University of Reading
Department of Mathematics

The Semi-Lagrangian Method in Atmospheric Modelling

Christopher J. Smith

Thesis submitted for the degree of Doctor of Philosophy.

March 2000

Abstract

In this thesis we present a review of the recent developments in the semi-Lagrangian method. This numerical method for advection problems has found particular favour in the meteorological community, where it provides particular advantages for operational weather forecasting. Our approach is to describe, in some detail, a selection of the significant contributions to the method's development which have appeared in the literature since the 1991 review of Staniforth and Côté.

We begin with an examination of available interpolation routines, and the need for shape-preserving properties in interpolation. The noninterpolating framework is next discussed.

The latter part of the thesis is directed towards identifying schemes optimised for use in global atmospheric simulation. We discuss a method for solving the shallow-water equations, using a fully semi-Lagrangian discretisation for both linear and nonlinear advection.

Finally, two issues of particular importance in atmospheric applications are discussed. First, we see that the lack of formal conservation properties in early semi-Lagrangian schemes has been addressed by various authors in the literature. Several schemes are now available which offer exact, or nearly exact conservation. Secondly, the problem of spurious numerical resonance is examined. Semi-Lagrangian methods have been found to be prone to spurious resonance in models containing orography. Again, suitable remedies for this problem have appeared in the literature, further demonstrating the richness of the semi-Lagrangian method as an area for development.

Acknowledgements

I would like to thank Prof. Mike Baines for many memorable conversations and for his spirited supervision of this work; Dr. Mike Cullen for his patient attempts to guide me toward deeper waters, and all staff (working and academic), students and visitors at Reading for providing such a friendly environment. Special thanks also to Anne and Alison for redefining office life, and for their continuing friendship and support.

I gratefully acknowledge the financial support of the EPSRC and U.K. Meteorological Office through the CASE studentship scheme.

Contents

Introduction **1**

1 The Semi-Lagrangian Method **3**

1.1 Introduction 3

1.2 Linear Advection using SL 6

1.2.1 Homogeneous Constant Velocity Advection 6

1.2.2 Non-constant Velocity Advection 9

1.2.3 Source Terms 11

1.3 Nonlinear Advection 12

2 Interpolation: One-Dimensional Schemes **16**

2.1 Introduction 16

2.2 Univariate Interpolation 17

2.2.1 Example: Piecewise Cubic Interpolation 18

2.3 A First Order Scheme 19

2.4 Higher Order Schemes 23

2.4.1 Linear Schemes 24

2.4.2 Linear Analysis 29

3 Quadratic Interpolation **36**

3.1 Centred Quadratic Interpolation 37

3.2 Computational Tests 40

3.2.1 Test 1: Interpolation . . . 41
3.2.2 Test 2: Advection Test . . . 43
3.2.3 Test 3: Monotone Advection . . . 44
3.2.4 Test 4: Comparison with Regular-Grid Scheme . . . 45
3.2.5 Conclusion . . . 47

4 Nonlinear Schemes **48**
4.1 ENO Interpolation . . . 48
4.2 Hermite Cubic Interpolation . . . 50
4.3 Quasi-Monotone Schemes . . . 52
4.3.1 QMSL Algorithm . . . 52
4.4 A Computational Example . . . 54

5 Interpolation: Multi-Dimensional Schemes **58**
5.1 Introduction . . . 58
5.2 Tensor Product Interpolation . . . 59
5.3 Leslie and Purser: Cascade Interpolation . . . 60

6 Non-interpolating Methods **65**
6.1 Ritchie's Scheme . . . 66
6.2 Olim's Scheme . . . 67
6.3 Smolarkiewicz et al. . . . 68
6.3.1 Stokes' Theorem . . . 68
6.3.2 Contour Decomposition . . . 69
6.4 Non-interpolating Trajectory Methods . . . 72

7 Nonlinear Advection **74**
7.1 Burgers Equation . . . 74
7.2 Setting up Computational Tests . . . 76
7.3 Inviscid Burgers' Test Problem . . . 77

7.3.1 Reference Solution 77
7.3.2 Time of Shock Formation 79
7.3.3 Results 81
7.4 Viscous Burgers' Test Problem 86
7.4.1 An Analytical Solution 88
7.4.2 Test 1: Departure Point Iterations 90
7.4.3 Error Measurements 90
7.4.4 Test 2: Interpolation 93
7.4.5 Test 3: Departure Point Method 95
7.4.6 Conclusion 99
7.5 Idealised Atmospheric Flow Results 99
7.6 Conclusion 103

8 Global Forecasting **105**
8.1 Geophysical Spherical Coordinates 107
8.2 Shallow Water Equations on the Sphere 108
8.3 Vector Discretisation of Momentum Equation 109
8.4 Calculating Departure Points 111
8.5 A Geodesic Method for Calculating Departure Point Coordinates. 113
8.5.1 Geodesic Equations for the Sphere 115
8.5.2 Departure Point Scheme — Outline 117
8.5.3 Departure Point Scheme — Practical Details 119
8.5.4 Great Circle Constant Speed Approximation 119
8.5.5 Efficient use of Trig Calculations 121
8.5.6 Departure Point Scheme — Final Form 122

9 Mass Conserving Schemes **125**
9.1 Conservation and Continuity 127
9.2 Eulerian Conservative Schemes 129

9.3 Lagrangian Conservative Scheme 132
9.4 The Rezoning Method of Rančić, Plante and Laprise 133
9.4.1 Upstream Scheme 133
9.4.2 Downstream Scheme 137
9.4.3 Performance of Schemes 138

10 Numerical Experiments with Conservative Schemes **142**
10.1 Introduction 142
10.2 Remapping Scheme 143
10.2.1 Exact Integration 143
10.3 Results of Comparison Tests 149

11 Spurious Orographic Resonance **155**
11.1 Three-Time-Level Schemes 158

Bibliography **163**

List of Figures

1.1 Eulerian and Lagrangian numerical models 4

1.2 Basic SL method 8

2.1 First order SL scheme, using linear interpolation 22

2.2 Lagrange basis function $l_1(x)$ for cubic interpolation 25

2.3 Lagrange basis function $l_2(x)$ for cubic interpolation 25

2.4 High order SL using cubic interpolation 26

2.5 Fourier analysis of SL using linear interpolation 33

2.6 Fourier analysis of SL using cubic interpolation 34

2.7 Fourier analysis of SL with Hermite cubic interpolation 35

3.1 Interpolated function on a 24 point grid. 42

3.2 Constant speed advection test for four different centred quadratic interpolants, on an irregular grid. Advected fields shown after 1000 timesteps, with local Courant number varying between 0.16 and 0.48. 45

3.3 Advection test comparing q^M with monotone filter against quadratic ENO method. Data advected at constant speed across an irregular mesh for 1000 timesteps. 46

3.4 Comparison of mean and Fromm quadratic interpolants. Constant speed advection test over an irregular grid. Solutions shown after 1000 timesteps. 47

4.1 Cubic Interpolation without monotonicity constraint 55

4.2 Monotone solution 56

5.1 Hybrid Grids 61

6.1 Ritchie's scheme 67
6.2 Contour Decomposition 69

7.1 Solution of inviscid Burgers' equation using standard SL scheme: cubic interpolation; departure points calculated using time extrapolated velocity, four iterations of implicit mid-point method. Dashed lines show solution obtained by Roe's scheme on a finer mesh. 82
7.2 Solution of inviscid Burgers' equation using standard SL scheme: as Figure 7.1 except $u_{min} = 0$ in initial data. 83
7.3 Solution of inviscid Burgers' equation using standard SL scheme: initial data and SL solution shown at 75 step intervals; reference solution in dashed line. SL solution is well-behaved for times before shock forms. . . 84
7.4 Plots of L2 error and conservation error against time for SL solution of inviscid Burgers' equation. Results shown for four different interpolation methods. 86
7.5 Maximum error against mesh ratio parameter. The maximum error is the largest error for inviscid Burgers simulation which runs to just before the time of shock formation. 87
7.6 Maximum error against mesh ratio, for SL solution of inviscid Burgers' problem. The integration extends only to eighty percent of the shock time. 88
7.7 Moving front solution of viscous Burgers equation. Analytical solution (dashed lines) and SL solution (solid lines) after 0, 71, 142, 213, 284 and 355 steps. Mesh ratio $\Delta t/\Delta x = 1.7$. Two iterations for calculating departure points. 91
7.8 As Figure 7.7 but with 4 iterations of departure point scheme. 92

7.9 Position and shape error for SL integration of viscous Burgers' equation, through 355 timesteps. Results shown for three different interpolants used to evaluate quantities at departure point. 94
7.10 Shape error for linear advection of front at constant speed. The accuracy of the solution depends precisely on the order of accuracy of the interpolation used. 95
7.11 Error in speed of front and error in shape of front plotted against mesh ratio, for three different methods of calculating departure points. 99
7.12 Horizontal and vertical wind fields at quasi-steady state. Model represents neutrally stratified and inviscid flow. 101
7.13 Normalised kinetic energy against time, showing deceleration of the quasi-steady state. Results shown for four interpolants of different formal accuracy applied to wind fields. 102

8.1 Geophysical spherical coordinates 124

9.1 Control Volumes and Grids 135

10.1 Mass histories for non-conservative schemes 149
10.2 Final distribution for non-monotone scheme 151
10.3 Final distribution for monotone scheme 152
10.4 Mass histories for conservative SL schemes 153
10.5 Final distribution for remapping scheme 153
10.6 Final distribution for QCSL scheme 154

List of Tables

2.1 Centred derivative estimates, employing linear combinations of discrete slopes. 29

3.1 Interpolation error and maximum and minimum values for four quadratic interpolants. 43

3.2 Advection error and maximum and minimum values for four quadratic interpolants. 44

3.3 Advection error and maximum and minimum values for four quadratic interpolants. 47

Introduction

For much of its history, the modern science of meteorology has been dominated by weather forecasting [41]. Even slight gains in forecasting accuracy are of great benefit to areas such as aviation, shipping and farming. Forecasting now commands the use of extensive ground-based and satellite data gathering networks. Weather simulations are run on the largest and most powerful computers. Newer uses of atmospheric simulation range from the long term predictions of climate modelling, to the modelling of local environmental effects of weather. Climate models have an impact on long-term policy making at a national and international level. While at the local level, the contribution of atmospheric modelling to areas such as pollution dispersion, building design and flood and storm warnings is just as great. The uses for atmospheric modelling increase to keep pace with this rapidly developing area of research.

Many physical processes may be represented in atmospheric models. Climate prediction requires the coupling together of many physical systems. Important considerations in this respect are, for instance, the coupling between the oceans and atmosphere; the interaction between cloud formation and the global energy budget [27]; and the feedbacks operating between the biota and the environment [26]. Basic to any model involving spatial and temporal variations in the atmosphere, must be a representation of the dynamics of atmospheric flow.

Of concern in this thesis will be the numerical methods employed in solving the equations describing fluid flow. In particular, we examine the semi-Lagrangian method which has found increasing popularity in atmospheric modelling [55]. Our approach will

be to survey the recent literature on this method.

In Chapter 1 we provide an outline of what constitutes a semi-Lagrangian (SL) method. One of the key elements of the basic SL method will be seen to be interpolation of spatially distributed data. In Chapter 2, 3 and 4, therefore, we examine methods of interpolation. The problem of interpolating data distributed in several dimensions is examined in Chapter 5.

Chapter 6 describes practical alternatives to the use of interpolation in SL schemes. The non-interpolating formalism is seen to offer a broad theoretical basis for the development of new numerical advection schemes.

In Chapter 5 we return to the meteorological theme, to consider global atmospheric modelling with SL.

Chapter 7 considers the application of the semi-Lagrangian method to nonlinear advection equations. This is done in the context of the one-dimensional Burgers' equation, which provides a simple analogue of the nonlinear momentum equation of fluid dynamics.

Chapter 8 presents a discussion of a line of development of SL schemes, towards conservative form.

In Chapter 10 a simple, but demanding test problem is applied to a selection of the schemes described in earlier chapters.

Chapter 11 concludes the thesis by demonstrating the flexibility of SL methods. A problem arising from the early use of SL methods in weather forecasting, was the generation of spurious resonance in the vicinity of mountain ranges. The breadth of the SL framework allowed a solution to be found to this problem [46], which we describe. This final chapter serves to underline the growing importance, and range of application, of the semi-Lagrangian method in atmospheric modelling.

Chapter 1

The Semi-Lagrangian Method

1.1 Introduction

Mathematical descriptions of fluid flow divide into two groups: Eulerian and Lagrangian. Eulerian descriptions work within a single fixed frame of reference, through which the fluid flows. This is rather like observing the flow of a river while stood on one of its banks. The second manner of description is that from within the fluid. This second kind of formalism is known as the Lagrangian description. Here the observer is moving with the fluid — the view-point is that of a swimmer drifting with the river. In contrast to the Eulerian system Lagrangian frames of reference are typically both space and time dependent, according to the nature of the fluid flow.

From these two basic forms of description we can envisage correspondingly distinct forms of numerical model. Representative examples are depicted in figure 1.1. In the Eulerian model we have fixed computational regions, or cells. Each fluid property, such as pressure, density or velocity, is approximated in some way within each cell. For instance the scheme may work with cell-averaged quantities, where each variable is represented by a single value within each cell. To obtain a time dependent simulation of fluid flow the movement of fluid between cells is modelled by considering fluxes across cell-interfaces. So in this kind of model the two basic constructions are cells and fluxes. The Lagrangian

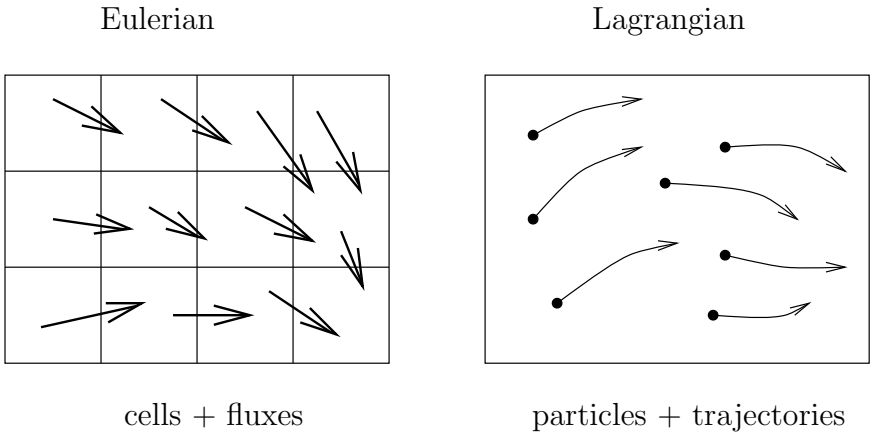

Figure 1.1: Eulerian and Lagrangian numerical models

model, by contrast, does not represent the fluid in terms of continuum ideas at all. The two basic constructions within the Lagrangian model are individual particles and their trajectories. Each particle carries values for all the fluid properties, and moves with the velocity of the fluid flow. Semi-Lagrangian models may be viewed as a hybrid of Eulerian and Lagrangian models. An approach to SL modelling, which combines cells with trajectories as its basic elements, will be examined in detail in Chapter 9.

Other model constructions are of course possible. The finite difference form of Eulerian model in particular provides a simple starting point for SL models. In a finite difference Eulerian scheme cells are replaced by nodes, each node being a point where values of fluid variables are held. Instead of fluxes, a finite difference scheme couples together the variables at different nodes through averaging and differencing operations. The form of these operations derives directly from the differential equations governing the fluid flow. This results in a model which does not necessarily encompass bulk aspects of the flow. For instance, by calculating fluxes between cells we have a frame-work for the global accounting of the movement of mass within a flow. Consequently the principle of mass conservation may quite naturally be incorporated into such a scheme. Whereas a scheme based on local differencing and averaging has no such global organisation, with the result that it may well fail to simulate conservation of mass exactly.

In fact any numerical model we make inevitably has limitations. One failing of a flux-

based Eulerian scheme is the difficulty of achieving stable and accurate simulations with long time-steps. For time-steps sufficiently long to allow fluid to move across several cells in a single step, the sequence of flux transfers between cells becomes extremely difficult to represent [22]. In practice, schemes of this form are operated such that only flows between each cell and its immediate neighbours need be considered during any single time-step. If the scheme is explicit this constraint becomes a necessary condition for stability (the Courant–Friedrichs–Lewy stability condition [6]).

Lagrangian schemes operate with a far less severe restriction on the time-step length, especially where the flow is strongly advection dominated, as is much the case in meteorological applications. For such flows the Lagrangian form of the evolution equations are more tractable numerically than are the equations in Eulerian form. As a result the time-step may be substantially increased beyond what would be acceptable for an Eulerian scheme, yet without detriment either to stability or accuracy.

However, Lagrangian numerical schemes are still methods of approximation and as such subject to limitations of some kind or other. The basic particle–trajectory model invariably falls foul of the profusion of scales which are typical in any realistic fluid flow. An initially evenly distributed collection of particles may, with time, become broken up into clusters and voids. The consequence of this is a loss of model representation for the fluid as a whole.

Semi-Lagrangian models address the above characteristic failings of Eulerian and Lagrangian methods. SL methods are Eulerian-Lagrangian hybrids, which retain the desirable properties of the two formalisms while avoiding their less desirable failings. Like Lagrangian methods, SL methods permit long time-step integrations by tracing trajectories of particles. Yet, like an Eulerian method, SL methods retain the same fixed computational grid throughout the duration of a simulation. This is achieved by producing the solution on the fixed grid at the end of each time step, rather than tracing the long-term histories of individual particles.

In the next section we apply the semi-Lagrangian method to a class of one-dimensional

linear advection problems.

1.2 Linear Advection using SL

In this section we consider how the semi-Lagrangian method may be used to solve linear advection equations in one dimension. The problems to be solved are of the following general form:

$$\begin{aligned} \frac{\partial u}{\partial t} + a(x,t)\frac{\partial u}{\partial x} &= g(x,t) \\ u(x,0) &= u_0(x). \end{aligned} \tag{1.1}$$

We shall develop a SL scheme for (1.1) in three steps. For the first two steps we restrict our attention to the homogeneous equation, for which $g \equiv 0$. For the first step a SL scheme is derived for advection at a constant velocity. The second step extends this result to any general velocity field. Finally, the third step deals with the source term, represented by $g(x,t)$ in (1.1). Each of the three steps highlights one of the three components required in a SL scheme for the full advection problem. These are interpolation, trajectory tracing and time discretisation.

1.2.1 Homogeneous Constant Velocity Advection

Setting $a(x,t) \equiv a$ (constant) the advection equation to be solved is

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0. \tag{1.2}$$

The analytical solution of this equation may be found using the method of characteristics, from which we obtain a system of two ordinary differential equations (ODE's):

$$\frac{dx}{dt} = a \tag{1.3}$$

$$\frac{du}{dt} = 0, \tag{1.4}$$

where there are now two object functions to be found, $x(t)$ and $u(x(t),t)$. Equation (1.3) describes the trajectory of a particle moving with the advecting velocity. If this particle

is at the point $x = x_0$ when $t = 0$, then the solution of this equation is

$$x(t) = x_0 + at, \quad t > 0.$$

The equation (1.4) for the advected quantity u has the solution

$$u(x(t), t) = \text{constant},$$

which states that u remains constant along a characteristic. Combining the solution of the two ODE's, we see that u satisfies

$$u(x_0 + at, t) = u(x_0, 0), \quad t > 0.$$

From this we obtain the solution of the linear advection problem (1.2):

$$u(x, t) = u_0(x - at).$$

We may follow the same basic outline as described above to obtain a numerical scheme for solving the advection problem. Doing so results in a semi-Lagrangian scheme for (1.2), which we next describe. This scheme introduces the first component required in the construction of a SL scheme for the more general problem, (1.1).

Step 1: Interpolation

The scheme uses a finite difference (grid point) representation of the solution. If the grid has space and time mesh Δx and Δt respectively, then u_j^n approximates the solution $u(x_j, t_n)$, where $x_j = j\Delta x$ and $t_n = n\Delta t$. To construct a semi-Lagrangian scheme we begin by assuming the solution is known at time $t = t_n$, and make a forward time step using the method of characteristics. That is, we are given the function $u(x, t_n)$ as initial condition and we wish to find the finite difference solution, u_j^{n+1}, at time t_{n+1} at each of the grid point x_j. Figure 1.2 shows the particular characteristic which passes through the grid-point $x = x_j$ at time t_{n+1}. This characteristic is a straight line with gradient $1/a$, as are all the characteristics for this problem. Extending the characteristic back to time t_n it passes through a point x_d at that time. This is the departure point for the particle

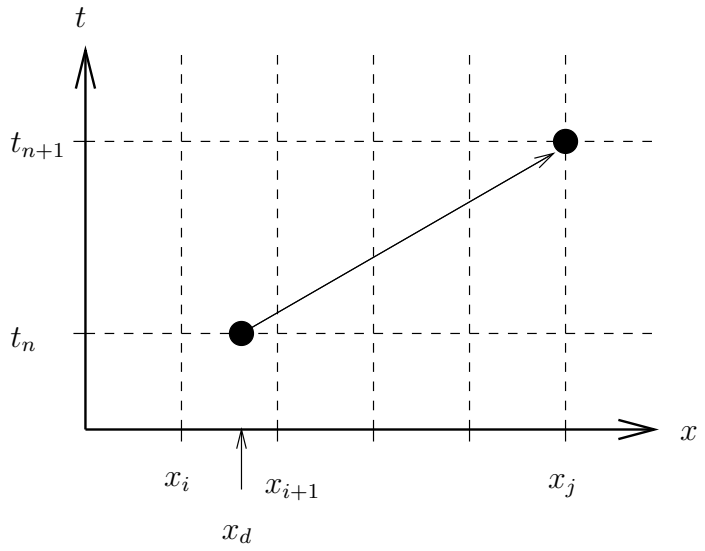

Figure 1.2: Basic SL method

which, starting at time t_n, arrives at the grid-point x_j at the later time t_{n+1}. How this is done for a general velocity field is considered in the next section.

