4 grids are shown in the sequence of figures 6.7 to 6.10. Each figure shows the mesh that was used and contour plots of the vorticity and stream function. In addition the vorticity along the line $y = 0$ is plotted. This profile can be compared to the approximate solution of Munk which was obtained in section 3.2. This approximate solution predicts the vorticity at the boundary to have a value given by

$$\varsigma(-1) \approx -\frac{2\tau}{\beta}\lambda^2(1 - \frac{1}{\lambda})$$

which for $E_k = 10^{-4}$, $\tau = -1$ and $\beta = 1$ gives $\lambda = 10^{\frac{4}{3}} \approx 20$ and $\varsigma(-1) \approx 900$. The vorticity profile along $y = 0$ is in qualitative agreement with the Munk approximation. There is a strong oscillating boundary layer on the western coast, and a smaller boundary layer on the eastern coast.

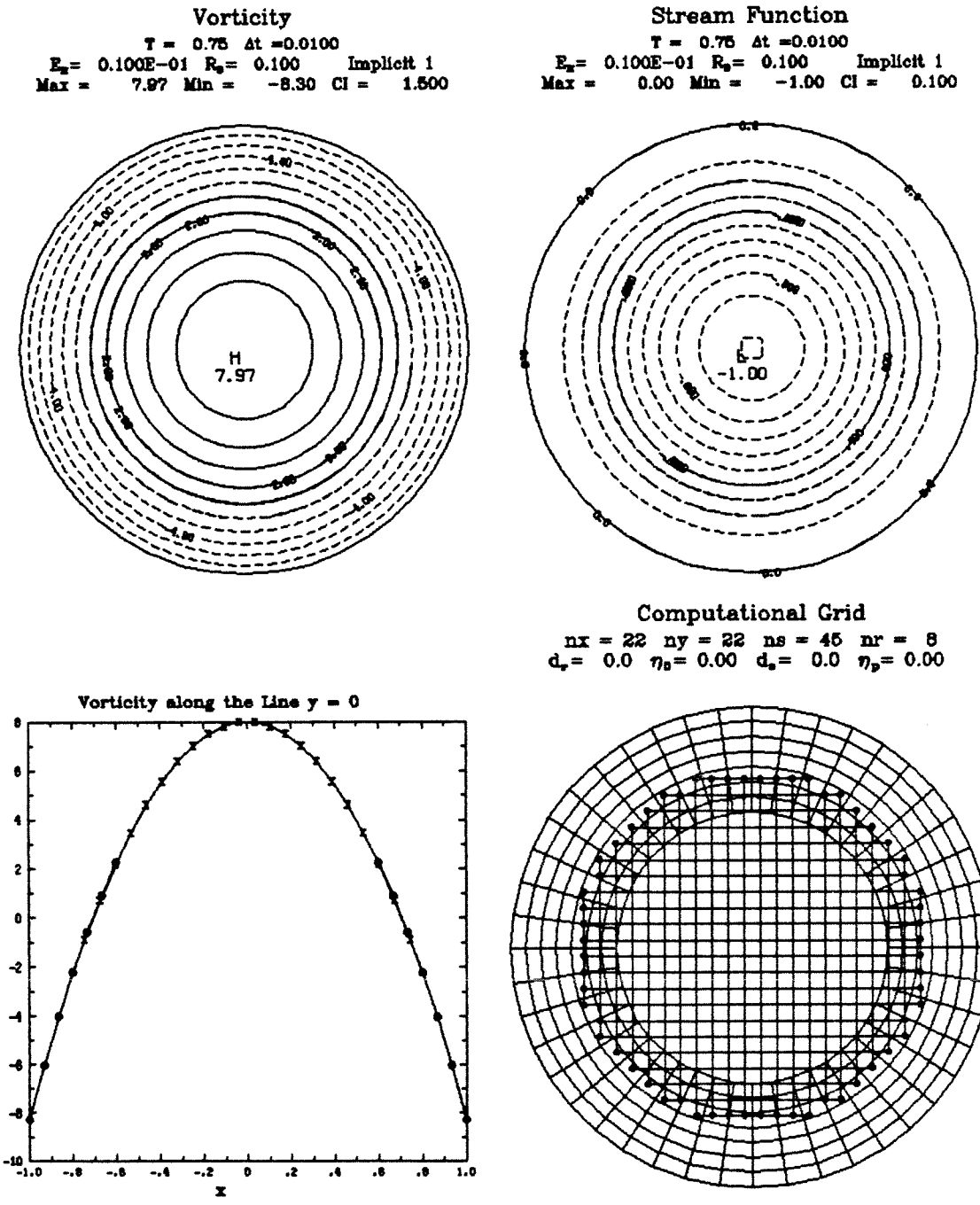
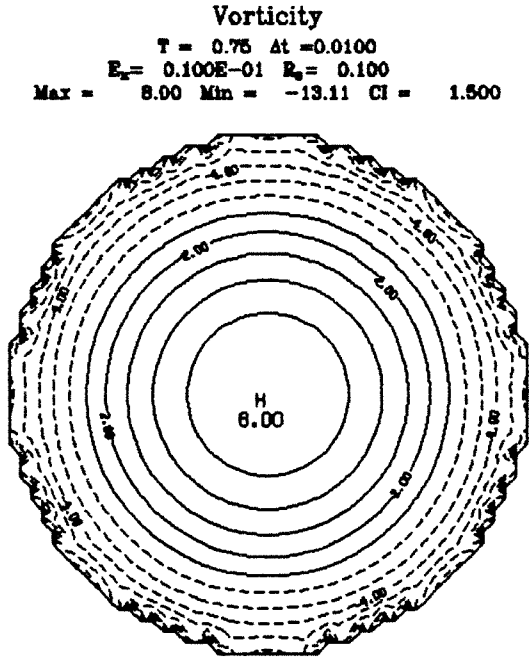
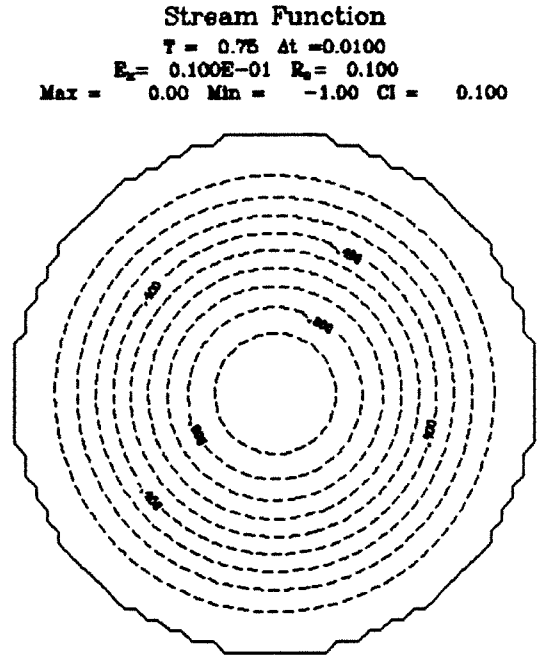
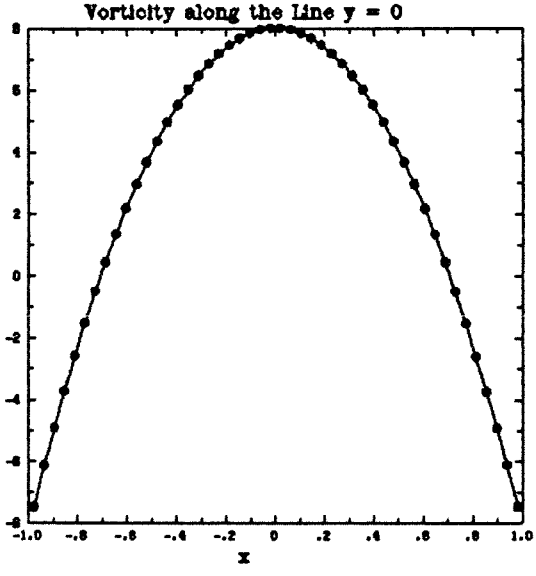