To continue with the development of the SL scheme, we next make use of the analytical solution which states that u remains constant on any characteristic. For more general advection problems it is at this point where suitable time discretisations must be considered. These are discussed in section 1.2.3 and Chapter 11. For now, we use this property of characteristics to obtain

$$u_j^{n+1} = u_d^n, \tag{1.5}$$

where u_d^n represents the value of u at the departure point. This equation provides the finite difference solution at the new time-level, from the data at the old level. There are now two possibilities. First, if x_d happens to coincide with a grid-point, say $x_d = x_i$, then (1.5) provides the scheme

$$u_j^{n+1} = u_i^n.$$

Generally the departure points will not be coincident with grid-points. This is where interpolation is required. For this purpose polynomial interpolation may be used. For instance, if x_d lies between grid-points x_i and x_{i+1}, then linear interpolation of the finite

difference data at these points provides

$$u_d^n = \left(\frac{x_d - x_i}{x_{i+1} - x_i}\right) u_{i+1}^n + \left(\frac{x_{i+1} - x_d}{x_{i+1} - x_i}\right) u_i^n,$$

where u_d^n is now only an approximation to $u(x_d, t_n)$. Using once again the constancy of u along characteristics, now coupled with the above interpolation, equation (1.5) provides a numerical scheme for obtaining the finite difference solution at time t_{n+1} from the finite difference data at t_n. This scheme has only first order accuracy. A more useful, higher order scheme may be obtained by using a higher order of interpolation. The subject of interpolation will be considered in Chapters 2 and 5. In the next section we continue with the development of the semi-Lagrangian method, towards a scheme applicable to the full advection problem, as outlined at the beginning of section 1.2.

1.2.2 Non-constant Velocity Advection

The semi-Lagrangian scheme outlined in the previous section solved the homogeneous advection equation in two stages. First, for each grid-point the associated departure point must be found. Then the existing finite difference data are interpolated to these departure points. And hence, by the method of characteristics, the new solution is found. Finding the departure points when the advecting velocity is a constant is a particularly simple matter. To be able to do this for any given velocity field, $a(x, t)$, a numerical approach is required for finding departure points.

Step 2: Trajectory Tracing

A method which has found favour for use in SL methods is the implicit mid-point rule [55]. Our goal is to solve the trajectory equation (1.3),

$$\frac{dx}{dt} = a(x, t),$$

subject to the condition $x(t_{n+1}) = x_j$, in order to find $x(t_n)$. The point $x(t_n)$ is then the departure point x_d corresponding to the grid-point x_j.

The implicit mid-point rule is based on a discretisation of the trajectory equation which is centred both in time and space, giving second order time accuracy. The method may be derived by making the assumption that the velocity remains fixed at its mid-step value, during each time-step. This results in an approximation where each trajectory is linear. The mid-point of such a trajectory is simply the mean of the positions of its end points. Using these approximations and a two-level discretisation of the derivative term, we arrive at the following discretisation of (1.3):

$$\frac{x_j - x_d}{\Delta t} = a(x_m, t_n + \Delta t/2), \tag{1.6}$$

where $x_m = \dfrac{1}{2}(x_j + x_d)$. This implicit equation may be solved for x_d using fixed-point iteration. If the displacement along a trajectory is defined as

$$\alpha = x_j - x_d,$$

then (1.6) may be re-written as $\alpha = \Delta t\ a(x_j - \alpha/2, t_n + \Delta t/2)$. From this we obtain the fixed point iteration

$$\alpha^{[r+1]} = \Delta t\ a\left(x_j - \frac{1}{2}\alpha^{[r]}, t_n + \frac{1}{2}\Delta t\right).$$

Typically only 2 or 3 iterations are used to obtain sufficient accuracy in practice. Once the displacement α has been found in this way, the departure point is given by $x_d = x_j - \alpha$.

Within a full fluid simulation model the flow velocity is one of the prognostic variables. (Prognostic variables in a numerical model are those which the integration solves for; diagnostic variables are functions of prognostic variables, which may be evaluated directly from the current model fields at each time step. As such only its representation on the finite difference grid is available. Furthermore, the mid-point of the trajectory, at which the velocity is to be evaluated, will not generally coincide with a grid-point. Interpolation of the spatially distributed velocity data is therefore required. Similarly we must also expect that velocity data is available only for the model time-levels up to the most current. That is, for times $t = 0, t_1, t_2, \ldots, t_n$. Yet our discretisation requires the velocity at the forward time $t_{n+1/2} = t_n + \frac{1}{2}\Delta t$. Extrapolation of the existing data is therefore necessary.

Simple extrapolation formulae may be obtained by considering Taylor series expansions. We mention here the two most practical formulae obtained in this way:

- 2nd order accurate: $a^{n+1/2} = \frac{1}{2}(3a^n - a^{n-1}) + \mathrm{O}(\Delta t^2)$
- 3rd order accurate: $a^{n+1/2} = \frac{1}{8}(15a^n - 10a^{n-1} + 3a^{n-2}) + \mathrm{O}(\Delta t^3)$.

Using interpolation in space and extrapolation in time we are able to evaluate the velocity values required by the fixed point iteration, and this completes the departure point method. The rest of the SL solution of the homogeneous advection equation follows as for the constant velocity case.

So far we have only considered homogeneous advection, for which we are able to exploit the method of characteristics exactly. Next we consider how SL schemes accommodate source terms.

1.2.3 Source Terms

With the presence of a source term in the advection problem (1.1) the value of u is no longer constant along trajectories. The evolution equation which u now satisfies along trajectories is

$$\frac{du}{dt} = g(x(t), t).$$

The time discretisation of this equation forms the third aspect of the general SL scheme.

Step 3: Time Discretisation

To discretise this equation we already have available the positions of departure points, from solving the trajectory equation as described in the previous section. This allows us to make a two-level discretisation of the form

$$\frac{u_j^{n+1} - u_d^n}{\Delta t} = (1-\theta)g_d^n + \theta g_j^{n+1}. \tag{1.7}$$

In this formula $u_d^n \approx u(x_d, t_n)$ and similarly for g_d^n; θ is an implicitness parameter, with $\theta = 1$ corresponding to a fully implicit discretisation and $\theta = 0$ fully explicit. Other

values of θ provide semi-implicit discretisations. Since these schemes only involve two time-levels, they give results which are at best second order accurate. Greater accuracy may be obtained by increasing the number of time-levels used. Such schemes are discussed in Chapter 11.

A solution procedure is required for obtaining u_j^{n+1} from (1.7). This will depend on the nature of the source terms. Explicit terms are evaluated by interpolation of the existing finite difference data. Implicit terms, however, may involve other fluid variables. Consequently there will be a system of discretisations of the form (1.7), one for each fluid variable. These together with whatever other equations are required to represent the fluid system are to be solved simultaneously for all fluid variables. This is the case, for instance, when solving the shallow water equations, as discussed in Chapter 8.

Interpolation is a procedure for increasing the extent of a given amount of data. This can only be achieved by making assumptions about the quantity represented by this data. For this reason it is not surprising that the quality of any SL scheme depends strongly on the chosen interpolation method. In the following two chapters we examine the development of interpolation methods suitable for SL advection.

1.3 Nonlinear Advection

Semi-Lagrangian methods are not restricted solely to linear advection problems. Central to any model of fluid flow is a nonlinear advection equation representing the transport of momentum.

The appropriate momentum equation in atmospheric simulation is the Navier-Stokes equation. Models which are used in practice are built around some approximation of this equation. In addition to a momentum equation, further equations describing continuity of mass, thermodynamic state and transport of fluid properties are required in a complete

model. The non-hydrostatic model of Tapp and White [58] uses the equation set

$$\begin{aligned}
\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}.\boldsymbol{\nabla})\boldsymbol{u} &= \boldsymbol{\mathcal{F}} + \boldsymbol{g} - \frac{1}{\rho}\boldsymbol{\nabla} p && \text{momentum} \\
\frac{\partial \rho}{\partial t} + \boldsymbol{u}.\boldsymbol{\nabla}\rho &= -\rho\boldsymbol{\nabla}.\boldsymbol{u} && \text{continuity of mass} \\
\frac{\partial c_p T}{\partial t} + \boldsymbol{u}.\boldsymbol{\nabla}(c_p T) &= Q && \text{conservation of heat} \\
\rho &= \frac{p\mathcal{M}_r}{RT} && \text{equation of state}
\end{aligned}$$

where the model variables are

$\boldsymbol{u}$	velocity
ρ	density
p	pressure
T	temperature

and the other parameters are

$\boldsymbol{g}$	gravity
$\boldsymbol{\mathcal{F}}$	friction
c_p	heat capacity
Q	heat source terms
$\mathcal{M}_r$	relative molecular weight of air
R	gas constant.

The above system of equations is written in Eulerian form, where $\boldsymbol{x}$ and t are the independent variables. In Lagrangian coordinates $\boldsymbol{a}$ and t are the independent variables,

where $\boldsymbol{a}$ is a label for individual particles. A suitable label is provided by a particle's position at some reference time, as that uniquely specifies a particle for all subsequent times. In order to distinguish time derivatives in the Lagrangian coordinate system from those in the Eulerian system, the notation $\frac{D}{Dt}$ is often used in place of $\frac{d}{dt}$. Since the transformation between the two coordinate systems is simply translation with the fluid velocity $\boldsymbol{u}$, the following identity holds

$$\frac{D}{Dt} \equiv \frac{\partial}{\partial t} + \boldsymbol{u}.\boldsymbol{\nabla}.$$

In the Lagrangian system the particle labeled $\boldsymbol{a}$ has position $\boldsymbol{x}(\boldsymbol{a}, t)$ at time t. Hence particle position is a dependent variable of the Lagrangian model in addition to those listed for the Eulerian model. As such it must also have an evolution equation. This is just the trajectory equation, that states that each fluid particle moves with the local fluid velocity:

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{u}(\boldsymbol{x}(t), t).$$

The other equations of the flow are

$$\begin{aligned} \frac{D\boldsymbol{u}}{Dt} &= \boldsymbol{\mathcal{F}} + \boldsymbol{g} - \frac{1}{\rho}\boldsymbol{\nabla} p \\ \frac{D\rho}{Dt} &= -\rho\boldsymbol{\nabla}.\boldsymbol{u} \\ c_p \frac{DT}{Dt} &= Q, \end{aligned}$$

and the equation of state is as before. When the SL method is applied to such a system of equations, the trajectory equation is first used to determine departure points. Then the remaining evolution equations are dealt with through suitable time-discretisations and spatial interpolations as described in section 1.2.

In this thesis our concern is with the numerical simulation of advection, which is a purely dynamical process. This leads us to consider equation sets which exclude all physical processes, other than the dynamics associated with advection. The shallow-

water equations are just such a set, and take the form

$$\begin{aligned}
\frac{Du}{Dt} &= -g\frac{\partial h}{\partial x} \\
\frac{Dv}{Dt} &= -g\frac{\partial h}{\partial y} \\
\frac{Dh}{Dt} &= -h(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}).
\end{aligned}$$

The three model variables are u and v, the two-dimensional velocity components, and h the fluid depth; g is the gravitational constant. This two-dimensional model of fluid flow is derived from a vertical integration of the momentum and continuity equations of the Tapp and White model, neglecting friction and assuming hydrostatic balance. In Chapter 8 we examine how the semi-Lagrangian method may be applied to these equations in spherical geometry.

Chapter 2

Interpolation: One-Dimensional Schemes

2.1 Introduction

In this chapter we begin the description of how multi-dimensional semi-Lagrangian methods are constructed. Any practical weather prediction simulation requires the modelling of fully three-dimensional flows. The three aspects of the semi-Lagrangian method outlined in the first chapter each apply in any number of space dimensions. No new theoretical considerations arise for the SL method as a whole. The task is one of finding suitable algorithms for each of the three stages: trajectory calculation, interpolation and time discretisation.

The algorithms presented in Chapter 1 for time discretisation and trajectory calculation may be extended for use in more than one dimension. This involves no further complication other than the use of vector-valued quantities for velocity and position. Unfortunately the same does not hold true for the interpolation stage of the scheme. Interpolation of data distributed in several dimensions introduces complications not present in one dimension. For this reason the largest part of our current task is to identify and assess methods for multi-variate interpolation. We begin by examining the options for

interpolation in one dimension.

2.2 Univariate Interpolation

Multivariate interpolation is typically carried out by combining several univariate interpolations. Tensor product interpolation provides the most familiar example of this approach, for which details are given in Chapter 5. A further method of this kind has been developed recently by Purser and Lesley [39], [40], [23]. In what they call "cascade interpolation" data is transferred efficiently from one grid to another. Such an emphasis on transferring whole representations of multivariate functions makes this method particularly suitable for SL. Cascade interpolation is also described in Chapter 5. Before considering these methods we must first examine the options for interpolation in one-dimension, from which multi-dimensional methods In choosing an interpolation method, the ultimate end for which it is to be used dictates the constraints to be satisfied by the interpolant. For instance, if we know that the data to be interpolated represent a quantity which is everywhere non-negative, such as atmospheric water content, then there will be an advantage in finding an interpolant which respects this property. This leads us to consider how the degrees of freedom possessed by an interpolant may be used to satisfy some externally imposed constraint.

We assume there is a set of data,

$$\mathcal{S} = \{(x_i, f_i) : i = 0, \cdots, N\},$$

for which an interpolating function, $p(x)$, is required. Throughout this chapter we take f_i to be scalar. For univariate interpolation x_i is also scalar. In Chapter 5 multi-variate interpolation is discussed, and there x_i is vector-valued. Continuing with the univariate case, we may impose an ordering on the labelling of the interpolation knots, or grid-points $\{x_i\}$. If the data and its interpolant are defined on the interval $[a, b]$ then we assume the ordering

$$a = x_0 < x_1 < \cdots < x_N = b.$$

The interpolant $p(x)$ is a function with a finite number of degrees of freedom. If p has M degrees of freedom, then we must impose M conditions on p for it to be determined completely. Since $p(x)$ is required to be interpolant for the set $\mathcal{S}$ it must be consistent with $\mathcal{S}$. That is, p must be such that $M \geq N$, and it must satisfy the interpolation conditions

$$p(x_i) = f_i \quad i = 0, \cdots, N \tag{2.1}$$

If it is the case that $M = N+1$ then (2.1) determine the interpolant completely. However, only piecewise constant functions fit this description. For most practical applications smoother interpolation is required, and we are led to consider interpolants for which $M > N + 1$. In addition to the $N + 1$ interpolation conditions, a further $M - N - 1$ conditions must be specified for such an interpolant, and we are free to choose what these should be. For example, consider piecewise interpolation using cubic polynomials.

2.2.1 Example: Piecewise Cubic Interpolation

On each sub-interval $[x_i, x_{i+1}]$ the interpolant is a cubic polynomial,

$$p(x) = p_i(x) = A_i x^3 + B_i x^2 + C_i x + D_i.$$

Since there are N such sub-intervals and each cubic has four degrees of freedom the interpolant has $M = 4N$ degrees of freedom. We can match conditions to these degrees of freedom in various ways. Piecewise cubic Lagrange interpolation is obtained by requiring $p_i(x)$ to interpolate the four points $\{(x_{i+k}, f_{i+k}) : k = -1, 2\}$, which surround its interval of definition. To ensure the interpolants in the first and last sub-intervals also satisfy four such conditions, two extra data points are required, one beyond each end of the interval $[a, b]$.

It is not necessary for all four degrees of freedom in each cubic to be assigned through Lagrange interpolation. In order for the interpolant to satisfy the conditions (2.1) and be continuous, each cubic $p_i(x)$ need only interpolate the data points at the two ends of its sub-interval, $[x_i, x_{i+1}]$. This accounts for $2N$ degrees of freedom, half the number which

must be specified. If we require the interpolant to be continuous in its first and second derivatives, so guaranteeing a large degree of smoothness, we obtain a further $2(N-1)$ conditions:

$$\left.\begin{array}{rcl} p_i'(x_i) & = & p_{i+1}'(x_i) \\ p_i''(x_i) & = & p_{i+1}''(x_i) \end{array}\right\} \; i = 1, \cdots, N-1.$$

Interpolants having global properties of smoothness are termed splines.

In total there are now $4N-2$ conditions, leaving 2 final conditions to be specified in order to completely determine the interpolant. Again it is at the boundaries of the interval where extra information is required. The specifics of the particular application for which the interpolation is to be used often suggest appropriate boundary conditions. We mention only the 'natural' boundary conditions, for which the interpolant has zero slope at either end of the interval:

$$p_0'(a) = 0$$

$$p_N'(b) = 0$$

— the final two conditions required to determine $p(x)$.

Finally, a third option would be to begin again with the $2N$ conditions required for continuous interpolation. But now, rather than impose smoothness conditions, we could require derivative values to be interpolated at the ends of each sub-interval. This is Hermite interpolation which, together with Lagrange interpolation, will be examined in detail later in this chapter.

2.3 A First Order Scheme

The accuracy of an interpolant is assessed by introducing a notional smooth function $f(x)$ underlying the data set $\mathcal{S}$. That is, $f(x_i) = f_i$ for all i. Accuracy is then described in terms of the order of convergence of $p \to f$ as $N \to \infty$, in some suitable norm. For

piecewise polynomial interpolation of order n, using the maximum norm, the order of accuracy is $O(h^{n+1})$ where $h = \max_i(x_{i+1} - x_i)$.

Although formal accuracy is an important consideration, it is not a sufficient criterion for selecting an interpolant for SL advection. As indicated earlier, it may often be desirable to build certain known properties of the data, such as positivity, into the interpolant. It is apparent that the only way to reliably compare different interpolants when applied to SL, is to apply each of them to a suitably chosen advection test-problem. Our needs are met by the following simple test-problem:

$$\begin{aligned} \frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} &= 0 \qquad\qquad 0 > t, \quad -\infty < x < \infty \\ u(x,0) &= u_0(x) \\ a &= \text{const} > 0. \end{aligned} \tag{2.2}$$

The initial profile, $u_0(x)$, must be specified. Analytically the solution for this problem consists of the initial profile moving to the right, without change of shape, at constant speed a.

This problem may be solved numerically by making a regular discretisation of the domain, using time and space mesh lengths Δt and Δx respectively. The space and time locations of the grid points are denoted x_j and t_n respectively, where $j, n \in \mathbb{Z}$. A SL discretisation requires the governing PDE to be written in Lagrangian form,

$$\frac{Du}{Dt} = 0.$$

A two time-level time discretisation of this leads to an explicit semi-discrete scheme,

$$u_j^{n+1} = u^n(x_d) \tag{2.3}$$

where $u_j^n = u(x_j, t_n)$, $u^n(x) = u(x, t_n)$; and x_d is defined in the following. To complete the SL scheme, the trajectory equation,

$$\frac{dx}{dt} = a,$$

must be solved to find the departure point x_d corresponding to each grid point x_j. Since the velocity is constant this is a simple matter, and the analytical solution may be used:

$$x_d = x_j - a\Delta t. \tag{2.4}$$

It is important to note that, for this particular test problem, the time discretisation behaves exactly as the analytical solution: along particle trajectories, u remains constant. The only part of the SL scheme in this case, therefore, which is not exact is the interpolation required to approximate $u^n(x_d)$. Smolarkiewicz and Grell [52] have provided a formal framework in which numerical advection and interpolation are seen to be equivalent operations. We shall return to this in Chapter 4.

Before applying piecewise linear interpolation to the test-problem, we first non-dimensionalise the displacement represented in (2.4). For this problem the dimensionless displacement is identical to the dimensionless velocity, or Courant number,

$$\nu = a\frac{\Delta t}{\Delta x} = \frac{x_j - x_d}{\Delta x}.$$

The integer part of ν represents the number of whole cells traversed in one time-step. If ν happens to be an integer then we may use $u^n(x_d) = u^n_{j-\nu}$, and the resulting fully discrete scheme agrees with the exact solution. In general, however, the fractional part of ν, which we define by $\xi = \nu - \text{int}(\nu)$, will be non-zero. This quantity (ξ) is a convenient form for the one independent variable of the scheme. It takes values in the range $0 \leq \xi < 1$.