Figure 6.4 Accuracy Test - Grid 1







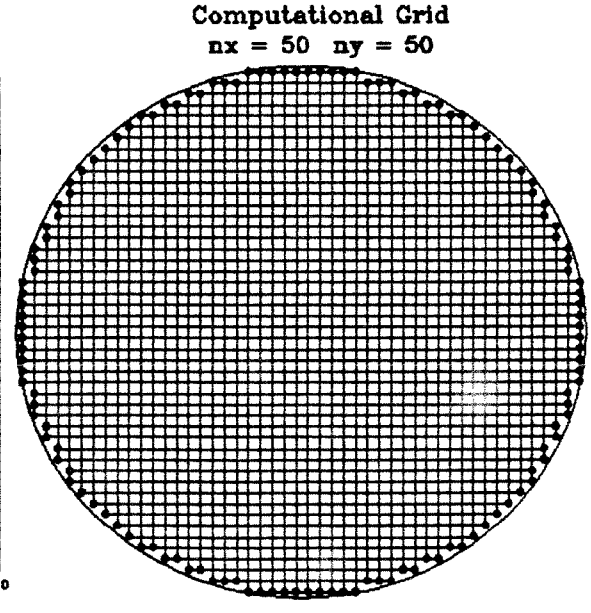


Figure 6.5 Accuracy Test - Grid 4

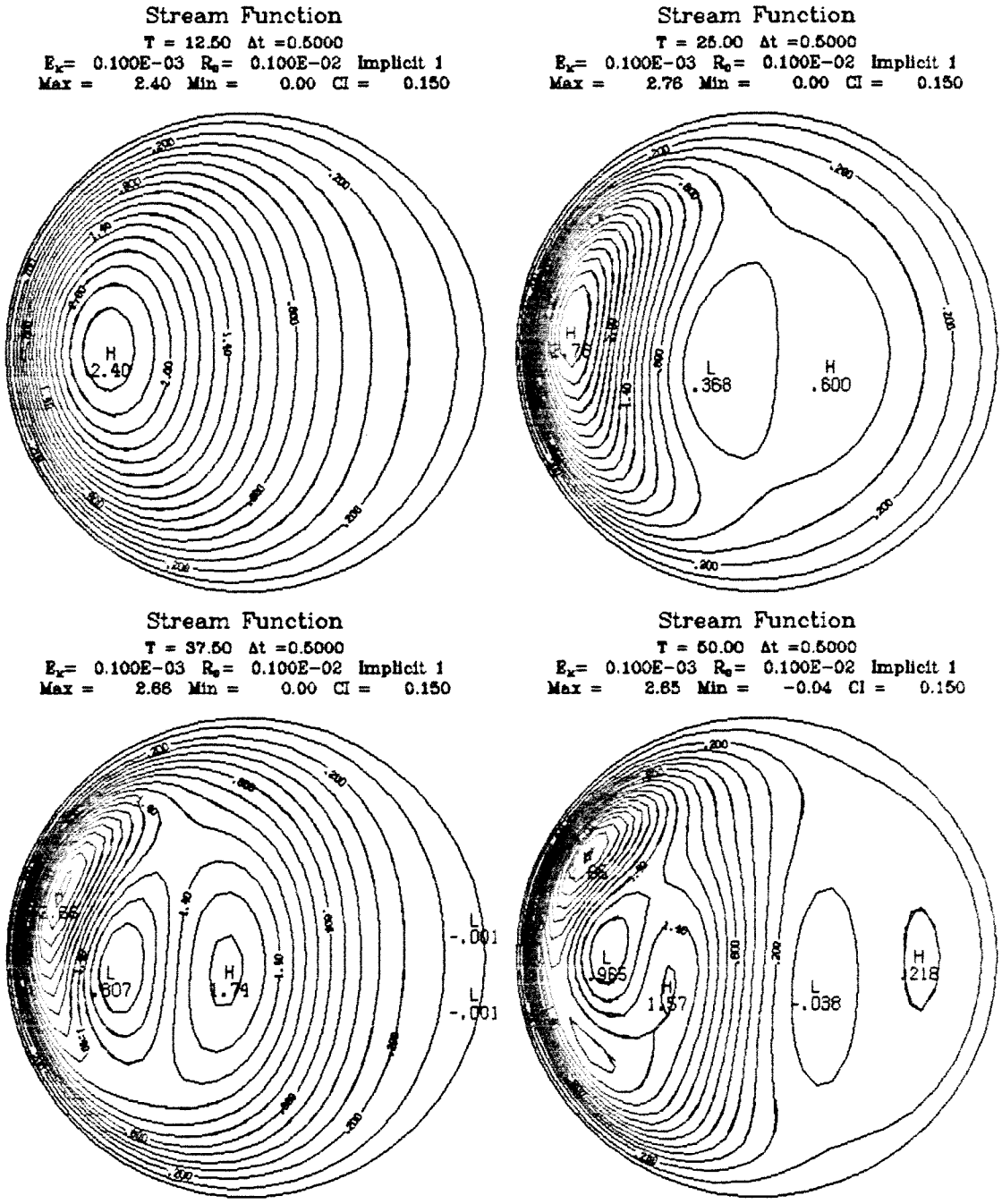