Using the above definitions, we see that if x_d lies within the kth grid-cell, that is if $x_{k-1} < x_d \leq x_k$, then linear interpolation provides

$$u^n(x_d) \approx (1-\xi)u^n_k + \xi u^n_{k-1}, \tag{2.5}$$

since ξ increases from zero at $x = x_k$ and approaches unity as x decreases towards x_{k-1}. The result of applying SL to (2.2) using linear interpolation is shown in figure 2.1. In figure 2.1(a) the initial data is shown; after one hundred time steps, figure 2.1(b), the wave profile has advanced, though it lags behind the analytical solution shown by the dotted line. The peak height of the numerically advected wave has decreased to around forty

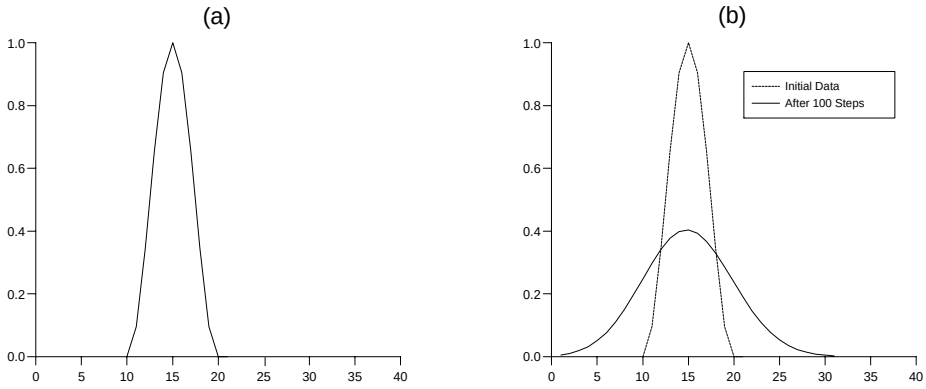

Figure 2.1: First order SL scheme, using linear interpolation

percent of the initial height, while its spatial extent has correspondingly spread. Despite these poor results, the scheme does have several significant properties: throughout all time steps the initially positive data remains positive; no new local extrema have been introduced; and lastly it is found that the area beneath the profile is the same at all time steps. This last property corresponds to the conservation of mass in the advection process. More precisely, if we define the numerically advected mass after n time steps to be

$$M_n = \sum_j u_j^n \, \Delta x,$$

then it is a simple matter to verify that this is a constant of the motion:

$$\begin{aligned} (2.5) \Longrightarrow \sum_j u_j^{n+1} \, \Delta x &= (1-\xi) \sum_j u_j^n \, \Delta x + \xi \sum_j u_{j-1}^n \, \Delta x \\ &= \sum_j u_j^n \, \Delta x. \end{aligned}$$

Hence $M_{n+1} = M_n$.

In the case where there are spatial boundaries, any difference between M_{n+1} and M_n is wholly accounted for by boundary processes. We shall treat the issue of mass conservation in greater depth in Chapter 6. Our aim for the present will be to find interpolants more accurate than linear interpolation, ideally possessing the properties of positivity, conservation of mass and monotonicity.

By increasing the order of the interpolating polynomial we are able to gain higher

accuracy. We shall continue to use a local piecewise approach, first as it may be argued that this reflects the local nature of advection; and secondly to avoid the increasingly oscillatory, non-convergent behaviour to which polynomial interpolation is prone as the order is increased. Since accuracy is the prime goal, the possibility of directional bias is avoided by using only centred interpolation stencils. That is, we require the interpolant to use data which is symmetrically distributed about the point of interpolation. As a consequence of this we are lead to consider odd degree polynomials. Cubic and quintic polynomials therefore provide suitable classes from which to select interpolants.

2.4 Higher Order Schemes

In order to gauge how well a particular interpolant performs in a SL scheme, it is not sufficient to consider only its order of accuracy. Different interpolants with the same order of accuracy need not lead to identical results in a SL fluid simulation. A detailed analysis of numerical errors is required. The analysis we employ is in the context of the test problem (2.2). The analytical solution of this problem is a traveling wave, and consequently any analysis of numerical errors ought to reflect this fact.

We have so far mentioned a number of quantities relating to bulk motion, namely positivity, monotonicity and mass conservation, in which minimisation of error is desirable. To this list we now add the wave properties of phase velocity and dispersion.

The first high order interpolants we examine will be linear functions in the interpolated data. To analyze the SL schemes formed by these interpolants we therefore employ the machinery of Fourier analysis. However, as may be anticipated with reference to any text on numerical analysis [16], these schemes suffer from severe loss of positivity and monotonicity under certain conditions. Only by allowing the interpolant to be nonlinear are these problems to be avoided. It is therefore necessary to find a more general approach to analyzing errors in numerically propagated waves, which does not rely on linearity.

2.4.1 Linear Schemes

Lagrange Interpolation

A unique polynomial of degree n can always be fitted to $n+1$ data points, provided all the points are at distinct spatial locations. Many algorithms exist for finding such interpolating polynomials. Lagrange interpolation constructs the polynomial from basis functions, $l_i(x)$. Each basis function is a degree n polynomial, satisfying the interpolation conditions,

$$l_i(x_j) = \begin{cases} 1 & i = j \\ \\ 0 & i = 1, \cdots, n+1; \quad i \neq j \end{cases}$$

The unique polynomials satisfying these conditions may be expressed in the form

$$l_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n+1} \frac{x - x_j}{x_i - x_j}.$$

With the basis functions defined in this way, the interpolating polynomial for $n+1$ points of the data set $\mathcal{S}$ is

$$p(x) = \sum_{i=1}^{n+1} f_i \, l_i(x). \tag{2.6}$$

As indicated earlier, our principal interest lies with cubic and quintic interpolating polynomials. It is instructive to examine the form the basis functions take for cubic interpolation of regularly spaced data. Figure 2.2 shows the basis function which takes value one at the first of the four interpolation nodes, and figure 2.3 shows the second basis function. The other two basis functions are merely reflections of the first two. A standard theorem of approximation theory (see, for instance, [36]) states that the interpolating polynomial is most accurate between the two central nodes. This provides some justification for using a centred stencil for interpolation in a SL scheme. In addition, von Neumann stability analysis shows that off-centred stencils are liable to form unstable schemes. In figure 2.2 we see that not all of the cubic basis functions are positive in the central interval. This leads us to expect that, for certain arrangements of positive data, the interpolating polynomial may be non-positive. Figure 2.4 shows the failure of positivity for advection of

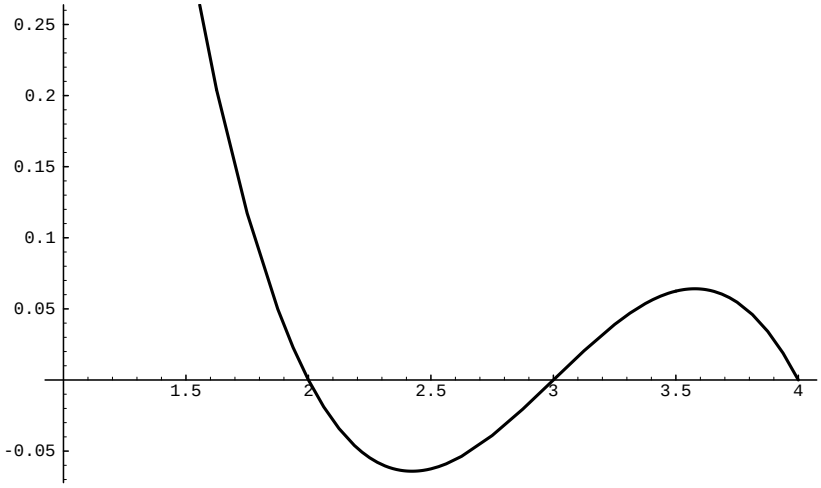

Figure 2.2: Lagrange basis function $l_1(x)$ for cubic interpolation

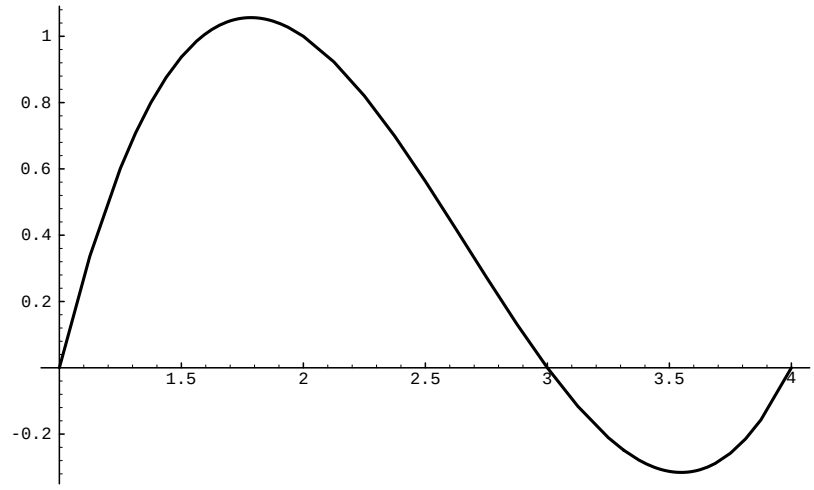

Figure 2.3: Lagrange basis function $l_2(x)$ for cubic interpolation

a square pulse, using cubic interpolation; equally apparent is the loss of monotonicity in the numerical solution.

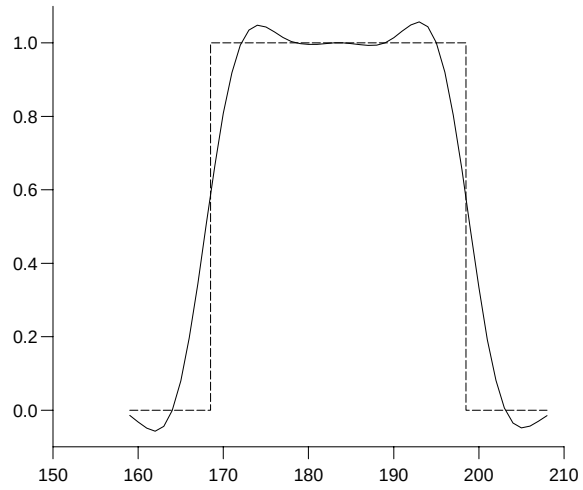

Figure 2.4: High order SL using cubic interpolation

Divided Difference Interpolation

In practice, interpolating polynomials are often constructed using the method of divided differences. This approach allows the order of the interpolant to be increased through successively adding further terms to the interpolant. Divided differences for the data set $\mathcal{S}$ are defined in the following way:

$$\begin{aligned} f[x_i] &= f_i \\ f[x_i, x_{i+1}, \cdots, x_{i+j}] &= \frac{f[x_{i+1}, \cdots, x_{i+j}] - f[x_i, \cdots, x_{i+j-1}]}{x_{i+j} - x_i}. \end{aligned}$$

The degree n interpolating polynomial through the points $\{(x_{i+j}, f_{i+j}) : j = 0, \cdots, n\}$ is

$$\begin{aligned} p(x) = f[x_i] &+ (x - x_i) f[x_i, x_{i+1}] \\ &+ (x - x_i)(x - x_{i+1}) f[x_i, x_{i+1}, x_{i+2}] \\ &+ \cdots + (x - x_i) \cdots (x - x_{i+p}) f[x_i, \cdots, x_{i+n}]. \end{aligned}$$

This polynomial is identical to the Lagrange form (2.6), merely being a reordering of its terms, and is known as Newton's interpolating formula. The importance of this algorithm for numerical advection schemes is that a slight nonlinear modification yields an interpolant which substantially reduces spurious oscillations. This is the method of essentially non-oscillatory interpolation (ENO), which will be examined in section 4.1.

Hermite Interpolation

The interpolating polynomials discussed so far may be used to approximate a function, $f(x)$, by matching the function at $n+1$ points x_{i+j}, $j = 0, \cdots, n$. That is, a degree n polynomial is obtained from the interpolation conditions

$$p(x_{i+j}) = f(x_{i+j}), \quad j = 0, \cdots, n.$$

If we wish to increase the order of accuracy of the interpolation, this is achieved by increasing the number of points to be interpolated. This necessarily involves using data progressively further and further from the region where the interpolation is to be evaluated.

Hermite and Hermite-Birkhoff interpolation, alternatively, offer the possibility of increasing the accuracy while retaining a compact stencil. The extra interpolation conditions required to achieve this are obtained by requiring the first and possibly higher derivatives of $f(x)$ to be matched at some, or all, of the nodes x_{i+j}. In Hermite interpolation the number of derivatives to be matched is the same at each node. Hermite-Birkhoff interpolation allows the number of conditions to differ from node to node.

Hermite cubic interpolation offers a combination of simplicity and accuracy which makes it particularly suitable for SL schemes. This method requires values of the function and its first derivative to be given at two nodes, x_i and x_{i+1}. The four interpolation conditions are:

$$p(x_i) = f_i, \quad p(x_{i+1}) = f_{i+1};$$

$$p'(x_i) = d_i, \quad p'(x_{i+1}) = d_{i+1},$$

where $f_i = f(x_i)$ and $d_i = f'(x_i)$. The polynomial may be constructed from basis functions, in much the same way as for Lagrange interpolation [15]:

$$p(x) = f_i\, H_1(x) + f_{i+1}\, H_2(x) + d_i\, H_3(x) + d_{i+1}\, H_4(x),$$

where

$$\begin{aligned}
H_1(x) &= \frac{(x_{i+1}-x)^2}{h^2} + 2\frac{(x-x_i)(x_{i+1}-x)^2}{h^3}, \\
H_2(x) &= \frac{(x-x_i)^2}{h^2} + 2\frac{(x_{i+1}-x)(x-x_i)^2}{h^3}, \\
H_3(x) &= \frac{(x-x_i)(x_{i+1}-x)^2}{h^2}, \\
H_4(x) &= -\frac{(x_{i+1}-x)(x-x_i)^2}{h^2},
\end{aligned}$$

and $h = x_{i+1} - x_i$.

To obtain a complete SL scheme using Hermite interpolation we require some means of estimating the derivative values, d_i and d_{i+1}. At this point we appear to be no further forward than with Lagrange interpolation, since to approximate derivatives to an accuracy consistent with cubic interpolation a broad stencil is necessary. However, the value of Hermite interpolation is the flexibility it allows through the selection of derivative estimates. Rasch and Williamson [43] show how this flexibility may be used to obtain SL advection schemes which conserve monotonicity. The technique they use will be presented in section 4.2.

Derivative estimates may either be linear or non-linear in the data $\{u_i\}$. Linear estimates are perhaps the most suitable for SL schemes, since they find a simple and natural expression in terms of the discrete piecewise gradient of the data. The discrete gradient on the cell $[x_i, x_{i+1}]$ is

$$\Delta_i = \frac{u_{i+1} - u_i}{x_{i+1} - x_i}.$$

By taking weighted averages of discrete gradients, high order approximations to the first derivative may be obtained. Hermite interpolation requires values of derivatives at grid points; the most accurate estimates for this purpose are obtained from centred approximations, Table 2.1.

Method	Formula	Accuracy
Arithmetic Mean	$d_i = \frac{\Delta_{i-1} + \Delta_i}{2}$	$\mathrm{O}(h^2)$
Hyman [14]	$d_i = \frac{-\Delta_{i-2} + 7\Delta_{i-1} + 7\Delta_i - \Delta_{i+1}}{12}$	$\mathrm{O}(h^4)$
Priestley [37]	$d_i = \frac{-3\Delta_{i-2} + 19\Delta_{i-1} + 19\Delta_i - 3\Delta_{i+1}}{32}$	$\mathrm{O}(h^4)$

Table 2.1: Centred derivative estimates, employing linear combinations of discrete slopes.

As an example of a non-linear derivative estimate, we mention the Akima method [1]:

$$d_i = \frac{a\Delta_{i-1} + b\Delta_i}{a + b}$$

where $a = |\Delta_{i+1} - \Delta_i|$ and $b = |\Delta_{i-1} - \Delta_{i-2}|$. In regions where this estimate fails, it may be replaced by its nearest linear equivalent, the arithmetic mean.

2.4.2 Linear Analysis

The techniques of Fourier analysis provide a good deal of useful information about numerical advection schemes. This form of analysis describes how well a given scheme represents the amplitude and phase speed of an infinite, single frequency, progressive harmonic wave.

Given the initial data

$$u_0(x) = e^{ikx},$$

where k is the wavenumber, the analytic solution of the advection problem (2.2) is

$$u(x,t) = e^{ik(x-at)}.$$

From this we see that solutions separated by a time interval Δt are related by

$$u(x, t + \Delta t) = e^{-ika\Delta t} u(x,t). \tag{2.7}$$

For general initial data the solutions at different times are related through the evolution operator, E:

$$\begin{aligned} u(x, t + \Delta t) &= E(\Delta t)\, u(x, t) \\ &\equiv u(x - a\Delta t, t). \end{aligned}$$

The factor $\exp(-ika\Delta t)$ in (2.7) is therefore the Fourier symbol of the evolution operator.

In order to apply Fourier analysis to a numerical advection scheme, its evolution operator must first be identified and corresponding Fourier symbol calculated. Differences in the modulus and argument between the numerical and analytic Fourier symbols represent errors in the amplitude and phase, respectively, of the numerically advected wave.

So far the numerical solution depends on four variables, namely the time-step Δt, the spatial mesh Δx, the wave speed, a, and the wavenumber k. From these four variables two dimensionless parameters may be formed, on which alone the solution depends. The first of these is the Courant number

$$\nu = \frac{a\Delta t}{\Delta x}.$$

This indicates the number of cells crossed by a trajectory in one time step. The wavenumber of the analytic solution is related to the wavelength, λ, through the relation

$$k = \frac{2\pi}{\lambda}.$$

From the ratio of the wavelength to the mesh, $\lambda/\Delta x$, we may define a dimensionless wavenumber:

$$\phi = \frac{2\pi}{\lambda/\Delta x} \equiv k\Delta x.$$

With these definitions, the analytic Fourier symbol (F) becomes

$$\begin{aligned} F &= e^{-ika\Delta t} = e^{-i\phi a\Delta t/\Delta x} \\ &= e^{-i\nu\phi}. \end{aligned}$$

To summarise, we have found that the analytical solution of (2.2) has the following properties:

- Evolution operator: $u(x, t + \Delta t) = E\, u(x, t) \equiv u(x - a\Delta t, t)$
- Fourier symbol: $F = e^{-i\nu\phi}$.

We next demonstrate how equivalent properties are found for a numerical scheme, using linear interpolation as an example. Let S be the spatial shift operator, defined by

$$S\, u_i = u_{i+1}.$$

In a SL scheme the finite difference stencil adapts to the trajectory of the flow. To represent this mechanism in the analysis, the Courant number is split into its integer and non-integer parts:

$$\nu = l + \alpha \quad 0 \leq \alpha < 1, \quad l \in \{0, 1, 2, \ldots\}.$$

The scheme may now be written in the form

$$\begin{aligned} u_i^{n+1} &= (1-\alpha)u_{i-l}^n + \alpha u_{i-l-1}^n \\ &= S^{-l}[(1-\alpha) + \alpha S^{-1}]\, u_i^n. \end{aligned}$$

Hence the numerical evolution operator is

$$E_1 = S^{-l}[(1-\alpha) + \alpha S^{-1}],$$

where the subscript denotes linear interpolation.

The Fourier symbol, F_1, for this scheme is found by considering the action of the evolution operator E_1 on the discrete wave function $\exp(ij\phi)$, where j is a spatial index:

$$F_1 = E_1 e^{ij\phi} = e^{-il\phi}[(1-\alpha) + \alpha e^{-i\phi}].$$

This differs from the amplification factor of the first order upwind scheme by the factor $\exp(-il\phi)$. Since this factor has modulus unity, the von Neumann stability analysis for this SL scheme is identical to that of the upwind scheme with Courant number in the range $[0, 1)$. It follows from this that the SL scheme with linear interpolation has unconditional

linear stability. Next considering the complex argument of F_1 we again see that the integer part of the Courant number is rendered exactly by the numerical scheme:

$$\begin{aligned}\arg(F_1) &= \arg[e^{-il\phi}(1-\alpha+\alpha e^{-i\phi})] \\ &= \arg(e^{-il\phi}) + \arg(1-\alpha+\alpha e^{-i\phi}) \\ &= -l\phi + \arg(1-\alpha+\alpha e^{-i\phi}).\end{aligned}$$

The last term in the above is an approximation relating to the fractional part of the Courant number:

$$\arg(1-\alpha+\alpha e^{-i\phi}) = -\alpha\phi + \mathrm{O}(\alpha\phi^3).$$

For any linear SL scheme on a regular grid the Fourier symbol has a factor $\exp(-il\phi)$, due to the shift in the stencil. Consequently, as demonstrated above, a complete analysis need only consider Courant numbers in the unit interval, $0 \leq \nu \leq 1$.

Once the Fourier symbol, F_s, of a scheme has been found, we may define two error measurements:

- Amplitude error: $\epsilon_a = \dfrac{|F_s|}{|F|} = |F_s|$
- Phase error: $\epsilon_p = \dfrac{\arg(F_s)}{\arg(F)} = -\dfrac{\arg(F_s)}{\nu\phi}$

Figure 2.5 shows the dependency of these errors on wavenumber for the SL scheme with linear interpolation; results for several representative values of the Courant number ("nu") are plotted.