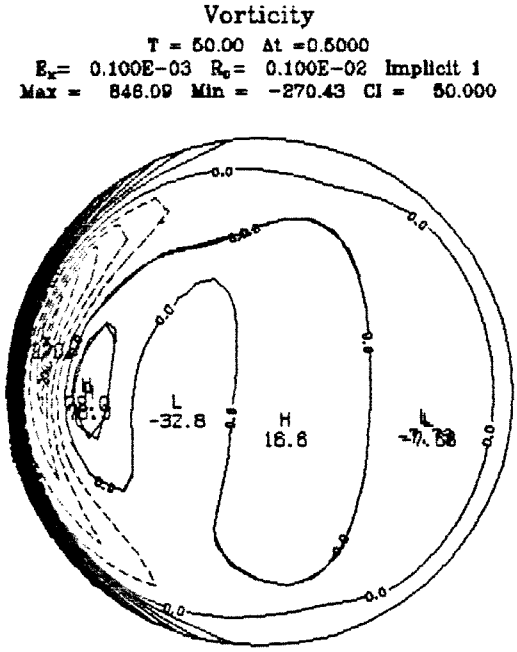
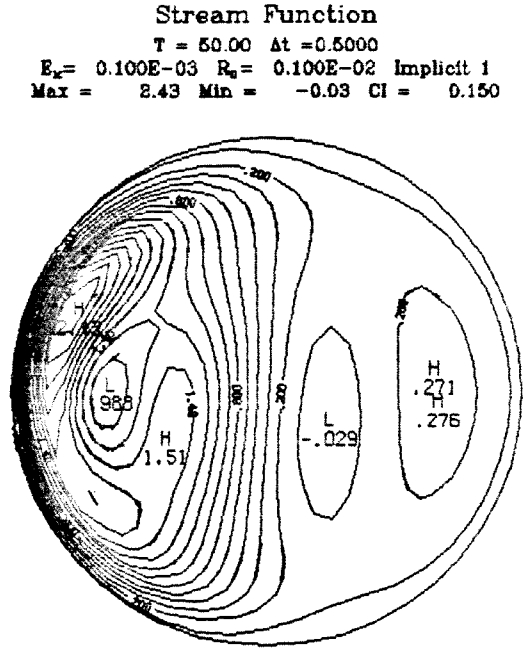
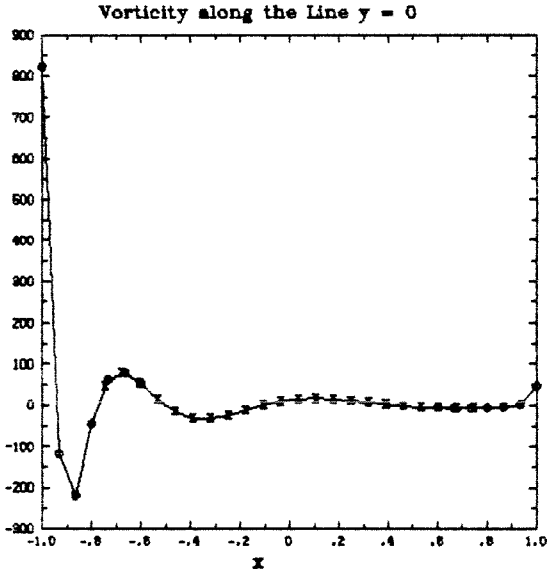