For the next example we consider cubic Lagrange interpolation. Restricting the Courant number to the unit interval, this scheme takes the following form:

$$\begin{aligned}u_j^{n+1} &= -\frac{(1-\nu)\nu(1+\nu)}{6}u_{j-2}^n + \frac{(2-\nu)\nu(1+\nu)}{2}u_{j-1}^n \\ &\quad + \frac{(2-\nu)(1-\nu)(1+\nu)}{2}u_j^n - \frac{(2-\nu)(1-\nu)\nu}{6}u_{j+1}^n\end{aligned}$$

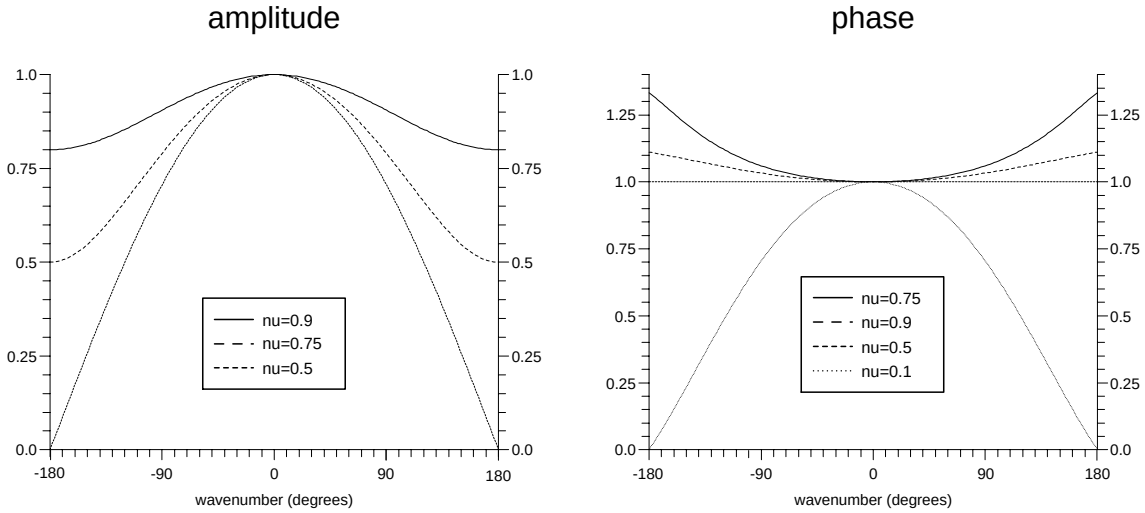

Figure 2.5: Fourier analysis of SL using linear interpolation

Setting $u_j^n = \exp(ij\phi)$, the scheme reduces to the evolution equation for a single Fourier mode,

$$u_j^{n+1} = F_3(\nu, \phi)e^{ij\phi},$$

where $F_3(\nu, \phi)$ is the Fourier symbol for the scheme. The amplitude and phase errors for the cubic scheme are shown in figure 2.6. Comparing these results with those for the linear interpolation scheme we find the following. First, the amplitude for the cubic scheme remains closer to the analytic value (one) over a wider range of wave-numbers than does the linear scheme. Both schemes show their most severe attenuation in amplitude for Courant numbers of one-half.

In terms of the phase error cubic interpolation performs better than linear, at all Courant numbers, for wave-numbers within $\pi/2$ (90 degrees) of zero. Beyond this point, where $|\phi| > \pi/2$, the two schemes have practically identical phase errors; only at low Courant numbers does the cubic scheme significantly improve on the linear. Since $\phi = k\Delta x = 2\pi\Delta x/\lambda$, a wavenumber $\phi = \pi/2$ corresponds to a wavelength of $\lambda = 4\Delta x$. Wavelengths of less than $4\Delta x$ are therefore subject to similar phase errors in SL schemes using either linear or cubic Lagrange interpolation.

Finally, we apply the linear analysis of this section to Hermite cubic interpolation,

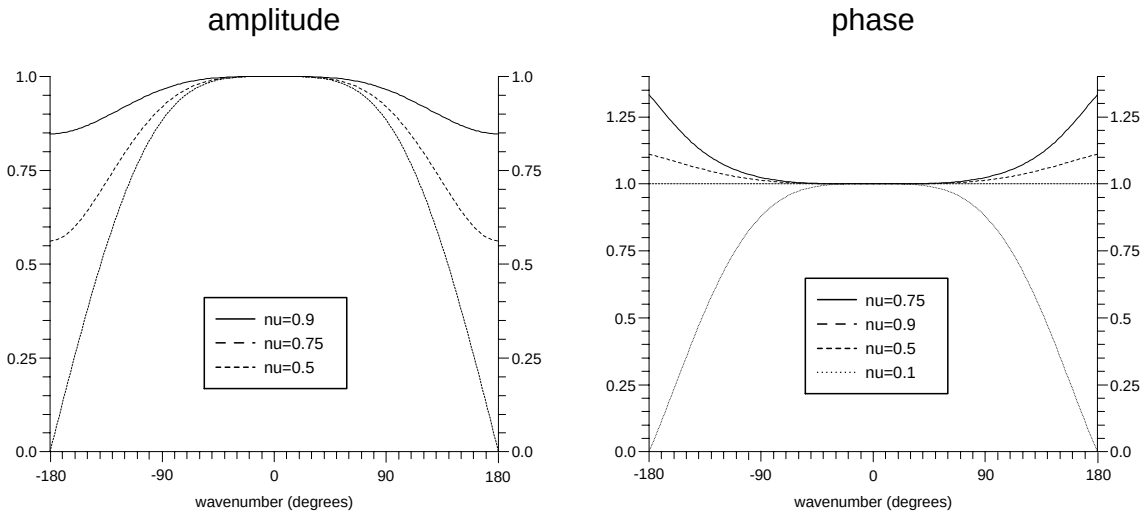

Figure 2.6: Fourier analysis of SL using cubic interpolation

using Priestley's method for estimating the derivative values [37]. Applying the definitions of section 2.4.1 to a regular grid of unit mesh, the scheme takes the form,

$$\begin{aligned} u_j^{n+1} &= \frac{3\nu^2(1-\nu)}{32}u_{j-3}^n - \frac{(3+19\nu)(1-\nu)\nu}{32}u_{j-2}^n \\ &\quad + \frac{(11-7\nu)\nu(1+3\nu)}{16}u_{j-1}^n + \frac{(1-\nu)(4-3\nu)(4+7\nu)}{16}u_j^n \\ &\quad - \frac{(1-\nu)\nu(22-19\nu)}{32}u_{j+1}^n + \frac{3(1-\nu)^2\nu}{32}u_{j+2}^n. \end{aligned}$$

The results of Fourier analysis are shown in figure 2.7. The amplitude error shows improvement over the cubic Lagrange scheme, while the phase errors for these two schemes are almost identical. It is interesting to note that these two schemes have the same order of accuracy, by which we mean they have the same convergence rate under mesh refinement.

All of the schemes described in this chapter are dissipative — they each attenuate advected waves. Only under two conditions are waves propagated without attenuation in these schemes. The first of these results from the "unit CFL property", which is a product of the interpolation conditions (2.1): the numerical solution is exact for integer Courant numbers. Second is the trivial case when the wavenumber is zero, that is, when both the analytic and numerical solutions are constant everywhere.

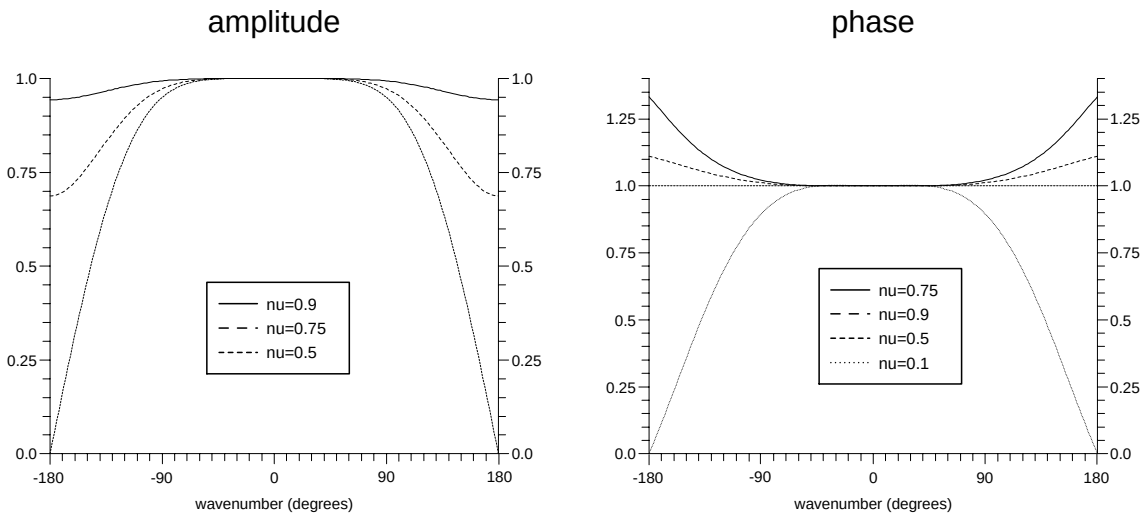

Figure 2.7: Fourier analysis of SL with Hermite cubic interpolation

From the Fourier analysis of the three example schemes presented here, we see that attenuation is dependent on wavenumber. The highest wave-numbers and, consequently, shortest wavelengths are damped the most. This behaviour is favourable in an advection scheme, where the shortest wavelengths are least well resolved by the grid, and are prone to being non-physical in a fluid simulation. Unfortunately it is often found in practice that the degree of damping inherent in these schemes is insufficient. Additional methods are required for selectively damping the non-physical, short-wave modes.

Chapter 3

Quadratic Interpolation

Introduction

Semi-Lagrangian schemes are generally constructed with polynomial interpolants of odd order. Linear interpolation gives insufficient accuracy for most purposes. Quintic interpolation requires considerable extra computational effort for diminishing returns, when compared with cubic interpolation. For this reason cubic interpolation is most often used. However, even-order polynomial interpolants are equally applicable to SL schemes as are odd order interpolants. One reason for not choosing an even-order interpolant is that they are the unique interpolating polynomials for an odd number of data points. Consequently, if an even-order unique interpolator is used then the interpolation stencil will have an odd number of points. This means that the interpolation is not centred and will have some directional bias. However, by relaxing the requirement of unique interpolation, even-order interpolants may be constructed using data at an even number of points, thus providing a centred stencil.

In this section we examine two forms of quadratic interpolation which use a four point stencil.

3.1 Centred Quadratic Interpolation

Consider a grid of regularly spaced points with mesh interval Δx :

$$x_j = j\Delta x \quad j = 1, \cdots, N. \tag{3.1}$$

Let f_j be data values on this grid. These data are to be interpolated to a departure point x_d, which has been calculated as part of a semi-Lagrangian advection step. Let x_k be the grid point immediately to the left of x_d,

$$x_k \leq x_d < x_{k+1}, \tag{3.2}$$

for some value of k. Let $q(x)$ be a piecewise quadratic interpolating polynomial for the data. And let $q_k(x)$ be the quadratic polynomial which interpolates the data for values of x in the range $[x_k, x_{k+1})$.

Define the interpolation parameter,

$$\xi = \frac{x_d - x_k}{x_{k+1} - x_k}. \tag{3.3}$$

A unique quadratic interpolation polynomial may be constructed by selecting three points at which the data are to be interpolated. Since our aim is to construct a viable advection algorithm, we are not free to select any three points. We should wish the scheme to behave appropriately under certain special circumstances. For instance, the scheme should not alter the initial data when time-stepping with a timestep set to zero. If the chosen interpolant does not interpolate the data at grid point x_k, this will not be the case. Further, if the interpolant is chosen to interpolate to grid point x_k, then the semi-Lagrangian scheme will give exact results for advection at constant integer Courant numbers. This is because departure points will lie precisely on grid points under such conditions, so that $x_d = x_k$. The data should also be interpolated at point x_{k+1}. If this were not the case, then the interpolant would behave differently in the limit as x_d approaches a grid point x_i, according to whether it approaches from the right or from the left.

From the above considerations we see that any interpolating polynomial, q_k, for use

in a semi-Lagrangian scheme, must necessarily satisfy only two interpolation conditions

$$q_k(x_k) = f_k \tag{3.4}$$

$$q_k(x_{k+1}) = f_{k+1}. \tag{3.5}$$

We are free to determine the remaining degrees of freedom of the polynomial in any way we choose. It is still necessary, though, to analyse the stability properties of the resulting SL scheme, since interpolation alone doesn't guarantee stability. We next consider three centred quadratic interpolants.

Interpolant 1: Mean of Left and Right Interpolants

For a quadratic polynomial which satisfies the interpolation conditions (3.4) and (3.5), there remains just one degree of freedom to be determined. There are two possibilities for a further interpolation condition which we might consider. Either the next data point to the left may be interpolated, giving

$$q_k^L(x_d) = -\frac{\xi(1-\xi)}{2} f_{k-1} + (1-\xi^2) f_k + \frac{\xi(1+\xi)}{2} f_{k+1}. \tag{3.6}$$

Or the next data point to the right may be used, for which case the interpolating polynomial is

$$q_k^R(x_d) = \frac{(1-\xi)(2-\xi)}{2} f_k + \xi(2-\xi) f_{k+1} - \frac{\xi(1-\xi)}{2} f_{k+2}. \tag{3.7}$$

For Courant numbers in the range between zero and one, the second of these interpolants corresponds to the Lax-Wendroff advection scheme; the first corresponds to the second order upwind scheme of Warming and Beam [31]. The interpolants (3.6) and (3.7) represent the semi-Lagrangian generalisation of these two schemes to arbitrary Courant numbers.

Both the interpolants (3.6) and (3.7) employ a one-sided interpolation stencil. A centred stencil may be obtained by taking the mean of these two,

$$\begin{aligned} q_k^F &= \frac{q_k^L + q_k^R}{2} \\ &= -\frac{\xi(1-\xi)}{4}(f_{k-1} + f_{k+2}) + \frac{(1-\xi)(4+\xi)}{4} f_k + \frac{\xi(5-\xi)}{4} f_{k+1}. \end{aligned} \tag{3.8}$$

This polynomial provides the SL equivalent of Fromm's scheme. The polynomial (3.8) interpolates, as required, to the two central data points at x_k and x_{k+1}. It doesn't interpolate the data at any other grid-points.

For interpolation on an irregular grid, we may still use q^F as a piecewise interpolant. However, on such a grid it is no longer the mean of two quadratic interpolants. This is because (3.6) and (3.7) are the interpolating quadratic polynomials only on a regular grid. If instead we define q^L and q^R to be quadratic interpolants for arbitrarily placed knots, x_{k-1}, x_k, x_{k+1} and x_k, x_{k+1}, x_{k+2}, respectively, then we may define the mean quadratic interpolant,

$$q^M = \frac{1}{2}(q^L + q^R). \tag{3.9}$$

It is convenient to write this interpolant in terms of divided differences,

$$q^M = f[x_k] + (x - x_k) f[x_k, x_{k+1}] + \frac{1}{2}(x - x_k)(x - x_{k+1}) \left(f[x_{k-1}, x_k, x_{k+1}] + f[x_k, x_{k+1}, x_{k+2}] \right). \tag{3.10}$$

Interpolant 2: Least Squares Minimisation

On a regular grid it may be shown that q_k^M is the quadratic which minimises the quantity,

$$Q = \frac{1}{2} \left[q_k(x_{k-1}) - f_{k-1} \right]^2 + \frac{1}{2} \left[q_k(x_{k+2}) - f_{k+2} \right]^2. \tag{3.11}$$

This suggests that best-fitting, in a least squares sense, is a possible criterion to use in determining the extra degree of freedom in even-order interpolants. On an irregular grid, the quadratic q_k^M no longer minimises Q. We derive the quadratic polynomial which does, which is known as Bessel's second interpolation formula.

Let the quadratic polynomial obtained by minimising Q be q_k^Q. This polynomial must interpolate the data at x_k and x_{k+1}. By inspection, we see that any quadratic which interpolates to these two points may be written as

$$q_k = l_k(x) + C(x - x_k)(x - x_{k+1}), \tag{3.12}$$

where C is some constant to be determined and $l_k(x)$ is the linear interpolant,

$$l_k(x) = \frac{(x_{k+1} - x) f_k + (x - x_k) f_{k+1}}{x_{k+1} - x_k}. \tag{3.13}$$

We next determine the value of C for the quadratic which minimises Q.

Substituting (3.12) into (3.11), we obtain

$$Q = \frac{1}{2}[Ca + l_k(x_{k-1}) - f_{k-1}]^2 + \frac{1}{2}[Cb + l_k(x_{k+2}) - f_{k+2}]^2, \tag{3.14}$$

where

$$a = (x_k - x_{k-1})(x_{k+1} - x_{k-1}), \tag{3.15}$$

$$b = (x_{k+2} - x_k)(x_{k+2} - x_{k+1}). \tag{3.16}$$

Applying the minimisation condition,

$$\frac{dQ}{dC} = 0, \tag{3.17}$$

we obtain

$$C = \frac{a[f_{k-1} - l_k(x_{k-1})] + b[f_{k+2} - l_k(x_{k+2})]}{a^2 + b^2}. \tag{3.18}$$

Interpolant 3: Weighted Least Squares Minimisation

Minimisation of the quantity Q gives an equal weighting to the outlying data points, at $k-1$ and $k+2$. On an irregular grid, one of these points may be considerably further from the interpolation point, x_d, than is the other. Such grid irregularity may be represented in the minimisation process by introducing weighting factors. Consider the weighted quantity,

$$W = \frac{(x_{k+2} - x_{k+1})}{2}[Ca + l_k(x_{k-1}) - f_{k-1}]^2 + \frac{(x_k - x_{k-1})}{2}[Cb + l_k(x_{k+2}) - f_{k+2}]^2, \tag{3.19}$$

where the weight factors are simply the lengths of the two outer-most mesh intervals. Minimisation of W provides the weighted minimisation quadratic, $q^W(x)$, for which

$$C = \frac{(x_{k+1} - x_{k-1})[f_{k-1} - l_k(x_{k-1})] + (x_{k+2} - x_k)[f_{k+2} - l_k(x_{k+2})]}{(x_{k+1} - x_{k-1})a + (x_{k+2} - x_k)b}. \tag{3.20}$$

3.2 Computational Tests

We next conduct a series of computational tests with the quadratic interpolants described above. The aim of these tests is to investigate the relative merits of these interpolants for

use in semi-Lagrangian schemes. We begin by examining interpolation error and then SL advection error.

3.2.1 Test 1: Interpolation

We compare the three interpolants q^M, q^Q and q^W, together with the quadratic ENO interpolant q^E, see Section 4.1. The data to be interpolated are given by the function

$$f(x) = \begin{cases} \cos\left[\frac{\pi}{2}(x-1)\right], & 0 \le x < 2 \\ x-2, & 2 \le x < 3 \\ 4-x, & 3 \le x < 4 \\ 1, & 4 \le x < 6 \\ \exp\left[-25(x-7)^2\right], & 6 \le x \le 8. \end{cases} \tag{3.21}$$

Data obtained with this function are interpolated on K different irregular grids. Let these grids be $x^{(n)}$, where $n = n_1, \cdots, n_1 + K - 1$. Grid $x^{(n)}$ has $n+1$ grid points, $x_j^{(n)}$, $j = 0, \cdots, n$, where $x_0^{(n)} = 0$ and $x_n^{(n)} = 8$. For the results presented here, we use $n_1 = 24$ and $K = 217$. With this choice of parameters, the first grid has approximately six grid intervals covering each of the four sections in the initial data; the last grid has ten times that number. The interior grid points are given by first calculating y_j, $j = 0, \cdots, n$, where

$$\begin{aligned} y_0 &= 0 \\ y_j &= y_{j-1} + 2 + \sin(j), \quad j = 1, \cdots, n. \end{aligned} \tag{3.22}$$

The required grid points are then obtained using

$$x_j^{(n)} = 8\left(\frac{y_j}{y_n}\right), \quad j = 0, \cdots, n. \tag{3.23}$$

For each grid $x^{(n)}$, the data are interpolated to m points, which are equally spaced

between $x_1^{(n)}$ and $x_{n-1}^{(n)}$. Let these points be z_i, $i = 1, \cdots, m$. We use $m = 4000$ for all the grids tested here.

Define the error for each interpolant, q, on each grid, $x^{(n)}$, by

$$\mathrm{err}^{(n)}[q] = \left[\frac{\sum_{i=1}^{m} (q(z_i) - f(z_i))^2}{m} \right]^{\frac{1}{2}}. \tag{3.24}$$

For each interpolant, a weighted summation is made of the error on all the grids,

$$\mathrm{err}[q] = \frac{\sum_{n=n_1}^{N_1+K-1} n \, \mathrm{err}^{(n)}[q]}{\sum_{n=n_1}^{n_1+K-1} n}. \tag{3.25}$$

This provides an over-all measure of interpolation error for each interpolant. Another important factor to consider, is the degree to which each interpolant undershoots or overshoots the limits set by the interpolated data. Figure 3.1 shows the results obtained on the coarsest resolution grid. All of the interpolants, apart from quadratic ENO, show

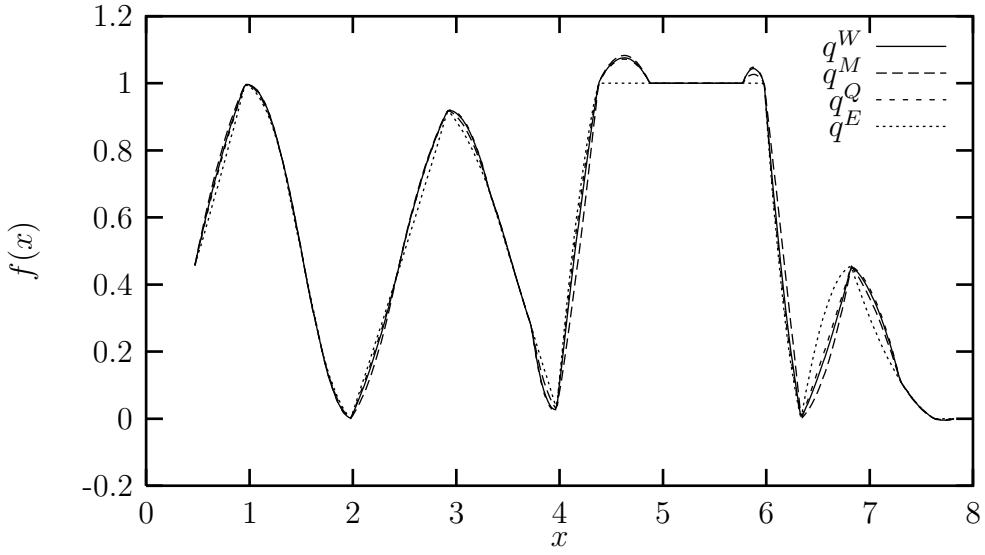

Figure 3.1: Interpolated function on a 24 point grid.

significant overshoots. ENO, however, does tend to clip extrema in an uneven fashion.