Figure 6.6 Comparison Run - Time Development







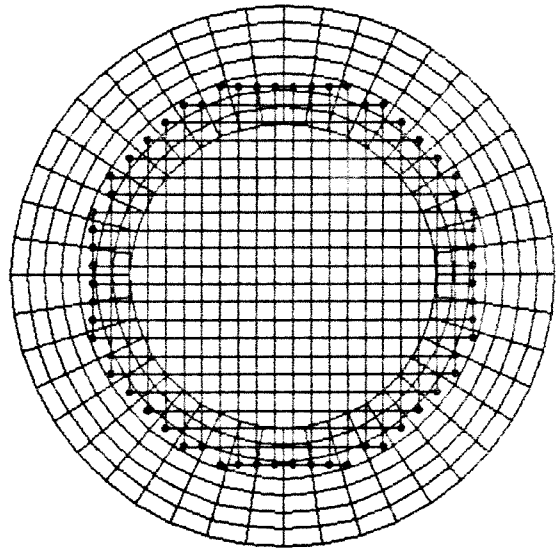


Figure 6.7 Comparison Run - Grid 1

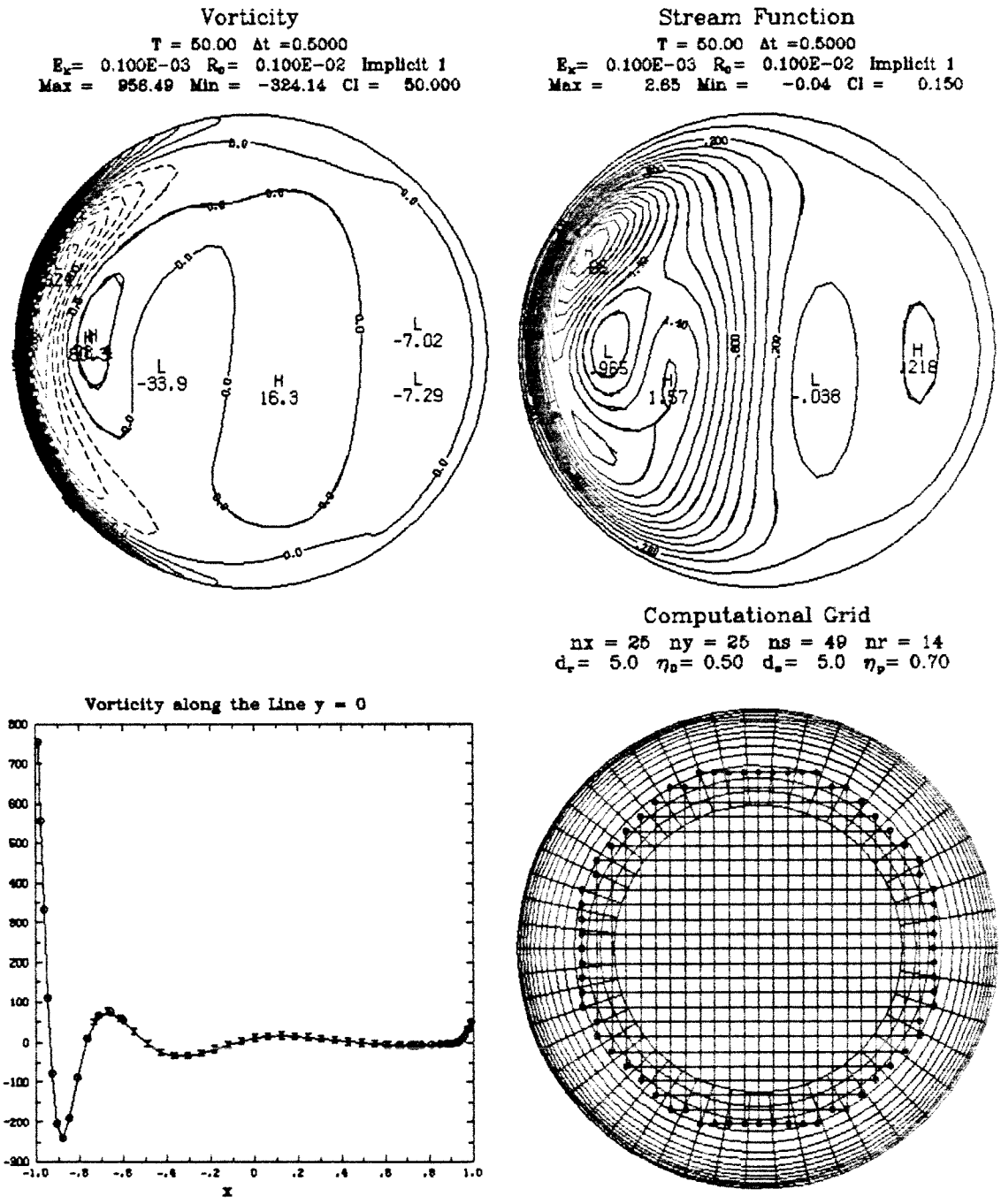


Figure 6.8 Comparison Run - Grid 2

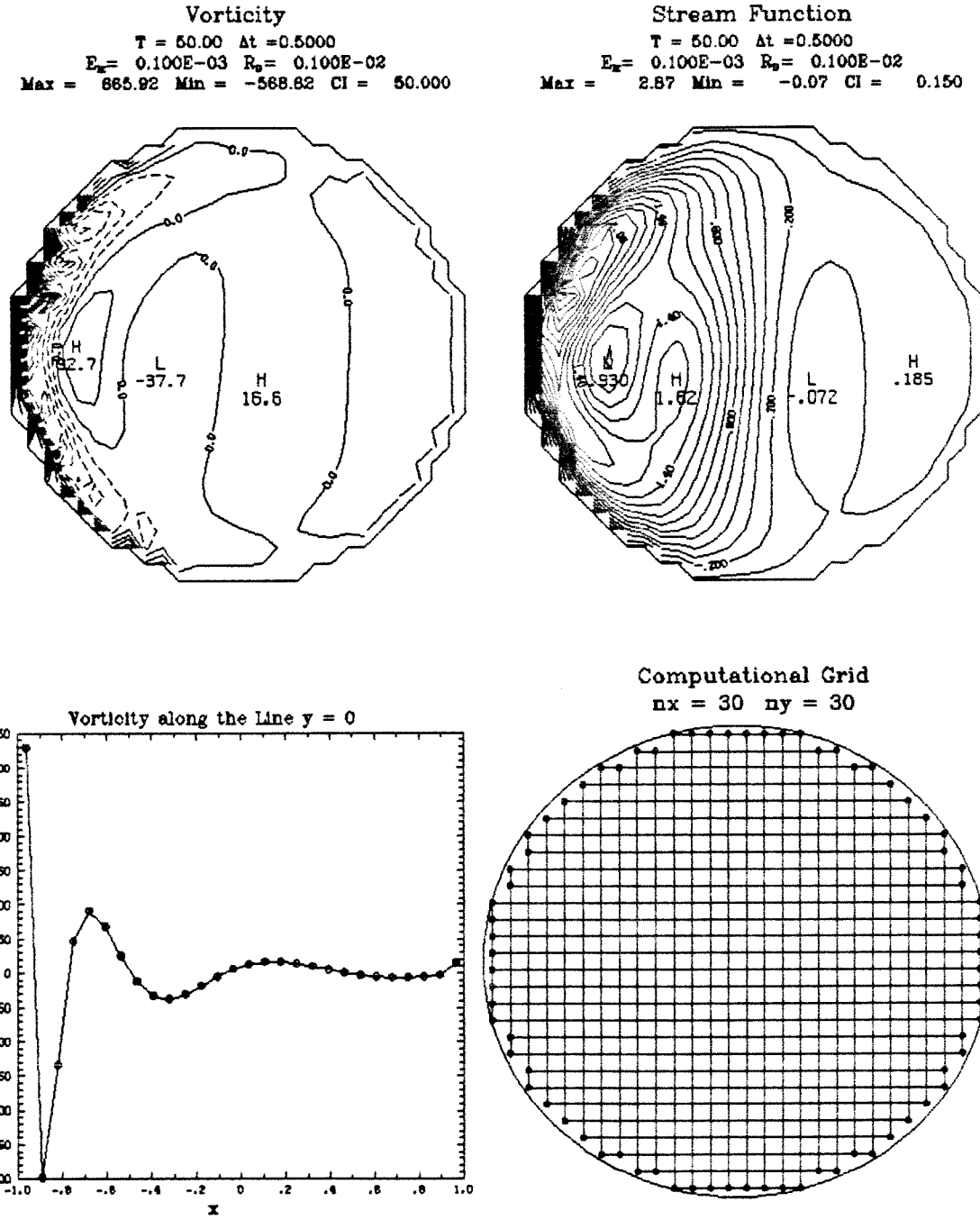


Figure 6.9 Comparison Run - Grid 3

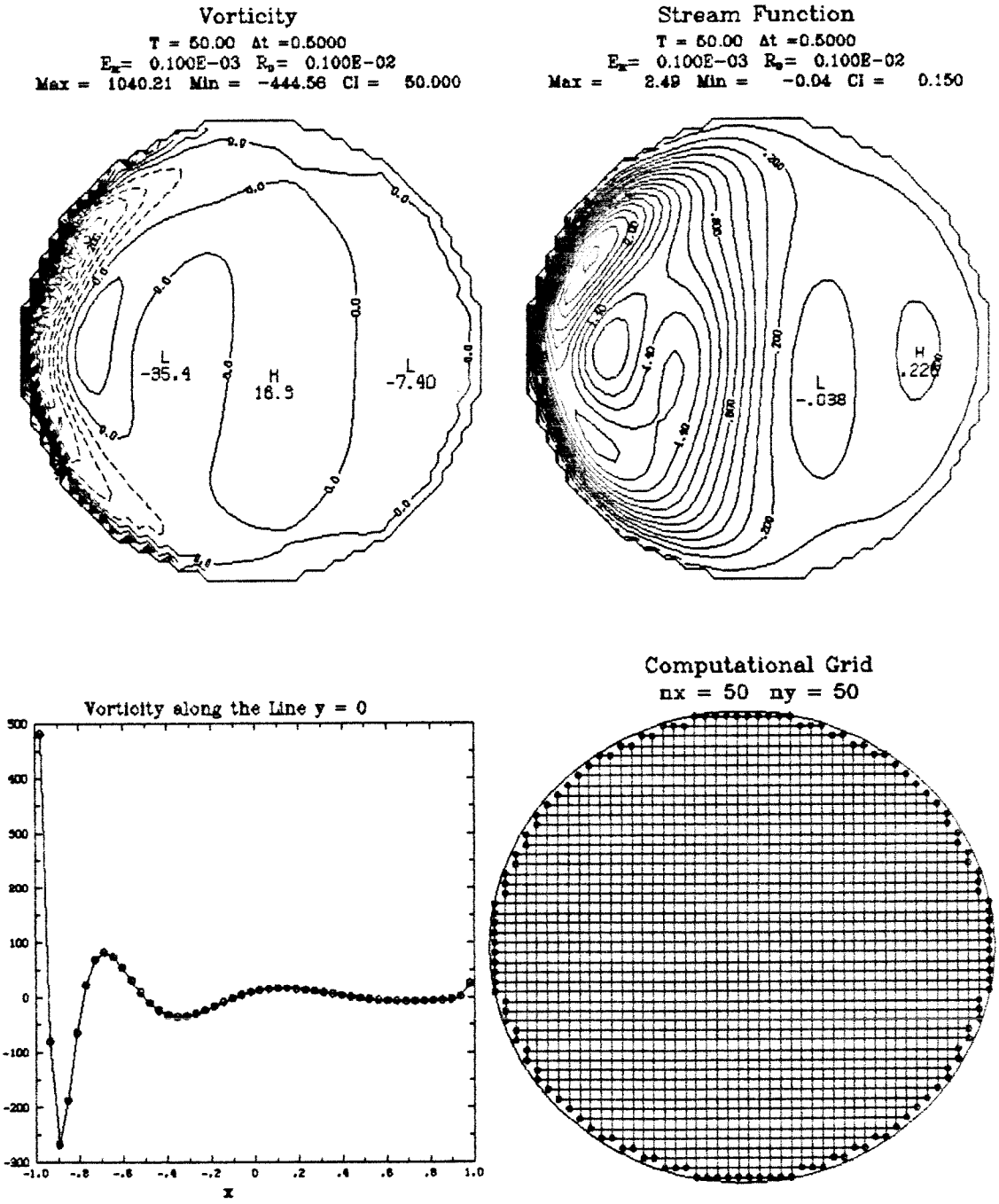


Figure 6.10 Comparison Run - Grid 4

The global features of the stream function solutions agree fairly well on all grids. The vorticity near the wall for the rectangular grid models, however, is not very smooth. Only the stretched composite mesh, grid 2, has enough points to resolve the strong western boundary layer. When looking at these results keep in mind that a second derivative of the vorticity must still be estimated using divided differences. This estimate will be accurate only if the vorticity is fairly smooth.

6.3 A Run on a Large Ocean

As a final example the vorticity stream function equations are solved on the non-circular grid shown at the top of figure 6.11. The shape of the region was chosen in order to study the behaviour of a strong current as it rounds a sharp corner. One might imagine the eastern basin to be the Indian ocean and the central peninsula to be the southern tip of Africa. In dimensional units, the parameters describing the run have the values

$$L = 1 \times 10^6 m$$

$$F = 10^{-11} s^{-2}$$

$$B = 2 \times 10^{-11} m^{-1} s^{-1}$$

$$A_H = 2000 m^2 s^{-1}$$

These correspond to values of the nondimensional parameters of

$$R_0 = 2.5 \times 10^{-2}$$

$$E_k = 1 \times 10^{-4}$$

The curl of the wind stress is chosen to be

$$f(x, y, t) = \begin{cases} \sin((\pi/l_0)(y - y_0)) & y > y_0 \\ 0 & y < y_0 \end{cases}$$

where y_0 is the vertical position of the bottom tip of the peninsula and l_0 is the distance from this tip to the top of the eastern ocean basin. This forcing causes the ocean to rotate in a counter-clockwise direction. Figures 6.11 to 6.13 show the

solutions at the nondimensional times of $t = 10, 20, \ldots, 100$. The stream function was given the initial value of zero. The flow gradually increases in intensity until the velocities are large enough so that nonlinear effects become important. This run was made primarily to show the feasibility of the method. The number of points on the grid is small enough to allow the program to be run on a Vax. The total CPU time for this run, including input and output was about 1 hour. Further studies are planned.