Table 3.1 shows the interpolation error for each of the interpolants. Also shown are the maximum and minimum values that each interpolant achieves throughout interpolation on all the grids. Any values below zero represent an undershoot, and values above one are an overshoot. We see that each interpolant, apart from ENO, has significant

Interpolant (q)	$\text{err}[q] * 10^{-2}$	$\min[q]$	$\max[q]$
q^M	6.40	-0.16	1.14
q^Q	6.21	-0.14	1.09
q^W	6.24	-0.15	1.08
q^E	6.03	-0.12	1.00

Table 3.1: Interpolation error and maximum and minimum values for four quadratic interpolants.

undershoots and over shoots of ten to fifteen percent. The interpolation error is much the same for each interpolant, with that for q^M being slightly larger than for the rest.

3.2.2 Test 2: Advection Test

Next we apply the same four interpolants to semi-Lagrangian advection. Again we use an irregular grid but one which is extended sufficiently to allow advection of the initial data. Let the advection equation be

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0, \tag{3.26}$$

and the initial data be given by

$$u(x,0) = \begin{cases} f(x), & 0 \le x \le 8 \\ \\ 0, & x < 0,\ x > 8, \end{cases} \tag{3.27}$$

where $f(x)$ is defined by (3.21). In the notation of the Section 3.1, we use grid $x^{(96)}$ to span the non-zero part of the initial data. That is, the interval $0 \le x \le 8$ is spanned by approximately 96 mesh intervals. This grid is then extended to larger values of x, again using the irregular grid formula (3.22).

At each time step the advected distance is 0.02, which provides a range of local Courant numbers between 0.16 and 0.48. One thousand timesteps are made. The L_2 error in the numerical solution is calculated using only those grid points at which the analytical solution is non-zero. Results are summarised in Table 3.2. The solution obtained with

Interpolant (q)	$\mathrm{err}_{\mathrm{L2}}[q]$	$\min[q]$	$\max[q]$
q^M	0.180	-0.105	1.118
q^Q	0.157	-0.092	1.068
q^W	0.159	-0.091	1.082
q^E	0.269	-0.004	1.001

Table 3.2: Advection error and maximum and minimum values for four quadratic interpolants.

each scheme is plotted in Figure 3.2. Although all these schemes are of the same formal order of accuracy, we see that the ENO method performs significantly less well than the other methods. In terms of computational expense, q^M is the most efficient of the four interpolants so far considered. We conclude this chapter by making three further advection tests with this interpolant.

3.2.3 Test 3: Monotone Advection

The ENO method is close to being monotone. In this test we examine whether this is the reason for the ENO interpolant's poor performance. The mean of left and right quadratics, q^M, provides a monotone advection scheme when used with the monotone limiter of Bermejo and Staniforth [3], see Section 4.3. Linear interpolation is used for the low order, monotone scheme required by this limiter. The algorithm is as follows:

1. Calculate the maximum and minimum values of the data neighbouring the departure

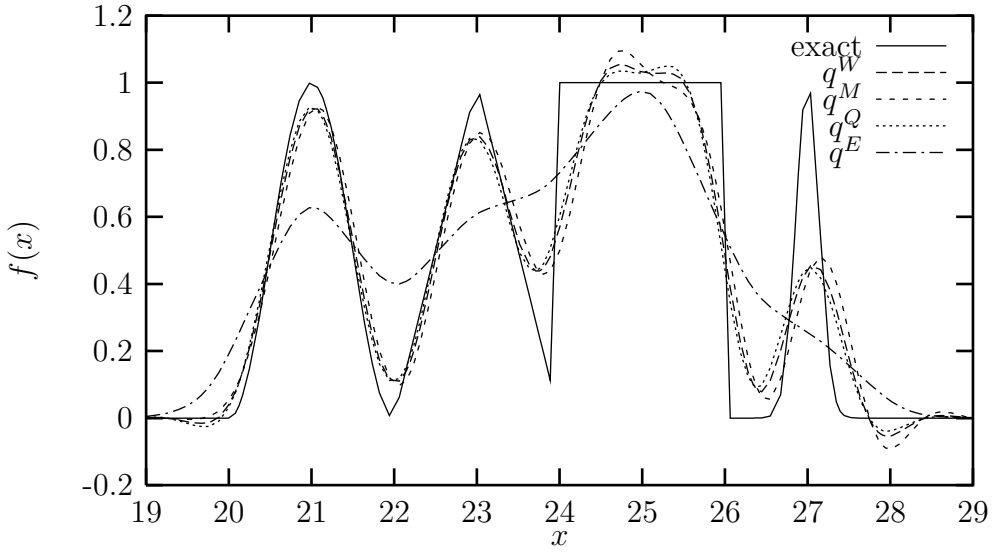

Figure 3.2: Constant speed advection test for four different centred quadratic interpolants, on an irregular grid. Advected fields shown after 1000 timesteps, with local Courant number varying between 0.16 and 0.48.

point,

$$u_{min} = \min\left(u_k^n, u_{k+1}^n\right) \tag{3.28}$$

$$u_{max} = \max\left(u_k^n, u_{k+1}^n\right); \tag{3.29}$$

2. Limit the quadratic interpolation to lie within the bounds set by u_{min} and u_{max},

$$u_j^{n+1} \leftarrow \min\left(\max\left(u_j^{n+1}, u_{min}\right), u_{max}\right). \tag{3.30}$$

Figure 3.3 shows the result obtained when the advection test is repeated with q^M made monotone in this way. Clearly, the ENO approach to monotonicity is not to be favoured in the context of quadratic interpolation.

3.2.4 Test 4: Comparison with Regular-Grid Scheme

Computationally, the mean quadratic, q^M, is more expensive to implement than the Fromm quadratic, q^F. This is because q^M is based on interpolants which correctly respect

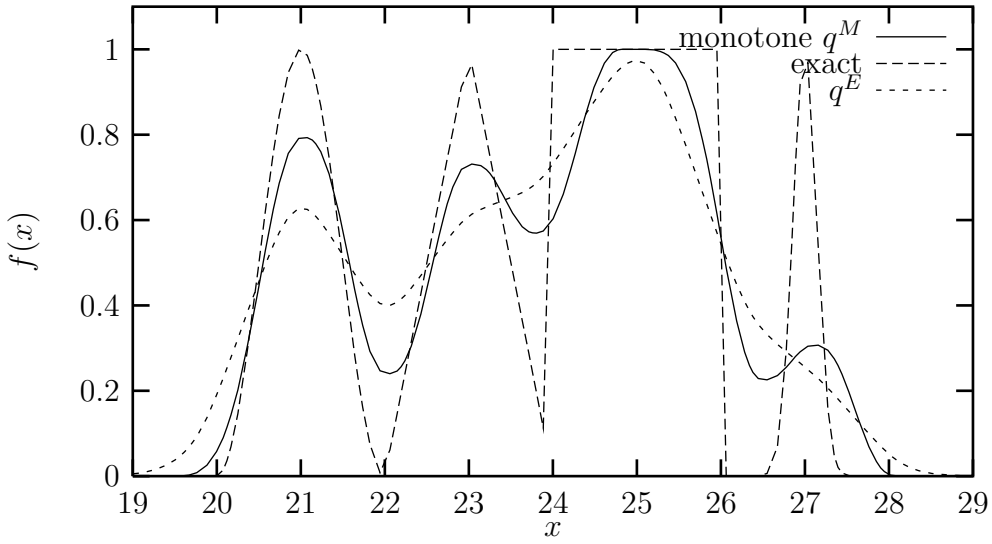

Figure 3.3: Advection test comparing q^M with monotone filter against quadratic ENO method. Data advected at constant speed across an irregular mesh for 1000 timesteps.

the positions of the outlying knots, x_{k-1} and x_{k+2}, on an irregular grid. The Fromm interpolant assumes that x_{k-1}, x_k, x_{k+1} and x_{k+2} are equally spaced. Generally, the semi-Lagrangian method is applied on regular grids, or on grids where the variation in mesh length is very gradual. The error in using q^F, rather than q^M as the basis for a quadratic SL scheme, therefore, may be expected to be slight. The grid used in the current advection test has more extreme variation in mesh length than would be used in practice. It has a maximum ratio of successive mesh intervals of roughly 1.7. By way of comparison, the idealised two-dimensional version of The Met. Office Unified Model Version 5 uses a maximum mesh ratio of 1.1. Figure 3.4 shows the result obtained when q^F is used in the advection test. In terms of undershoots and overshoots it performs better than q^M. However, the over-all error in the solution is larger for q^F, Table 3.3. Nevertheless, the difference in performance between the two schemes is reasonably small. Therefore the Fromm scheme may be used as a replacement for the mean scheme, where computational expense is an issue.

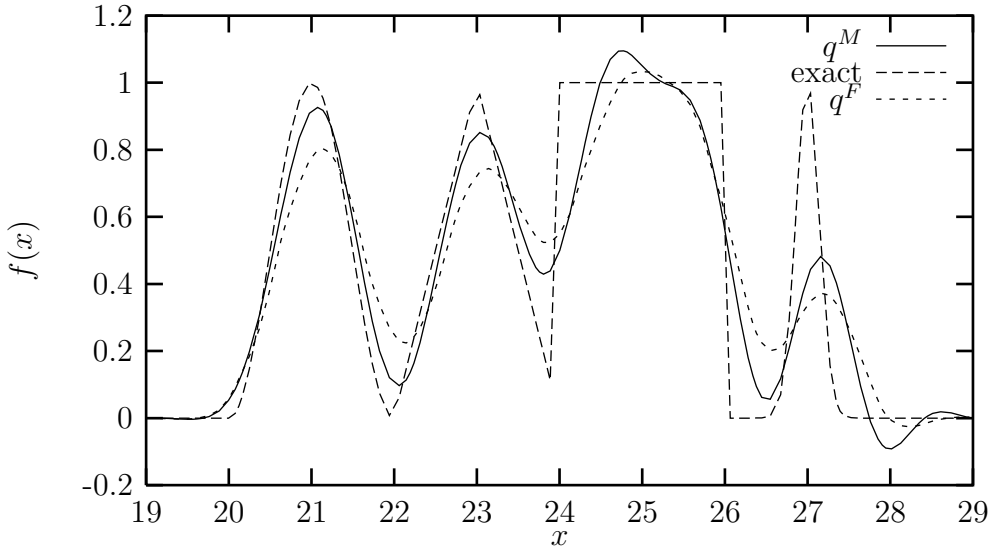

Figure 3.4: Comparison of mean and Fromm quadratic interpolants. Constant speed advection test over an irregular grid. Solutions shown after 1000 timesteps.

3.2.5 Conclusion

The computational tests show that the quadratic Fromm interpolant (3.8) compares well against other, more accurate quadratic interpolants. This scheme has potential for application in semi-Lagrangian advection schemes.

Interpolant (q)	$\mathrm{err}_{\mathrm{L2}}[q]$	min[q]	max[q]
q^M	0.180	-0.105	1.118
monotone q^M	0.210	0.000	1.000
q^F	0.221	-0.071	1.075

Table 3.3: Advection error and maximum and minimum values for four quadratic interpolants.

Chapter 4

Nonlinear Schemes

Linear schemes with high order accuracy introduce spurious small-scale oscillations into the numerical solution, as shown in Figure 2.4. In order to prevent these oscillations nonlinear interpolation methods are required. Three such methods are reviewed here.

4.1 ENO Interpolation

The essentially non-oscillatory (ENO) interpolation employed by Harten, Engquist, Osher and Chakravarthy in their advection schemes [12], [11] is immediately applicable to SL schemes. In this context ENO is a nonlinear adjustment to the Lagrange interpolating polynomial in divided difference form. The following description of ENO interpolation employs the notation of Section 2.4.1.

Starting with linear interpolation, the order of accuracy of the ENO interpolant is increased by the addition of successive terms, producing a sequence of interpolants $q_1, \ldots, q_N$ up to the desired accuracy. Each new term is a polynomial in x and has a divided difference, of appropriate order, as leading coefficient. ENO differs from standard divided difference interpolation through involving a wider range of divided difference values which may be calculated from the data. Using the notation of Section 2.4.1, the

first order ENO interpolant at the point x, where $x_i \leq x < x_{i+1}$, is

$$q_1(x) = f[x_i] + (x - x_i)f[x_i, x_{i+1}].$$

Second order ENO interpolation, $q_2(x)$, is formed by adding a quadratic term to the above, of the form

$$f[\ ,\ ,\ ](x - x_i)(x - x_{i+1}),$$

where $f[\ ,\ ,\ ]$ is a second divided difference. The first divided difference, in the linear term, uses data at the two points x_i and x_{i+1}. These same two points occur in two second order divided differences, namely $f[x_{i-1}, x_i, x_{i+1}]$ and $f[x_i, x_{i+1}, x_{i+2}]$. It is at this point that ENO interpolation first differs from Newton interpolation. For now it is whichever of these two values which has least modulus that is used to form the second order term.

At the next stage, where a cubic term is added to q_2 to form q_3, ENO interpolation again differs from the Newton formula. Not only do we select the third divided difference of least modulus for the coefficient of this term, but also the cubic polynomial must be chosen to be consistent with the quadratic term already chosen. If the coefficient of the quadratic term was $f[x_{i-1}, x_i, x_{i+1}]$, then the polynomial in the cubic term must be $(x-x_{i-1})(x-x_i)(x-x_{i+1})$; otherwise the cubic polynomial is $(x-x_i)(x-x_{i+1})(x-x_{i+2})$. All subsequent terms needed to form $q_4, q_5, \ldots$ are calculated in the same manner.

The essentially non-oscillatory property of this interpolation is best described in terms of the total variation of the data [10], which is defined by

$$\mathrm{TV}\{u_j^n : -\infty < j < \infty\} = \sum_j \left| u_{j+1}^n - u_j^n \right|.$$

The ENO interpolation, composed of terms up to and including order N polynomials, preserves the total variation of the data to order Δx^N. That is, a SL advection scheme using ENO interpolation of order N satisfies

$$\mathrm{TV}\{u_j^{n+1}\} = \mathrm{TV}\{u_j^n\} + \mathrm{O}(\Delta x^N). \tag{4.1}$$

A number of consequences follow from this property. The most important of these is that any spurious oscillations introduced into the numerical solution by ENO interpolation can

only grow at a rate limited by (4.1). In practice such oscillations remain small compared to those introduced by Lagrange interpolation. Furthermore the oscillations remain localised about steep gradients in the data, and are rapidly damped away from these regions. A second consequence of (4.1) also makes ENO beneficial for us in advection schemes: pre-existing extrema in the data are damped to a lesser degree than they would be by Lagrange interpolation. Unfortunately these first two properties lead to a third which diminishes the suitability of ENO interpolation for some applications. Since ENO allows the depth of a minimum point in the solution to decrease, positivity of interpolation may be lost even when the data are strictly positive. This would make a SL scheme based on ENO interpolation unsuitable for moisture calculations, for instance, unless further steps are taken to preserve positivity.

4.2 Hermite Cubic Interpolation

We briefly mention here a method for constructing "shape-preserving" interpolation which has been well surveyed in the SL literature, [61], [43]. By shape-preserving we mean any interpolation which respects some assumed property of the data, such as positivity, monotonicity and convexity, [7]. The shape, in the above sense, of Hermite interpolation is determined by the values given to the derivative estimates. Fritsch and Carlson [8] show that the following procedure produces monotone interpolation with a Hermite cubic polynomial.

Let $p(x)$ be the Hermite cubic interpolant on the interval $[x_i, x_{i+1}]$, see Section 2.4.1. Four conditions are required to determine a cubic polynomial. The first two are simply the interpolation conditions,

$$p(x_i) = f(x_i)$$

$$p(x_{i+1}) = f(x_{i+1}).$$

Two degrees of freedom remain. These are to be accounted for by assigning derivative

values to the Hermite cubic at the ends of the interval:

$$d_i \approx f'(x_i)$$

$$d_{i+1} \approx f'(x_i).$$

Any method, such as one of those listed in Section 2.4.1, may be used for the derivative estimates. Once initial estimates for d_i and d_{i+1} have been made, these are then adjusted if necessary to ensure the interpolant remains monotone across the interval. This is done by comparing the estimates against the discrete slope of the data across the interval, $\Delta_i = (f(x_{i+1}) - f(x_i))/(x_{i+1} - x_i)$. If $\Delta_i = 0$ then the interpolation will be monotone across the interval only if both derivative estimates are zero, producing a constant valued interpolation. That is, if $\Delta_i = 0$ then d_i and d_{i+1} are set to zero, regardless of their original values. If Δ_i is non-zero, then a sufficient condition for monotonicity is given in terms of the ratios between derivative estimates and the discrete slope. Let $\alpha = d_i/\Delta_i$ and $\beta = d_{i+1}/\Delta_i$. The interpolation is monotone if the following conditions are satisfied:

$$\begin{cases} 0 \leq \alpha \leq 3 \\ 0 \leq \beta \leq 3. \end{cases}$$

The simplest approach to ensuring these conditions are satisfied, is to reset the derivative estimates according to

$$d_i \leftarrow 0 \qquad \text{if } \alpha < 0$$

$$d_i \leftarrow 3\Delta_i \quad \text{if } \alpha > 3.$$

The value of d_{i+1} is adjusted with reference to β in the same way.

4.3 Quasi-Monotone Schemes

Introduction

Bermejo and Staniforth [3] describe a general method for implementing monotone interpolation in SL schemes. This method, which they term quasi-monotone semi-Lagrangian (QMSL), is general in the sense that it may be applied to any existing SL scheme. The term monotone is here used to denote any interpolation which does not introduce new extrema into the data. Indeed the method is applicable in any number of dimensions, being monotone in the above sense. In this section we describe the method in detail.

It is assumed that there is a high accuracy, non-monotone scheme which we wish to make monotone. Also available is a scheme using monotone interpolation, but which is of a low order of accuracy. Following an approach similar to the flux-corrected transport algorithm of Boris & Book [4] and Zalesak [62], solutions from the two schemes are merged to produce a new monotone, high-order solution. The algorithm for merging the solutions is as follows.

4.3.1 QMSL Algorithm

Let the high order and low order solutions at node j be u_{Hj}^{n+1} and u_{Lj}^{n+1} respectively. And let $u_d^{\max}$ and $u_d^{\min}$ be the maximum and minimum values respectively of the solution at the previous time-level, in the neighbourhood of the departure point of node j. This neighbourhood may be defined as the grid-points forming the corners of the cell in which the departure point lies. If the result of merging the two solutions is the new solution u_j^{n+1}, then the scheme is monotone if

$$u_d^{\min} \leq u_j^{n+1} \leq u_d^{\max}.$$

Priestley [38] provides an extension to QMSL, which allows its use in nonlinear or inhomogeneous advection problems. In such problems the solution at one time-level does not necessarily provide a local bound for the solution at the next. By allowing the low

order, but monotone, solution to provide a less severe bound where required, we avoid any difficulties in this respect. So, following Priestley, we set

$$\begin{aligned} u_j^+ &= \max\{u_d^{\max}, u_{Lj}^{n+1}\} \\ u_j^- &= \min\{u_d^{\min}, u_{Lj}^{n+1}\}, \end{aligned}$$

and require the QMSL solution to satisfy

$$u_j^- \leq u_j^{n+1} \leq u_j^+. \tag{4.2}$$

This requirement ensures that no new extrema are created by the interpolation stage of the QMSL scheme. Next we consider how to merge the high and low order solutions to obtain a solution satisfying the constraint (4.2), while preserving the maximum degree of accuracy possible.

Beginning with the low order solution, a correction is sought which will increase the accuracy in the solution,

$$u_j^{n+1} = u_{Lj}^{n+1} + \delta_j^{n+1}.$$

The correction term δ_j^{n+1} is derived from the high order solution. In the following computations the time superscript is dropped for clarity. Let

$$\delta_j = \alpha_j(u_{Hj} - u_{Lj}), \quad 0 \leq \alpha_j \leq 1,$$

where α_j are to be determined. If all the $\alpha_j = 1$ the QMSL solution is just the high order solution. The task is now to find α_j as large as possible, within the allowed range, such that the monotonicity constraint (4.2) is yet satisfied. This may be achieved by first computing the maximum and minimum allowable corrections,

$$\begin{aligned} Q^+ &= u_j^+ - u_{Lj} \\ Q^- &= u_j^- - u_{Lj}, \end{aligned}$$

and the difference between the two existing solutions,

$$P = u_{Hj} - u_{Lj}.$$

There are now three possibilities:

$$1) \quad P > 0 \quad : \quad \alpha_j = \min\{1, \frac{Q^+}{P}\}$$

$$2) \quad P < 0 \quad : \quad \alpha_j = \min\{1, \frac{Q^-}{P}\}$$

$$3) \quad P = 0 \quad : \quad \alpha_j = 0.$$

It is a simple matter to verify that these formulae provide the desired result,

$$u_j^{n+1} = \begin{cases} u_{Hj}^{n+1} & \text{if} \quad u_j^- \leq u_{Hj}^{n+1} \leq u_j^+ \\ u_j^+ & \text{if} \quad u_j^+ < u_{Hj}^{n+1} \\ u_j^- & \text{if} \quad u_{Hj}^{n+1} < u_j^- . \end{cases}$$

A simpler computational route to the same result is

$$u_j^{n+1} = \min[\max(u_{Hj}^{n+1}, u_j^-), u_j^+].$$

However, there is a considerable advantage to be gained in calculating the correction terms α_j, as will be seen in Section 9.4.3.