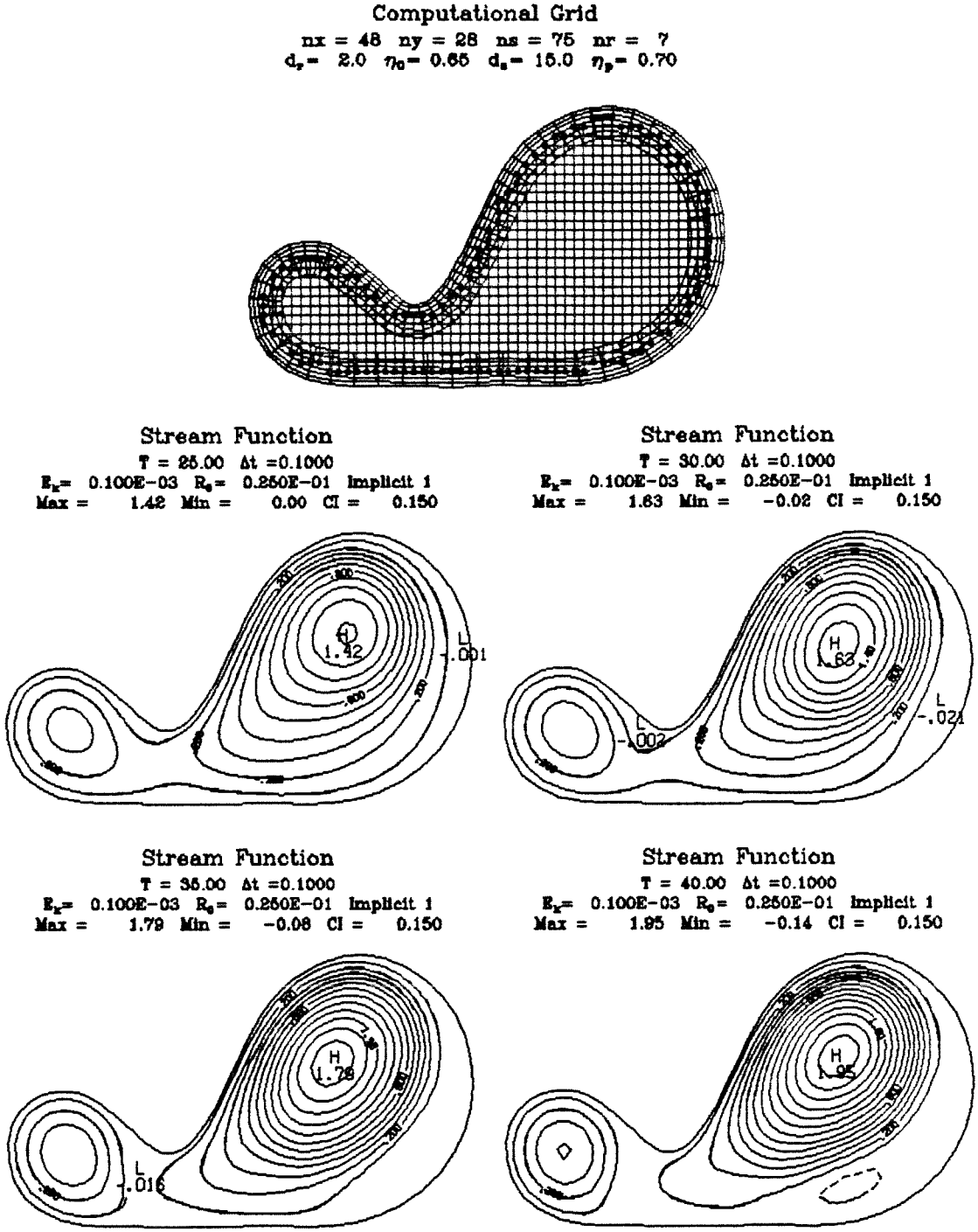


Figure 6.11 Big Ocean Run

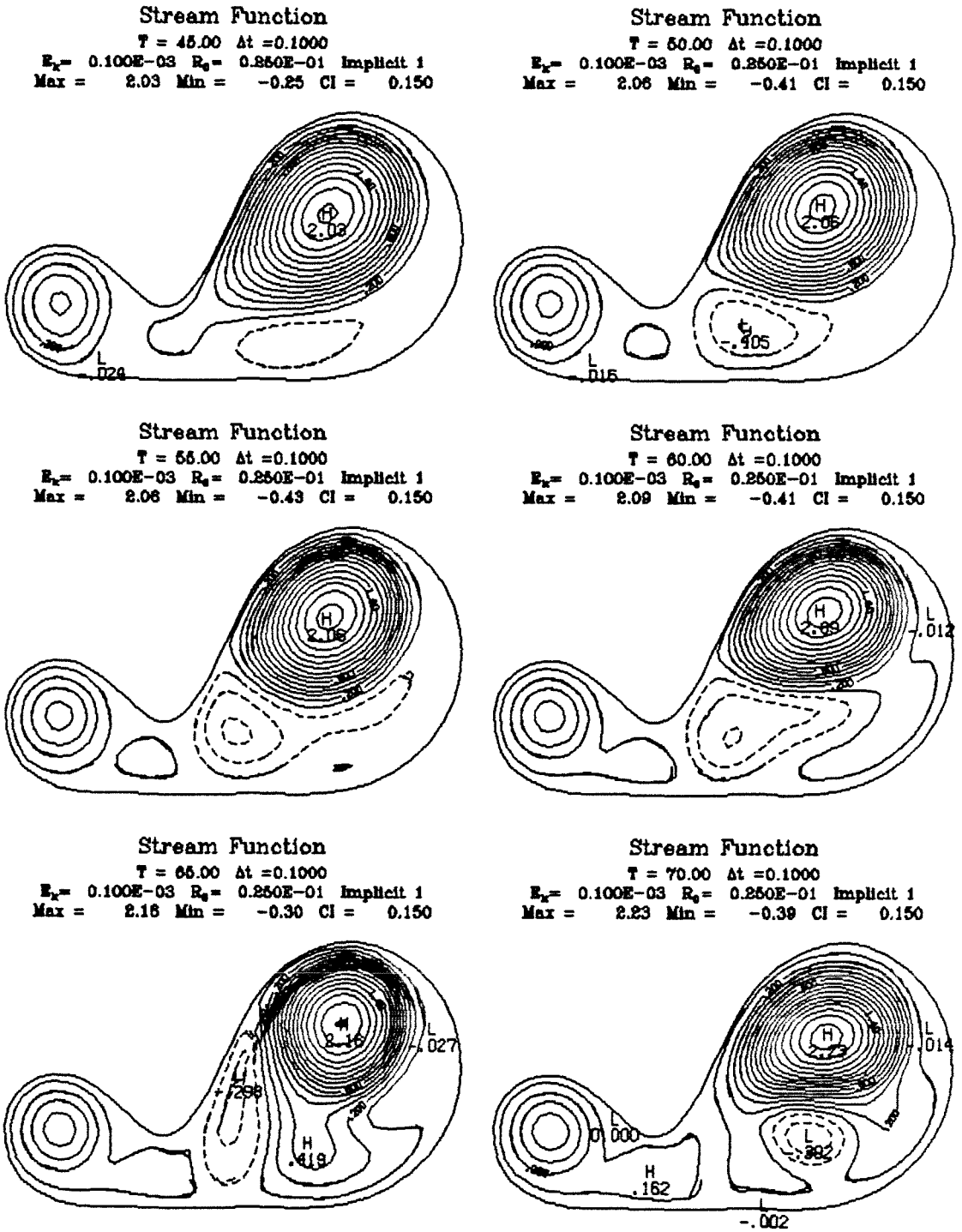


Figure 6.12 Big Ocean Run (continued)

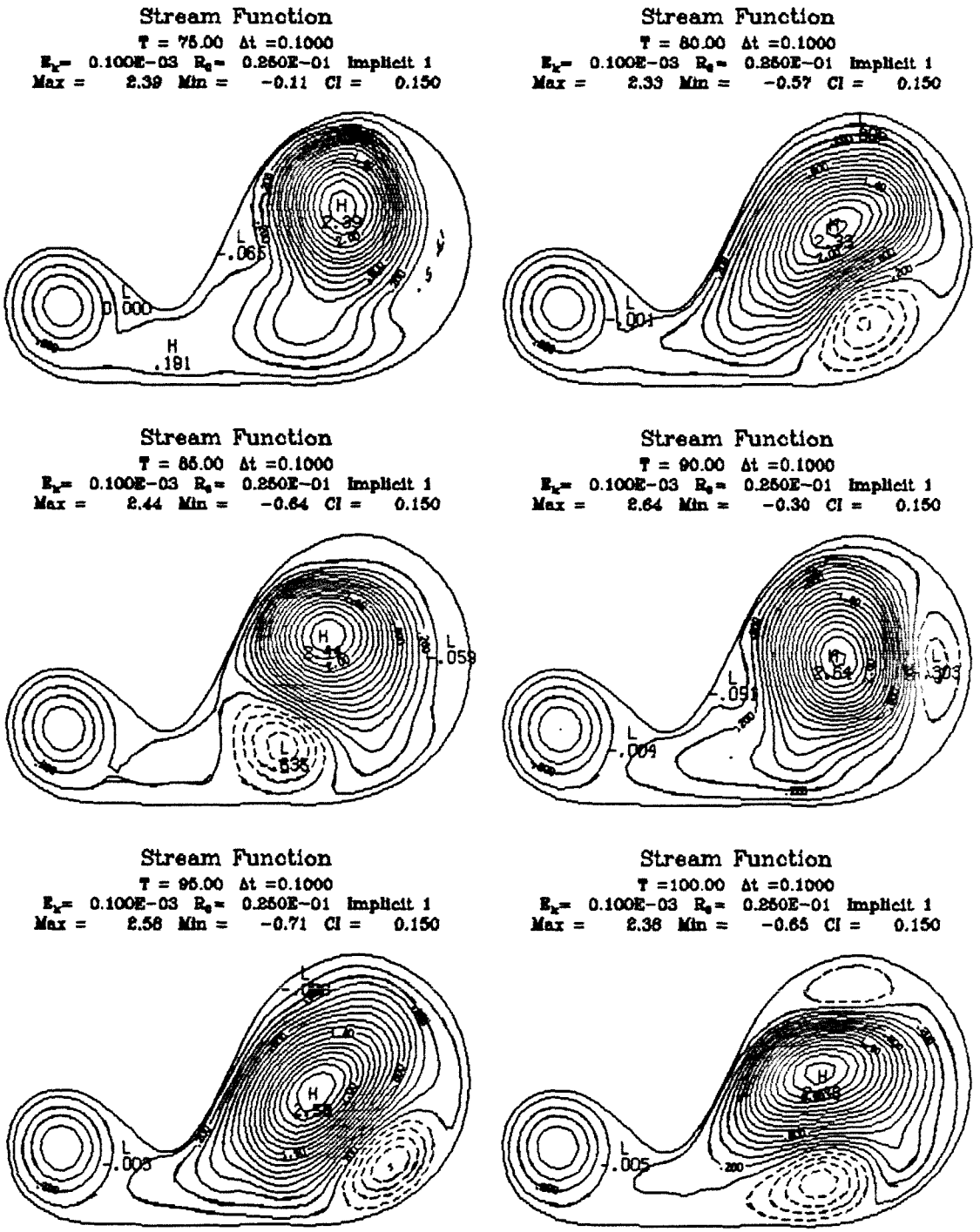


Figure 6.13 Big Ocean Run (continued)

References

R.C. Beardsley *A Numerical Model of the Wind Driven Ocean Circulation in a Circular Basin*, Geophysical Fluid Dynamics, **4**, pp. 211-241, 1973.

M. Berger, *On Conservation at Grid Interfaces*, ICASE report no. 84-43, 1984.

P. Bontoux, B. Gilly and B. Roux, *Analysis of the Effect of Boundary Conditions on the Numerical Stability of Navier Stokes Equations*, J. Comp. Phys., **15**, pp. 417-427, 1980.

A. Brandt, *Multi-Level Adaptive Solutions to Boundary Value Problems*, Math. Comp., **31** no. 138, pp. 333-390, April 1977.

W.R. Briley, *A Numerical Study of Laminar Separation Bubbles using the Navier Stokes Equations*, J. Fluid Mech., **47**, part 4, pp. 713-736, 1971.

G.S. Chesshire, *A Composite Grid Generation Program*, to appear.

S.C. Eisenstat, M.C. Gursky, M.H. Schultz and A.H. Sherman, *Yale Sparse Matrix Package I. The Symmetric Codes, II. The Nonsymmetric Codes*, Research Reports 112 and 114, Yale University, Department of Computer Science, May 1977.

M.M. Gupta and R.P. Manohar, *Boundary Approximations and Accuracy in Viscous Flow Computations*, J. Comp. Phys, **31**, pp. 265-288, 1979.

W. Hackbusch and U. Trottenberg (eds.), *Multigrid Methods*, Proceedings of the Conference Held at Koln-Porz, Lecture Notes in Mathematics, **960**, Springer-Verlag, Berlin 1982.

M. Israeli, *A Fast Implicit Numerical Method for Time Dependent Viscous Flows*, Studies in Applied Math, vol. XLIX, **4** , December 1970.

B. Kreiss, *Construction of a Curvilinear Grid*, SIAM J. Sci. Stat. Comput., Vol. 4, No. 2, pp. 270-279, June 1983.

W.H. Munk, *On Wind Driven Ocean Circulation*, J. Meteor. **7**, pp. 79-93, 1950.

S.A. Orszag and M. Israeli, *Numerical Simulation of Viscous Incompressible Flows*, Annual Rev. Fluid Mech., **6**, pp. 281-318, 1974.

J. Pedlosky, *Geophysical Fluid Dynamics*, Springer-Verlag, 1982.

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

L.G.M. Reyna, PhD. thesis, California Institute of Technology, 1982.

P.J. Roache, *Computational Fluid Dynamics*, Hermosa Publishers, Albuquerque, 1972.

G. Starius, *Composite Mesh Difference Methods for Elliptic and Boundary Value Problems*, Numer. Math., **28**, pp. 243-258, 1977.

G. Starius, *On Composite Mesh Difference Methods for Hyperbolic Difference Equations*, Numer. Math., **35**, pp. 241-255, 1980.

K. Stubën, and U. Trottenberg, *Mulitigrid Methods: Fundamental Algorithms, Model Problem Analysis and Applications*, Universität Bonn, preprint no. 544, 1982.