4.4 A Computational Example

We conclude this chapter with a brief illustration of the benefits to be gained from stable advection calculations at high Courant number. For an Eulerian explicit advection scheme the CFL stability criterion imposes a restriction on the length of a time-step, this limiting value being proportional to the flow speed. As we shall see in the next chapter, the corresponding stability restriction for a SL scheme is often far less severe, and depends on the rate of deformation of the flow. Freedom to take longer time-steps allows a given forecast time to be reached with fewer steps. Two benefits accrue from this. First, since fewer steps are taken it is possible to reduce the total operation count for the forecast, so

gaining operational speed. Secondly, by reducing the number of steps the accumulated error may be reduced, so improving forecast quality. Neither of these advantages follows automatically from switching to a SL scheme, but may be realised by appropriate choices for the operational details of the scheme. For instance, computational speed may be lost by using an unduly computationally intensive interpolation algorithm; and reduction of accumulated errors depends on careful balancing of the time-step length with the truncation error of the time discretisation.

We shall not pursue a thorough investigation of these issues here, but merely illustrate some of the issues with the following set of numerical tests. Again it is the linear advection problem that we solve, using initial data

$$u_0(x) = \begin{cases} \cos^2\left[\dfrac{\pi}{2}\dfrac{(x-15)}{5}\right] & 10 \leq x \leq 20 \\ \\ 0 & \text{elsewhere,} \end{cases}$$

and the discretisation lengths are $\Delta t = \Delta x = 1$. This pulse is advected through a distance of 1000 grid cells. Figure 4.1 shows the results obtained using cubic Lagrange

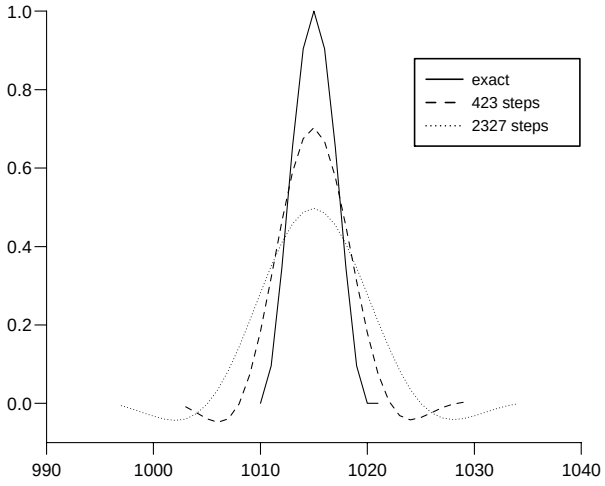

Figure 4.1: Cubic Interpolation without monotonicity constraint

interpolation, without any control for monotonicity. The accuracy of the result obtained is seen to depend strongly on the number of time steps taken. When 423 time steps are taken, which corresponds to a CFL number of $1000/423 \approx 2.36$, the peak height of the

numerical solution drops to about 75% of that of the exact solution. The position of the pulse appears to be very close to being exact, though with a slight lag, as anticipated by the Fourier analysis of Section 2.4.2

Repeating the simulation once more, this time using 2327 time-steps to achieve the total displacement of one thousand cells (CFL ≈ 0.43), a quite different result obtains. The peak height has now dropped close to 50% of its exact value. and there is a slightly greater lag in the position of the pulse. In addition, we see that the extent (or support) of the numerical solution has increased considerably: the plots show the portions of the solution where the magnitude is one percent or more of the exact peak height. From this we may conclude that the reduction in accumulated error, which results from running simulations at high Courant numbers, is a significant advantage for SL schemes.

Both of the above numerical solutions show a loss of monotonicity at the leading and trailing edges of the pulse. As a consequence positivity is also lost in these regions. Using the monotonicity preserving method of Bermejo and Staniforth (Section 4.3), and repeating once more the calculation with 423 time-steps, the result of which is shown in Figure 4.2, we find that positivity and monotonicity are preserved. The position of this

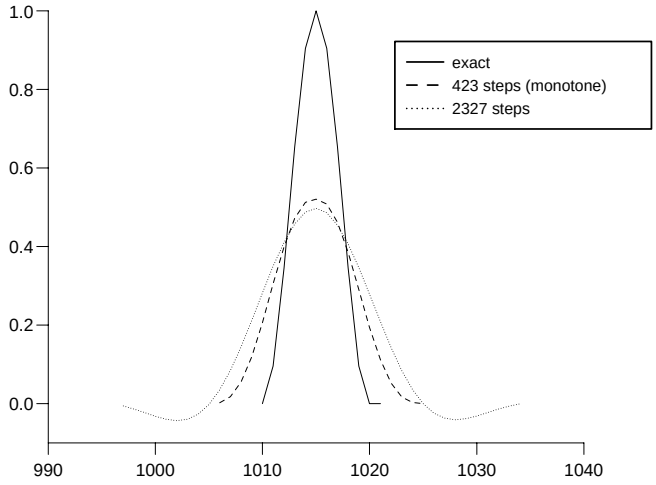

Figure 4.2: Monotone solution

new solution is comparable in accuracy to the two previous cases, and the support of the solution shows some improvement in compactness. Unfortunately these gains are achieved

at the expense of a large error in the value of peak height: the peak height for 423 steps with monotone interpolation is roughly that obtained by 2327 steps with unconstrained interpolation. In terms of the accuracy of the peak value, any gain made by solving at a high Courant number is lost when monotonicity constraints are enforced.

Chapter 5

Interpolation: Multi-Dimensional Schemes

5.1 Introduction

In Chapter 2 we reviewed a selection of methods for univariate interpolation. All of these methods were immediately applicable to SL schemes for one-dimensional problems. Any meaningful approach to numerical weather prediction (NWP) must, however, involve a representation of data throughout a three dimensional domain. Only in such a large model can we hope to describe adequately the processes of the atmosphere. There is a need, therefore, to find multivariate interpolation methods suitable for SL advection.

A simulation using finite differences typically involves a large array of grid points, structured into layers, rows and columns. For each of these grid points a SL scheme would have to compute a departure point. Then for each departure point perform a number of interpolations, one for each model variable. The precise number of variables in a model depends on the number of physical processes to be simulated. A basic weather model would include variables to describe the three components of velocity together with pressure, temperature and moisture. Atmospheric chemistry models may involve many chemical species, each requiring its own model field. Consequently, a SL model may

often involve an extremely large number of interpolations at each time-step. It is readily apparent that much of the computational effort in such simulations will be spent on interpolation. The viability of a SL scheme rests on the efficiency of its interpolation.

A number of methods exist for interpolating multi-dimensional data. Since we require interpolations on a regular grid, we shall restrict our attention to methods suitable for such an arrangement. The most familiar method of this kind is tensor product interpolation, which is described in the next section. Basis-spline methods have recently been applied to SL schemes [28]. Lastly, Section 5.3 contains details of cascade interpolation, whose flexibility is becoming apparent in SL schemes [39], [23], [42].

5.2 Tensor Product Interpolation

Tensor product interpolation consists of repeated univariate interpolations along coordinate directions. A simple example serves as a description of the method.

Let f_{ij} be data values on a two-dimensional grid. The nodes of this grid lie at the intersection points of the lines

$$\begin{aligned} x &= x_i \\ y &= y_j, \end{aligned}$$

where i and j nodal indices We wish to interpolate the data to a point (a, b) within the grid. If the interpolation point lies in grid-cell (i, j), then

$$x_i \leq a < x_{i+1}$$

$$y_j \leq b < y_{j+1}$$

for some i and j. Bi-cubic, tensor product interpolation consists of applying univariate cubic interpolation along each of the lines

$$y = y_{j+k}, \quad k = -1, 0, 1, 2.$$

Each of these interpolations is evaluated at the point $x = a$. Then a final cubic interpolation is made of the data just obtained, which is evaluated at the point $y = b$. If $l_i(x)$ and $l_j(y)$ are the basis functions for cubic Lagrange interpolation on the grid, then the bi-cubic tensor product is

$$f(a,b) = \sum_{r=-1}^{2} \sum_{s=-1}^{2} f_{i+r\,j+s}\; l_r(a) l_s(b).$$

5.3 Leslie and Purser: Cascade Interpolation

The cascade method of Leslie and Purser [39] is a method for multi-dimensional interpolation fulfilling many of the requirements of SL schemes. Recently it has been shown how the method can be made conservative [23].

The notation to describe this method is rather cumbersome, but the idea itself is straightforward. Instead of considering each interpolation in isolation, Purser and Leslie address the task of transferring data form the regular computational grid, to the irregular grid formed by all the Lagrangian departure points.

At time level t^n finite difference data $\{\phi_{ijk}\}$ is available on the grid formed by the intersection of the planes

$$\begin{aligned} x &= \hat{x}_i \qquad i = 1, \cdots, I \\ y &= \hat{y}_j \qquad j = 1, \cdots, J \\ z &= \hat{z}_k \qquad k = 1, \cdots, K \end{aligned} \tag{5.1}$$

At the intersections of each of these planes, we have the grid points $(\hat{x}_i, \hat{y}_j, \hat{z}_k)$. By tracing back along a parcel trajectory from each point $(\hat{x}_i, \hat{y}_j, \hat{z}_k, t^n + \Delta t)$, to time level t^n, we arrive at the departure points. Equivalently we can consider this same mapping applied to each of the planes in (5.1). In general each plane will map to some curved surface, and the intersection points between all these surfaces are just the departure points. However these curved surfaces also provide the notion of a departure grid (or Lagrangian grid) embedded within the computational grid. By locating the points of intersection between the two

grids, we can perform a sequence — or cascade — of one dimensional interpolations that transfers data from one to the other. Under certain conditions on the velocity field, the Lagrangian grid surfaces will provide a valid curvilinear coordinate system, (X, Y, Z).

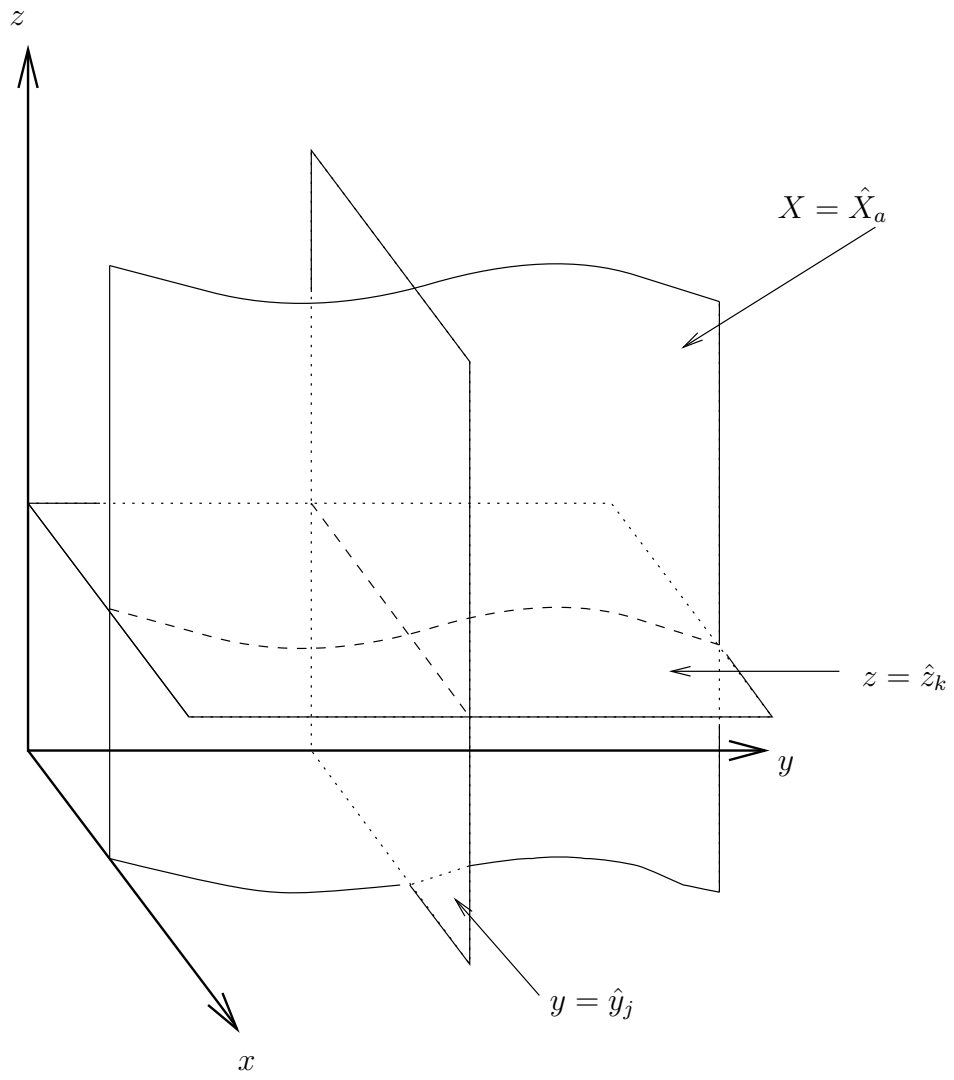

Figure 5.1: Hybrid Grids

The cascade method consists of a sequence of intermediate interpolations, which produce approximations to ϕ at points where the two grids intersect. These points of intersection are referred to as *hybrid* grid points. By transferring data through successive hybrid grids, we arrive at a representation of ϕ on the Lagrangian grid. We next consider this process in detail.

To illustrate what is meant by a hybrid grid point, consider first the regular grid

of points $(\hat{x}_i, \hat{y}_j, \hat{z}_k)$. The two coordinate planes, $y = \hat{y}_j$ and $z = \hat{z}_k$, intersect in a line $(\hat{y}_j, \hat{z}_k)$. This line is cut by the plane $x = \hat{x}_i$ at the regular grid point $(\hat{x}_i, \hat{y}_j, \hat{z}_k)$. Similarly the Lagrangian grid is obtained from the intersection of the coordinate planes $X = \hat{X}_a$, $Y = \hat{Y}_b$, $Z = \hat{Z}_c$, where the indices take the same ranges as in the regular grid. Each such plane in the (X, Y, Z) coordinate system will, in general, correspond to a curved surface in the (x, y, z) system. Now by considering intersections between (x, y, z) planes and (X, Y, Z) surfaces, we can construct various hybrid grid structures. For instance, the point of intersection between the planes $z = \hat{z}_k$ and $y = \hat{y}_j$ with the surface $X = \hat{X}_a$ defines the hybrid grid point $(\hat{X}_a, \hat{y}_j, \hat{z}_k)$. These various geometrical structures are indicated in figure 5.1.

The method splits into two parts. First we must find the (x, y, z) coordinates of certain hybrid grid points. Once these have been found, a cascade of one dimensional interpolations extends the representation of ϕ from the regular grid, to the Lagrangian grid. We shall first describe how the cascade of interpolations is organised, before considering how the location of hybrid grid points might be found. By considering the interpolation stage first, we can also see how each of the hybrid grids becomes involved.

Consider again the coordinate line $(\hat{y}_j, \hat{z}_k)$. This line is cut by the coordinate plane, or surface, $X = \hat{X}_a$ at the hybrid grid point $(\hat{X}_a, \hat{y}_j, \hat{z}_k)$. If we know the x-coordinate — $x(\hat{X}_a, \hat{y}_j, \hat{z}_k)$ — of this hybrid point, then we can interpolate ϕ to that point using the available finite difference data along the line $(\hat{y}_j, \hat{z}_k)$. That is, we form a one dimensional interpolant on the data set $\{(\hat{x}_i, \phi(\hat{x}_i, \hat{y}_j, \hat{z}_k)) : i = 1, \cdots, I\}$. Evaluating this interpolant at $x = x(\hat{X}_a, \hat{y}_j, \hat{z}_k)$, we obtain an approximation for $\phi(\hat{X}_a, \hat{y}_j, \hat{z}_k)$. The interpolation used here is local, perhaps a Lagrange interpolant or shape preserving Hermite cubic. The procedure is repeated for all the hybrid points $(\hat{X}_a, \hat{y}_j, \hat{z}_k)$.

The next step in the interpolation part of the scheme builds on the data we have gathered at the hybrid grid points $(\hat{X}_a, \hat{y}_j, \hat{z}_k)$. Our aim is eventually to find $\phi(\hat{X}_a, \hat{Y}_b, \hat{Z}_c)$. We can move a step nearer this goal by replacing $\hat{y}_j$ by $\hat{Y}_b$ in the hybrid representation we have just obtained. This can be done in much the same way as we replaced $\hat{x}_i$ by

$\hat{X}_a$ above. So consider the curve of intersection, $(\hat{X}_a, \hat{z}_k)$, between $X = \hat{X}_a$ and $z = \hat{z}_k$. This curve is cut by the plane $Y = \hat{Y}_b$ at the hybrid grid point $(\hat{X}_a, \hat{Y}_b, \hat{z}_k)$. All along the curve $(\hat{X}_a, \hat{z}_k)$ we have values of ϕ at the hybrid grid points $(\hat{X}_a, \hat{y}_j, \hat{z}_k)$. If we know the y-coordinate of each of these hybrid points, and also the y-coordinate of the *new* hybrid point $(\hat{X}_a, \hat{Y}_b, \hat{z}_k)$, then we can interpolate to find $\phi(\hat{X}_a, \hat{Y}_b, \hat{z}_k)$. Again this is repeated for all the new hybrid grid points.

The final step is straightforward. Following the usual procedure, we consider the intersection of $Z = \hat{Z}_c$ with the curve $(\hat{X}_a, \hat{Y}_b)$. This is just the Lagrangian grid point $(\hat{X}_a, \hat{Y}_b, \hat{Z}_c)$. As this is a departure point we're assuming we already know its (x, y, z) coordinates. This means the interpolation can follow immediately: we know the z-coordinate of all the hybrid grid points $(\hat{X}_a, \hat{Y}_b, \hat{z}_k)$, and we know the z-coordinate of the point $(\hat{X}_a, \hat{Y}_b, \hat{Z}_c)$; so a one dimensional interpolation in the z-direction gives us $\phi(\hat{X}_a, \hat{Y}_b, \hat{Z}_c)$.

The above procedure allows us to interpolate the data from the regular grid to all the departure points using only three one dimensional interpolations per point. It only remains to find the locations of all the hybrid grid points. This must be done before any of the interpolations can proceed.

In the cascade procedure just described, we made use of the (x, y, z) coordinates of hybrid grid points. But initially we only have the coordinates of the Lagrangian grid points. The first stage of the whole interpolation method, therefore, is to extend our coordinate information to cover all the hybrid grid points. Just as in the cascade stage, we will achieve this using a sequence of interpolations starting from the known data.

The problem can be set out as follows.
We know $(x(\hat{X}_a, \hat{Y}_b, \hat{Z}_c), y(\hat{X}_a, \hat{Y}_b, \hat{Z}_c), z(\hat{X}_a, \hat{Y}_b, \hat{Z}_c))\ \forall a, b, c$, as these are just the departure points found by solving the trajectory equation. The cascade of interpolations requires the x-coordinates of one set of hybrid grid points, and the y-coordinates of another set of hybrid points. These coordinates are $x(\hat{X}_a, \hat{y}_j, \hat{z}_k)$ and $y(\hat{X}_a, \hat{Y}_b, \hat{z}_k)$ respectively.

First consider the curve $(\hat{X}_a, \hat{Y}_b)$. Along this curve we know all the y- and z-coordinates of the Lagrangian grid points. This allows us to interpolate the function

y , along the curve $(\hat{X}_a, \hat{Y}_b)$, to the point where $z = \hat{z}_k$.

Secondly we require $x(\hat{X}_a, \hat{y}_j, \hat{z}_k)$. This is slightly more complicated to achieve, and requires a two-step approach. Just as we found $y(\hat{X}_a, \hat{Y}_b, \hat{z}_k)$, we can also find $x(\hat{X}_a, \hat{Y}_b, \hat{z}_k)$. Thus we have values of x, considered as a function of y, at points along the line $(\hat{X}_a, \hat{z}_k)$. These can then be interpolated to evaluate x at $y = \hat{y}_j$. The result of this is $x(\hat{X}_a, \hat{y}_j, \hat{z}_k)$, and the algorithm is complete.

When implemented in a SL code, the method of cascade interpolation is found to be substantially faster than the Cartesian product method. The formal accuracy of the method, however, is not a simple quantity to identify. Experimentally it is found that, for reasonably smooth Lagrangian grids, the accuracy of the two methods is much the same.

Neither is grid smoothness a problem. The cascade method relies on the Lagrangian coordinates being single valued functions of the regular coordinates. This will be the case if the deformational Courant number, $\Delta t \dfrac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}$, is less than one. This condition is precisely the convergence condition we have from the trajectory calculation, so we see that the cascade method is well suited to semi-Lagrangian advection.

Chapter 6

Non-interpolating Methods

In this chapter, we shall look at an alternative form for semi-Lagrangian schemes, which avoids the need for explicit interpolation. Two problems with the interpolation stage of SL algorithms have been identified, both of which have important implications for numerical weather prediction. The first is the damping which is associated with most interpolants. This is seen to present particular problems for climate models, where model resolution is low enough to allow such damping to seriously degrade accuracy. The second problem is one of computational efficiency: interpolation is an expensive operation, especially for three dimensional models. Ritchie [44] addresses these problems by proposing a non-interpolating version of the SL method.

Ritchie's approach is to decompose each fluid parcel trajectory into two components: one which connects grid points between time levels and a residual displacement. The scheme is designed such that the residual component is always sufficiently small to be treated stably by an Eulerian approach. The original form of his scheme has a three time-level format, but more recent work by Olim [34], and Smolarkiewicz and Pudykiewicz [53], has developed the method as a, now more conventional, two time-level scheme.

6.1 Ritchie's Scheme

Ritchie's original derivation of a non-interpolating scheme proceeds first by splitting the advecting velocity into two components. If $\boldsymbol{u}$ is the advecting velocity, then the decomposition is:

$$\boldsymbol{u} = \boldsymbol{u}_0 + \boldsymbol{u}', \tag{6.1}$$

where $\boldsymbol{u}_0$ is a vector field connecting grid points between time levels, and $\boldsymbol{u}'$ is a residual velocity field. Applying (6.1) to the linear advection equation

$$\frac{\partial \phi}{\partial t} + \boldsymbol{u}.\boldsymbol{\nabla}\phi = 0, \tag{6.2}$$

we obtain

$$\frac{\partial \phi}{\partial t} + \boldsymbol{u}_0.\boldsymbol{\nabla}\phi = -\boldsymbol{u}'.\boldsymbol{\nabla}\phi. \tag{6.3}$$

The scheme proceeds by first assigning node-by-node values to the two velocity fields, $\boldsymbol{u}_0$ and $\boldsymbol{u}'$. The vector $\boldsymbol{u}_0$ is chosen to connect the arrival node, $\boldsymbol{x}_i$, with the node, $[\boldsymbol{x}_d]$, which is closest to the departure point $\boldsymbol{x}_d$. Once $\boldsymbol{u}_0$ has been assigned the residual component $\boldsymbol{u}'$ is found from (6.1). For the one-dimensional problem, with a regular grid of spacing Δx, a two time-level scheme takes the following form.

For a given node, at position x_i , (see Fig. 6.1):

1) Find $[x_d]$, and let p be the number of mesh lengths from this node to the arrival point x_i.

2) Split the advection velocity: $u = p\dfrac{\Delta x}{\Delta t} + u'$.

3) The SL solution of (6.3) is given by:

$$\frac{\phi(x_i, t+\Delta t) - \phi(x_i - p\Delta x, t)}{\Delta t} = \left(-u'\frac{\partial \phi}{\partial x}\right)\left(x_i - \frac{p\Delta x}{2}, t + \frac{\Delta t}{2}\right). \tag{6.4}$$

The derivative terms in (6.4) are represented by central differences.

This method can be extended, in a straightforward manner, for use in two and three dimensional problems.

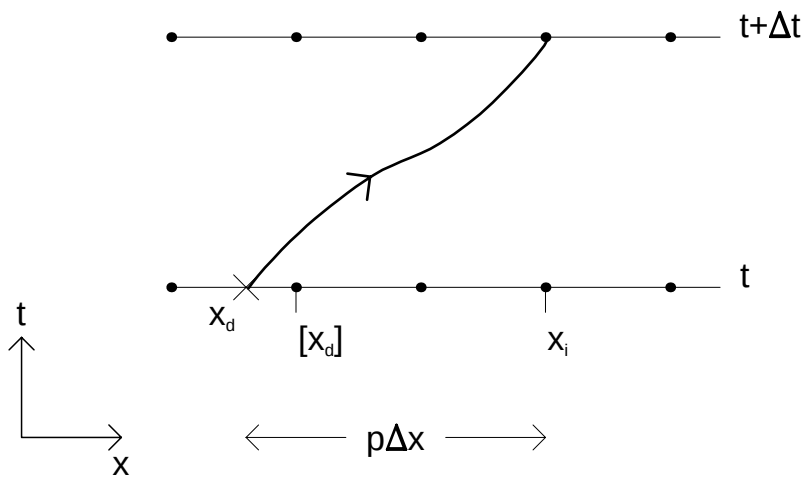

Figure 6.1: Ritchie's scheme

6.2 Olim's Scheme

Olim [34] points out that Ritchie's scheme does not totally eliminate the need for interpolation. The right hand side of (6.4) does not, in general, coincide with a grid point. Some form of interpolation must be used to carry information to the correct location. For the one dimensional scheme, this situation arises whenever p is an odd number, for then $x_i - \frac{p\Delta x}{2}$ lies between grid points. The situation becomes more complicated in two and three dimensions, for then the point of evaluation can lie at any of a considerable number of different positions within each grid cell. In a truly non-interpolating scheme, the right hand side would always be evaluated at a grid point. Olim achieves this by applying the Lax-Wendroff method, along characteristics, to solve (6.3).

Olim's scheme is derived by rewriting (6.3) as the Lagrangian equation,

$$\frac{d\phi}{dt} = -\boldsymbol{u}'.\boldsymbol{\nabla}\phi \tag{6.5}$$

where $\frac{d}{dt}$ represents differentiation along the trajectories given by

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{u}_0.$$

Expanding ϕ in a Taylor series along a trajectory, we have

$$\phi(\boldsymbol{x}(t+\Delta t), t+\Delta t) = \phi(\boldsymbol{x}(t), t) + \sum_{r=1}^{\infty} \frac{\Delta t^r}{r!} \left(\frac{d}{dt}\right)^r \phi(\boldsymbol{x}(t), t). \tag{6.6}$$

Since the field, ϕ, is quite general, (6.5) provides the identity

$$\left(\frac{d}{dt}\right)^r = (-\boldsymbol{u}'.\boldsymbol{\nabla})^r .$$

When this is substituted into (6.6) and truncated after the first two terms, we obtain the semi-Lagrangian, Lax-Wendroff method:

$$\phi(\boldsymbol{x}_i, t+\Delta t) = \phi(\boldsymbol{x}_i - \boldsymbol{u}_0\Delta t, t) + \left[\sum_{r=1}^{2} \frac{\Delta t^r}{r!}(-\boldsymbol{u}'.\boldsymbol{\nabla})^r\right] \phi(\boldsymbol{x}_i - \boldsymbol{u}_0\Delta t, t) + \mathrm{O}(\Delta t^2). \quad (6.7)$$

The spatial derivatives occurring in (6.7) are represented by centred differences to obtain the complete scheme.

6.3 Smolarkiewicz et al.

6.3.1 Stokes' Theorem

Smolarkiewicz and Pudykiewicz [53] carry the development of non-interpolating SL schemes a stage further. They offer an approach which allows the Eulerian and Lagrangian aspects of the scheme to be clearly separated. This non-interpolating framework builds on the work of Smolarkiewicz and Rasch [54] and Smolarkiewicz and Grell [52].

Consider the linear advection equation,

$$\frac{\partial \phi}{\partial t} + \boldsymbol{u}.\boldsymbol{\nabla}\phi = \mathcal{R}. \quad (6.8)$$

Stokes' Theorem, on the time-space domain, is applied to the field ϕ, to connect data at different points in the domain with integrals along arbitrary contours. If $\boldsymbol{x}_i$ is a grid point and $\boldsymbol{x}^*$ is some arbitrary point, then Stokes' Theorem provides an integral representation of (6.8):

$$\phi(\boldsymbol{x}_i, t^n + \Delta t) = \phi(\boldsymbol{x}^*, t^n) + \int_{\mathcal{C}} \left(\frac{\partial \phi}{\partial t}, \boldsymbol{\nabla}\phi\right), \quad (6.9)$$

where $\mathcal{C}$ is any arbitrary contour, connecting $(\boldsymbol{x}^*, t^n)$ with $(\boldsymbol{x}_i, t^n + \Delta t)$. Using (6.8), we can eliminate the time derivative in (6.9) to produce

$$\phi(\boldsymbol{x}_i, t^n + \Delta t) = \phi(\boldsymbol{x}^*, t^n) + \int_{\mathcal{C}} \boldsymbol{\nabla}.(d\boldsymbol{x} - \boldsymbol{u}dt) + \int_{\mathcal{C}} \mathcal{R}\ dt. \quad (6.10)$$

In order to obtain numerical schemes from (6.10), we must select a point, $\boldsymbol{x}^*$, and a contour, $\mathcal{C}$.

There are three possibilities for $\boldsymbol{x}^*$ which lead to numerical schemes:

(i) $\boldsymbol{x}^* = \boldsymbol{x}_i$: Eulerian schemes

(ii) $\boldsymbol{x}^* = \boldsymbol{x}_d$: Lagrangian schemes

(iii) $\boldsymbol{x}^* = [\boldsymbol{x}_d]$: non-interpolating SL schemes

where $\boldsymbol{x}_d$ is the departure point, at time level t^n, for the parcel trajectory which passes through $(\boldsymbol{x}_i, t^n + \Delta t)$; and $[\boldsymbol{x}_d]$ is the grid point nearest to $\boldsymbol{x}_d$. Making the particular choice $\boldsymbol{x}^* = [\boldsymbol{x}_d]$, we now look at how the contour, $\mathcal{C}$, might be chosen to produce numerical schemes.

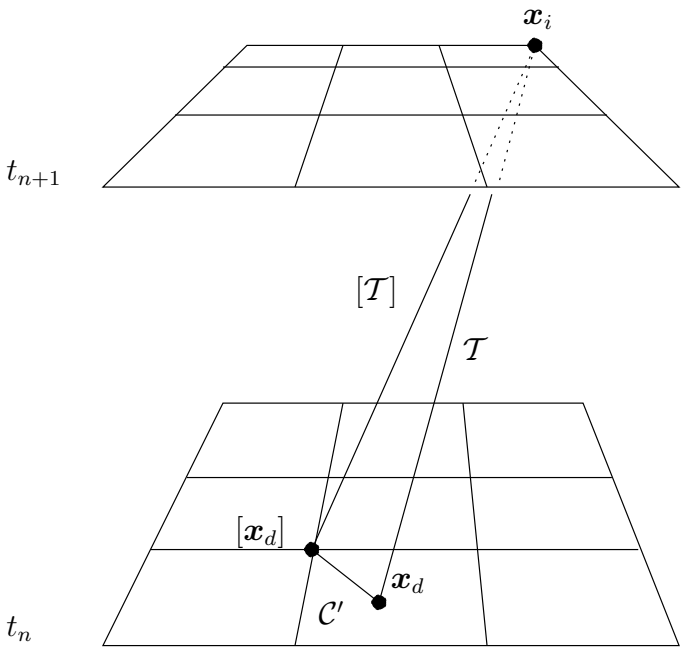

Figure 6.2: Contour Decomposition

6.3.2 Contour Decomposition

The contour $\mathcal{C}$ is effectively a route along which information held at $([\boldsymbol{x}_d], t^n)$ is to be transferred to $(\boldsymbol{x}_i, t^n + \Delta t)$. Smolarkiewicz and Rasch [54] investigated two classes of

contours. By introducing a little more notation, (see Fig. 6.2), we can write these as:

(A) $\mathcal{C} = \mathcal{C}' \bigcup \mathcal{T}$, where $\mathcal{C}'$ is an arbitrary contour in the $t = t^n$ plane, connecting $[\boldsymbol{x}_d]$ to $\boldsymbol{x}_d$; and $\mathcal{T}$ is the fluid parcel trajectory from $(\boldsymbol{x}_d, t^n)$ to $(\boldsymbol{x}_i, t^n + \Delta t)$.

(B) $\mathcal{C} = [\mathcal{T}]$, where $[\mathcal{T}]$ is the straight line trajectory from $([\boldsymbol{x}_d], t^n)$ to $(\boldsymbol{x}_i, t^n + \Delta t)$.

Contour (B) leads to a class of schemes which includes Ritchie's scheme. It can be interpreted as a local coordinate transformation, which establishes a new moving frame of reference displacing grid points to grid points, over one time step. However, for reasons elaborated by Smolarkiewicz and Rasch, contour (B) poses certain design problems if we wish to derive second order time accurate schemes. Selecting contour (A), Smolarkiewicz and Pudykiewicz produce a broad class of non-interpolating schemes, which we now consider.

With the choice of contour (A), equation (6.10) becomes

$$\phi(\boldsymbol{x}_i, t^n + \Delta t) = \phi([\boldsymbol{x}_d], t^n) + \int_{\mathcal{C}'} \boldsymbol{\nabla}\phi . d\boldsymbol{x} + \int_{\mathcal{T}} \mathcal{R}\ dt. \tag{6.11}$$

Here we have used a few simple facts about the chosen contour. For the first integral in (6.10), $d\boldsymbol{x} - \boldsymbol{u}dt = 0$ along the parcel trajectory $\mathcal{T}$; and $dt = 0$ along $\mathcal{C}'$, since $\mathcal{C}'$ lies in the plane $t = t^n$.

We now have one last arbitrary quantity to fix, namely the little contour $\mathcal{C}'$. Two particular options can be identified immediately, both of which lead to numerical schemes.

(a) $\mathcal{C}'$ = linear contour joining $([\boldsymbol{x}_d], t^n)$ directly to $(\boldsymbol{x}_d, t^n)$, and

(b) $\mathcal{C}'$ = sum of linear contours parallel to the coordinate axes.

We shall first consider contour (a). If we use s to parameterize the contour, so that $s = t^n$ corresponds to the point $([\boldsymbol{x}_d], t^n)$ and $s = t^n + \Delta t$ corresponds to $(\boldsymbol{x}_d, t^n)$, then (6.11) becomes

$$\phi(\boldsymbol{x}_i, t^n + \Delta t) = \phi([\boldsymbol{x}_d], t^n) - \int_{t^n}^{t^n + \Delta t} \boldsymbol{\nabla}.(\phi \boldsymbol{U})([\boldsymbol{x}_d], s)\ ds + \int_{\mathcal{T}} \mathcal{R}\ dt, \tag{6.12}$$

where $\boldsymbol{U} = \dfrac{[\boldsymbol{x}_d] - \boldsymbol{x}_d}{\Delta t}$. If we ignore the source term $\mathcal{R}$ for the moment, we can see that (6.12) is the formal integral, between $s = t^n$ and $s = t^n + \Delta t$, of the constant coefficient advection equation

$$\frac{\partial \phi}{\partial s} + \boldsymbol{\nabla}.(\phi \boldsymbol{U}) = 0. \tag{6.13}$$

There are many Eulerian schemes available for the solution of such equations. In particular there are classes of schemes which possess desirable properties such as monotonicity and shape-preservation ([4], [62], [57], [12], [11]). If, together with some suitable discretisation of the source term, we apply such a scheme to (6.12) then we finally arrive at a non-interpolating SL scheme.

If we were to choose contour (b) above, then the residual equation (6.13) would be replaced with a set of one dimensional advection equations: one equation for each space dimension. Applying a one dimensional Eulerian scheme to each of these would result in a dimensional splitting scheme.

Through the above approach, we have a means of extending any Eulerian scheme into a SL scheme. This allows the CFL stability limit of Eulerian schemes to be relaxed. Through our choice of contours, (b) or (a) respectively, we can combine one dimensional schemes for use in higher space dimensions, or use fully multidimensional Eulerian schemes. The generalized scheme inherits properties of shape-preservation from the original Eulerian scheme. However, conservative Eulerian schemes do not lead to conservative SL schemes using this approach.

This approach to constructing non-interpolating schemes replaces the conventional interpolation operators of SL schemes with Eulerian advection schemes. This equivalence between interpolation and numerical advection with finite differences is discussed in considerable detail by Smolarkiewicz and Grell [52].

6.4 Non-interpolating Trajectory Methods

In Chapter 1 we described the implicit mid-point rule for solving the trajectory equation in semi-Lagrangian schemes. This method requires interpolation of the finite difference data representing the velocity field. To eliminate interpolation from all aspects of SL schemes, Smolarkiewicz and Pudykiewicz [53] apply the advection-interpolation equivalence to solving the trajectory equation. With an Eulerian advection scheme in place of a more conventional interpolation operator, such methods may be considered non-interpolating.

A more elegant route to removing computationally expensive interpolations from trajectory calculations, has been suggested by McGregor [30]. In this approach to calculating departure points, the trajectory equation is no longer solved using the implicit mid-point rule. Instead, finite difference approximations of a truncated Taylor series expansion are used.

If the trajectory equation is

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{u}(\boldsymbol{x}, t),$$

where $\boldsymbol{x}(t)$ is the position along the trajectory at time t, and $\boldsymbol{u}$ is the velocity, then a Taylor series expansion of $\boldsymbol{x}(t_n)$ about time $t_n + \Delta t$ may be truncated to give

$$\boldsymbol{x}(t_n) \approx \boldsymbol{x}(t_n + \Delta t) + \sum_{r=1}^{M} \frac{(-\Delta t)^r}{r!} \frac{d^r \boldsymbol{x}}{dt^r}(t_n + \Delta t).$$

In this approximation the Taylor expansion has been truncated after M terms. The summation involves terms all to be evaluated at the forward time level, data which is not available directly from the finite difference solution computed to time t_n. Further approximations are therefore necessary. Use is made of the formal equivalence between the Eulerian convective derivative and the Lagrangian time derivative,

$$\frac{d}{dt} \equiv \frac{\partial}{\partial t} + \boldsymbol{u}.\boldsymbol{\nabla}.$$

An approximate form of this is

$$\frac{d}{dt} \approx \hat{\boldsymbol{u}}(\boldsymbol{x}(t_n + \Delta t, t_n + \frac{1}{2}\Delta t).\boldsymbol{\nabla}.$$

The new velocity, $\hat{\boldsymbol{u}}$, in this equation is located at the arrival point $\boldsymbol{x}(t_n + \Delta t)$, which we are here assuming to be a grid point as discussed in Chapter 1. It is to be evaluated mid-way through the time step for second order time accuracy. This may be accomplished by the use of time extrapolation, again using the formulae given in Chapter 1. For the higher order derivative terms, centred differencing may be used. The advantages of this scheme are that it does not require complicated interpolations, and all the calculations may be carried out relatively quickly.

By calculating the product $\hat{\boldsymbol{u}}.\boldsymbol{\nabla}$ in spherical polar coordinates, McGregor demonstrates that his scheme is applicable to advection schemes on the sphere. In the next chapter we examine the applicability of SL methods in general to global weather simulation, where the issues arising from spherical geometry must be addressed.

Chapter 7

Nonlinear Advection

Introduction

Up to this point we have only considered linear advection. Referring back to Section 1.3, we see that the semi-Lagrangian method must also be applied to nonlinear advection equations. In this chapter we investigate the behaviour of SL schemes when applied to the Burgers' equation. This equation provides a simple one-dimensional analogue of the nonlinear equation for momentum, which occurs in mathematical models of fluid flow.

7.1 Burgers Equation

The viscous form of Burgers' equation is

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \varepsilon\frac{\partial^2 u}{\partial x^2}, \tag{7.1}$$

where u is the flow velocity and ε a viscosity parameter. The left-hand side of this equation represents the advection of the velocity field. Advection of the field $u(x,t)$, by the velocity $u(x,t)$ gives rise to the nonlinear term in this equation.

We shall also be interested in the inviscid form of Burgers' equation,

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0, \tag{7.2}$$

which is obtained from (7.1) by setting $\varepsilon = 0$. While solutions of the viscous Burgers' equation are always smooth, the inviscid Burgers' equation is derived from an integral conservation law which admits shocks. Kuo and Williams [19] found that the semi-Lagrangian technique can be successfully applied to the inviscid Burgers' equation, in the context of stationary shock formation. Numerical schemes exist which are designed specifically to correctly handle such discontinuous solutions in conservation laws, [24]. We briefly summarise the important considerations in the design of such schemes.

First, it should be observed that solutions of the differential equation (7.2) must have well-defined first derivatives. However this equation, where it occurs in any physical application, will have been derived from an integral conservation law. The solutions of such integral equations need not be continuous. Hence there is a problem for any numerical scheme which attempts to solve the conservation law using the derivative form, (7.2). Suitable numerical methods may be obtained by integrating (7.2) as a product with a smooth test function. Solutions obtained in this way are termed *weak* solutions. A further problem arises from this approach, namely that weak solutions are not necessarily unique. Extra constraints are required. in order that the correct physical solution is obtained. Such constraints are obtained by applying further physical principles, in addition to the conservation law. These constraints are generally termed *entropy conditions.* Finally we note that physically realistic solutions are obtained for the inviscid Burgers' equation from solutions of the viscous equation (7.1) in the limit as ε approaches zero.

In view of the above considerations, we see that the inviscid Burgers' equation presents a significant challenge to the semi-Lagrangian schemes so far described. Our purpose here is not to derive SL schemes to meet this challenge. Rather, we shall apply the SL scheme to equations (7.1) and (7.2) in the spirit of destructive testing. Since the SL method is used for discretising the momentum equation in NWP models, it is important to understand their behaviour when applied to nonlinear advection.

7.2 Setting up Computational Tests

Two sets of test problems will be considered, one with the viscous Burgers' equation and one with the inviscid equation. We begin by describing the experimental set-up which we shall apply to both the viscous and inviscid Burgers' equation.

For both sets of test problems the solution is computed on the domain

$$0 \leq x \leq 10.$$

Fixed boundary conditions are imposed, which are set by the initial data.

A semi-implicit semi-Lagrangian discretisation is used for the viscous Burgers' equation,

$$\frac{u^{n+1} - u_d^n}{\Delta t} = \alpha\varepsilon\delta_x^2 u^{n+1} + (1-\alpha)\varepsilon\left(\delta_x^2 u^n\right)_d, \tag{7.3}$$

where Δt is the timestep and α is the implicitness parameter. The discretisation of the inviscid equation is simply

$$u^{n+1} = u_d^n. \tag{7.4}$$

It is apparent from an examination of (7.3) and (7.4), that the SL method does not require discretisation of the nonlinear terms. The nonlinearity is subsumed within the departure point calculations and the interpolation of u to those departure points. We therefore expect to see, in the following results, some sensitivity to the choice of departure point and interpolation schemes.

Initially each test problem is solved using cubic Lagrange interpolation, for evaluating quantities at departure points. The departure points are calculated using the implicit mid-point method, with time extrapolation of the velocity and linear interpolation to trajectory mid-points. This we term the standard SL scheme. Changes to the standard scheme are then made by changing the form of interpolation and changing the departure point scheme.

7.3 Inviscid Burgers' Test Problem

Initial data for the inviscid problem is chosen to lead to the formation of a shock in the solution after a finite time. Up until this time the evolution of the solution is smooth. This provides a test of the time accuracy of the SL scheme in the early, smooth stages of the flow. At around the time of the shock formation, the design assumptions of the SL scheme no longer hold. At this point the numerical solution is liable to fail in some way.

The initial data for the inviscid problem has the form

$$u(x,0) = \begin{cases} u_{min} + (u_{max} - u_{min}) \cos^2 \left[\frac{\pi}{2} \left(\frac{x - x_c}{a} \right) \right], & |x - x_c| \leq a \\ u_{min}, & |x - x_c| > a, \end{cases} \tag{7.5}$$

with parameter values

$$\begin{aligned} u_{min} &= 0.0 \\ u_{max} &= 1.0 \\ x_c &= 3.0 \\ a &= 2.0. \end{aligned}$$

A grid of 200 equally spaced mesh intervals is used, giving a mesh length $\Delta x = 0.05$. For this grid there are eighty mesh intervals covering the central feature of the initial data. For selecting the time step, the important parameter to consider is the ratio of temporal and spatial mesh intervals,

$$\nu = \frac{\Delta t}{\Delta x}. \tag{7.6}$$

For the current problem, this mesh ratio parameter corresponds to the maximum Courant number at the initial time.

7.3.1 Reference Solution

A reference solution is obtained using a numerical scheme specifically designed to converge on the correct weak solution. The accuracy of SL solutions is assessed by comparison

against this reference solution.

The scheme used for the reference solution is Roe's first order upwind scheme. For the general conservation law,

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x} f(u) = 0, \tag{7.7}$$

this scheme takes the form

$$u_j^{n+1} = u_j^n - \frac{\Delta t_R}{\Delta x_R} \left(\hat{f}_{j+\frac{1}{2}} - \hat{f}_{j-\frac{1}{2}} \right), \tag{7.8}$$

where Δt_R and Δx_R are the time and space mesh intervals; the numerical flux function, $\hat{f}$, is

$$\hat{f}_{j+\frac{1}{2}} = \frac{1}{2}[f(u_j^n) + f(u_{j+1}^n)] - \frac{1}{2} |A| (u_{j+1}^n - u_j^n) \tag{7.9}$$

and

$$A = \frac{df}{du}. \tag{7.10}$$

This scheme is in *conservation form*. It is this which guarantees that its solutions will converge, under mesh refinement, to a weak solution of the differential equation (7.7), [21, 24].

Applying Roe's scheme to the inviscid Burgers' equation, the flux function is

$$f(u) = \frac{1}{2} u^2. \tag{7.11}$$

This leads to the numerical flux function

$$\hat{f}_{j+\frac{1}{2}} = \frac{1}{4} \left[(u_j^n)^2 + (u_{j+1}^n)^2 - \left| u_j + u_{j+1}^n \right| (u_{j+1}^n - u_j^n) \right]. \tag{7.12}$$

In order to obtain sufficient accuracy in the reference solution, (7.8) is solved on a higher resolution grid than is used for the SL scheme:

$$\Delta x_R = 0.1 \Delta x. \tag{7.13}$$

The time step for Roe's scheme must be chosen to respect a Courant number condition; we choose

$$\frac{\Delta t_R}{\Delta x_R} \approx 0.1, \tag{7.14}$$

with the precise value being chosen to ensure that an integer number of Roe integration steps is made for each SL step.

7.3.2 Time of Shock Formation

The method of characteristics may be used to calculate the time at which the shock first forms. Consider the inviscid Burgers' problem,

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0, \quad u(x,0) = u_0(x). \tag{7.15}$$

We wish to solve this for $u(x,t),\ t > 0$. This may be done by considering characteristics.

The characteristics of (7.15) are the solution curves of

$$\frac{dx}{dt} = u(x(t),t). \tag{7.16}$$

Along a characteristic, the time rate of change of u is

$$\begin{aligned} \frac{du}{dt} &= \frac{\partial u}{\partial t} + \frac{dx}{dt}\frac{\partial u}{\partial x} \\ &= \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} \\ &= 0, \end{aligned} \tag{7.17}$$

where we have made use first of (7.16) then of (7.15). Hence u is constant in time along a characteristic. Consequently, the characteristic equation (7.16) may be written as,

$$\frac{dx}{dt} = u_0(x_0), \tag{7.18}$$

where x_0 is the position at time $t = 0$ of the characteristic which passes through the point x at time t. Integrating (7.18), we have

$$x = x_0 + tu_0(x_0). \tag{7.19}$$

If this last equation can be solved for x_0 then the original problem (7.15) is solved,

$$u(x,t) = u_0(x_0). \tag{7.20}$$

Even if (7.19) is not amenable to analytical solution, which depends on the nature of the initial data u_0, it still provides a means of determining when and where a shock will form. A shock forms where characteristics first cross. In order to determine this point,

we treat (7.19) as a family of curves, parametrised by x_0. The set of intersections of curves belonging to a family is termed the envelope of the family of curves. For a general family of curves, $\phi(x, y; \lambda) = 0$, with parameter λ, the envelope is found by simultaneously solving

$$\begin{aligned} \phi(x, y; \lambda) &= 0 \qquad (7.21) \\ \frac{\partial \phi}{\partial \lambda} &= 0. \qquad (7.22) \end{aligned}$$

For initial data defined by (7.5), the characteristics are parallel straight lines, except those for which the parameter x_0 lies in the range $[x_c - a, x_c + a]$. Without loss of generality, we set $x_c = 0$ for the purposes of calculating the shock time. The characteristics are given by

$$x_0 + u_{min}t + (u_{max} - u_{min})t \cos^2\left[\frac{\pi x_0}{2a}\right] - x = 0. \qquad (7.23)$$

Differentiating (7.23) with respect to the parameter x_0, we obtain

$$1 - (u_{max} - u_{min})t\frac{\pi}{2a}\sin\left(\frac{\pi x_0}{a}\right) = 0. \qquad (7.24)$$

Hence, on the envelope of the characteristics,

$$t = \frac{2a}{\pi(u_{max} - u_{min})\sin[\pi x_0/a]}, \quad -a \leq x_0 \leq a. \qquad (7.25)$$

For $t > 0$ the minimum value of t on this curve occurs when

$$x_0 = \frac{a}{2}, \qquad (7.26)$$

from which we obtain the shock time,

$$t_s = \frac{2a}{\pi(u_{max} - u_{min})}. \qquad (7.27)$$

Using (7.23) and (7.27), we see that the shock occurs at position

$$x_s = \frac{a}{2} + \frac{a(u_{max} + u_{min})}{\pi(u_{max} - u_{min})}. \qquad (7.28)$$

So far we have only used part of the initial data in calculating the shock position and time. If the above computed value of x_s is greater than a, then some of the characteristics

with $-a \leq x_0 \leq a$ will have crossed characteristics with $x_0 > a$ at some earlier shock time. Even if this is not the case, it is still necessary to check that characteristics from these two regions of initial data do not cross earlier than time t_s. For the case $u_{min} = 0$ we find that

$$x_s = \left(\frac{\pi + 2}{2\pi}\right) a < a. \tag{7.29}$$

Finally, a graphical check may be made to ensure that no characteristic with x_0 in the range $[-a, a]$ has reached $x = a$ by time t_s. Here we simply show the plausibility of t_s being the earliest shock time, as follows. The position of each of these characteristics at time $t = t_s$ is given by

$$x(t_s) = x_0 + \frac{2a}{\pi} \cos^2\left(\frac{\pi x_0}{2a}\right), \quad -a \leq x_0 \leq a. \tag{7.30}$$

Expanding (7.30) in a power series about $x_0 = a$,

$$x(t_s) = a - (a - x_0) - \frac{\pi}{2a}(a - x_0)^2 + \mathrm{O}((a - x_0)^4). \tag{7.31}$$

Hence characteristics close to $x_0 = a$ have not reached $x = a$ by time t_s. This suggests that the crossing of characteristics at position x_s, time t_s, is the earliest time at which a shock occurs.

7.3.3 Results

Inviscid Test 1: Standard Scheme.

Semi-Lagrangian schemes are of interest precisely because they allow long time steps to be used. Setting

$$\nu = 1.70,$$

the standard SL scheme produces the results shown in Figure 7.1. For comparison, the same problem is solved using Roe's scheme and is shown by the dashed lines. The solution is plotted at intervals of ten timesteps for a total of fifty timesteps. The semi-Lagrangian solution follows the reference solution remarkably well as it slumps towards the right and

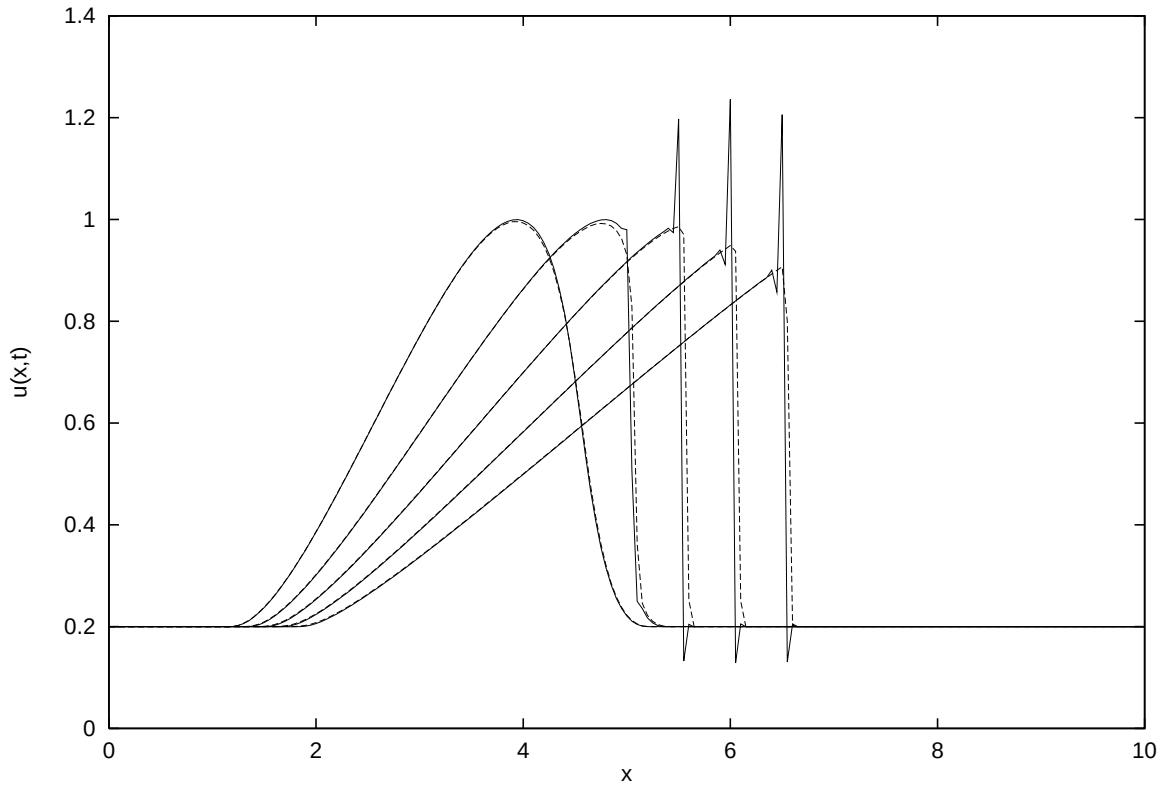

Figure 7.1: Solution of inviscid Burgers' equation using standard SL scheme: cubic interpolation; departure points calculated using time extrapolated velocity, four iterations of implicit mid-point method. Dashed lines show solution obtained by Roe's scheme on a finer mesh.

forms a shock. Once the shock has formed, however, sharp, spurious spikes enter the SL solution. These are due to the lack of smoothness in the solution around the shock.

It would appear from this first result that the standard SL method is giving a solution which is qualitatively correct: it gives an accurate solution while the flow is smooth and has a shock moving at more or less the correct speed. However this result is due mainly to a favourable choice of parameters. If we change the initial data by setting $u_{min} = 0$, then the result is as shown in Figure 7.2. The behaviour of the standard SL scheme is now quite different. The spikes at the shock position are much reduced, but now the shock speed is entirely wrong. Having verified that the standard SL scheme only solves the inviscid Burgers' equation up to the time of shock formation, we look more closely at

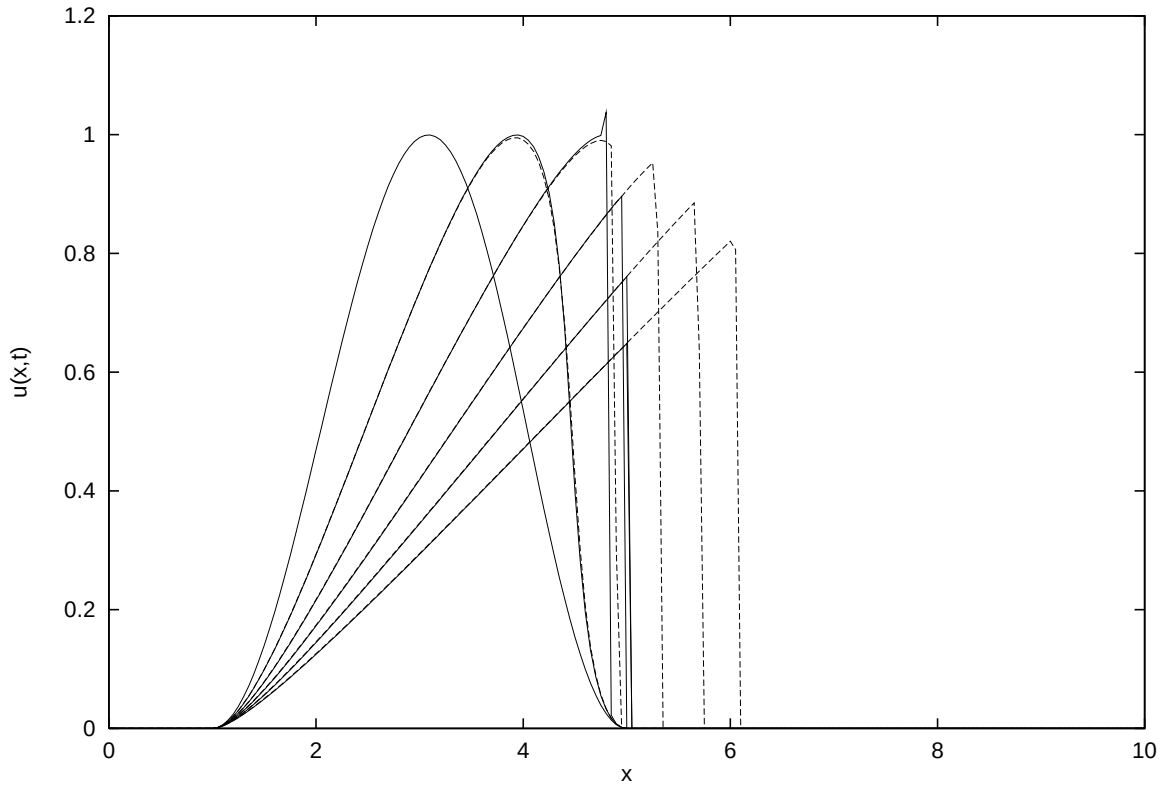

Figure 7.2: Solution of inviscid Burgers' equation using standard SL scheme: as Figure 7.1 except $u_{min} = 0$ in initial data.

the scheme's accuracy during this smooth stage of the flow.

Inviscid Test 2: Accuracy of Solution before Shock Formation.

The initial data parameters are now set as:

$$\begin{aligned} u_{min} &= 0.0 \\ u_{max} &= 0.01 \\ x_c &= 4.5 \\ a &= 4.0. \end{aligned}$$

In order to pose a more severe test, both the mesh interval and the mesh ratio are doubled: $\Delta x = 0.02$, $\nu = 3.40$. For this combination of parameters the shock occurs at time $t \approx 254.65$; the timestep is $\Delta t = 0.68$. Making a total of 375 steps, and plotting the

solution after every 75 steps, the solution obtained with the standard scheme is shown in Figure 7.3.

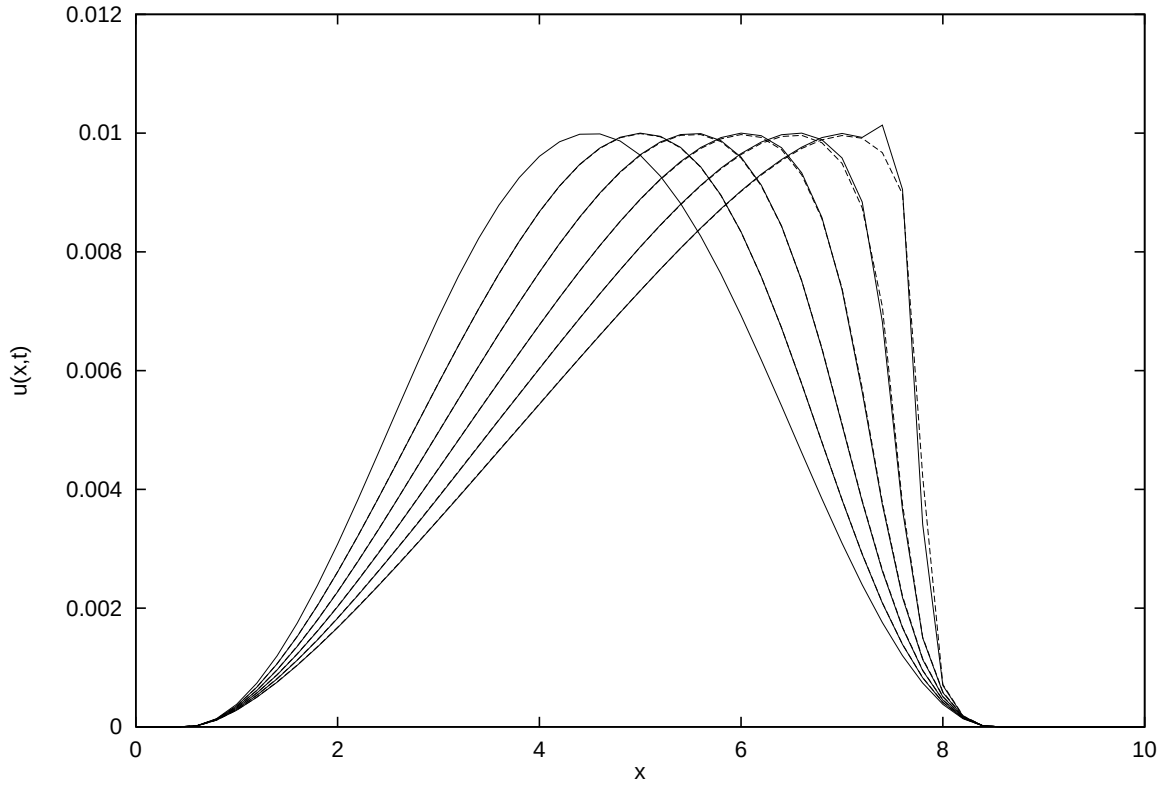

Figure 7.3: Solution of inviscid Burgers' equation using standard SL scheme: initial data and SL solution shown at 75 step intervals; reference solution in dashed line. SL solution is well-behaved for times before shock forms.

Next we define two measures of error for the numerical solutions. The first is the L2 norm of the difference between the SL solution and the reference solution,

$$e_{L2}(t_n) = \frac{\sum_{j=1}^{N}[(u_j^n)_{SL}^2 - (u_j^n)_{Roe}^2]}{N}, \tag{7.32}$$

where $(u_j^n)_{SL}$ is the SL solution for time $t_n = n\Delta t$ at position $x_j = j\Delta x$, and $(u_j^n)_{Roe}$ is the reference solution for the same time and place; N is the number of grid points in the interior of the domain. Since the SL schemes considered here do not have the numerical conservation property, it is appropriate also to consider the error in conservation,

$$e_m(t_n) = \frac{M_n}{M_0}, \tag{7.33}$$

where M_n is the numerical "mass" at time t_n, given by the summation

$$M_n = \sum_{j=1}^{N} u_j^n \, \Delta x. \tag{7.34}$$

The error measurements defined by (7.32) and (7.33) allow us to compare results obtained with different SL schemes. Figure 7.4 shows error quantities plotted against model time, for four different interpolants: the cubic Lagrange of the standard scheme; cubic ENO; the quadratic 'Fromm' interpolant, and quintic Lagrange. All four schemes behave in broadly the same manner. Each scheme performs well up until roughly time $t = 200$. Beyond this time all the solutions become increasingly erratic as the shock time is approached. The ENO scheme performs least well. This is due to the steepening of the solution near the forming shock. The ENO scheme will respond to this loss of smoothness by shifting the interpolation stencil to where the solution is smoothest. Though still formally of fourth order accuracy, the one-sided shift of the stencil leads to comparatively large interpolation errors. The quadratic 'Fromm' interpolant performs best. This may be attributed to the fact that this method requires less smoothness in the data, than do the higher order interpolants, in order to fully realise its formal order of accuracy. In general we may conclude that semi-Lagrangian schemes are capable of tracking the smooth development of the flow, almost to the point of shock formation.

Test 3: Variation of Mesh Ratio Parameter

For this group of tests, we use $u_{max} = 0.1$, for which the shock forms at approximately time $t = 25.5$. We examine the effect of varying the mesh ratio parameter on the solutions obtained. For fifty evenly spaced values of $\Delta t/\Delta x$ between 0.1 and 6.0, numerical integrations are made up to just one time step before the shock time. The maximum error during each integration is plotted against the value of the mesh ratio, Figure 7.5. We see that the quadratic Fromm scheme performs better than both the cubic and quintic Lagrange schemes, at all values of the mesh ratio. The cubic Lagrange scheme is more accurate than the quintic scheme, for mesh ratios below about 2.5. Beyond this point the quintic scheme gives results which are more accurate than those with the cubic scheme.

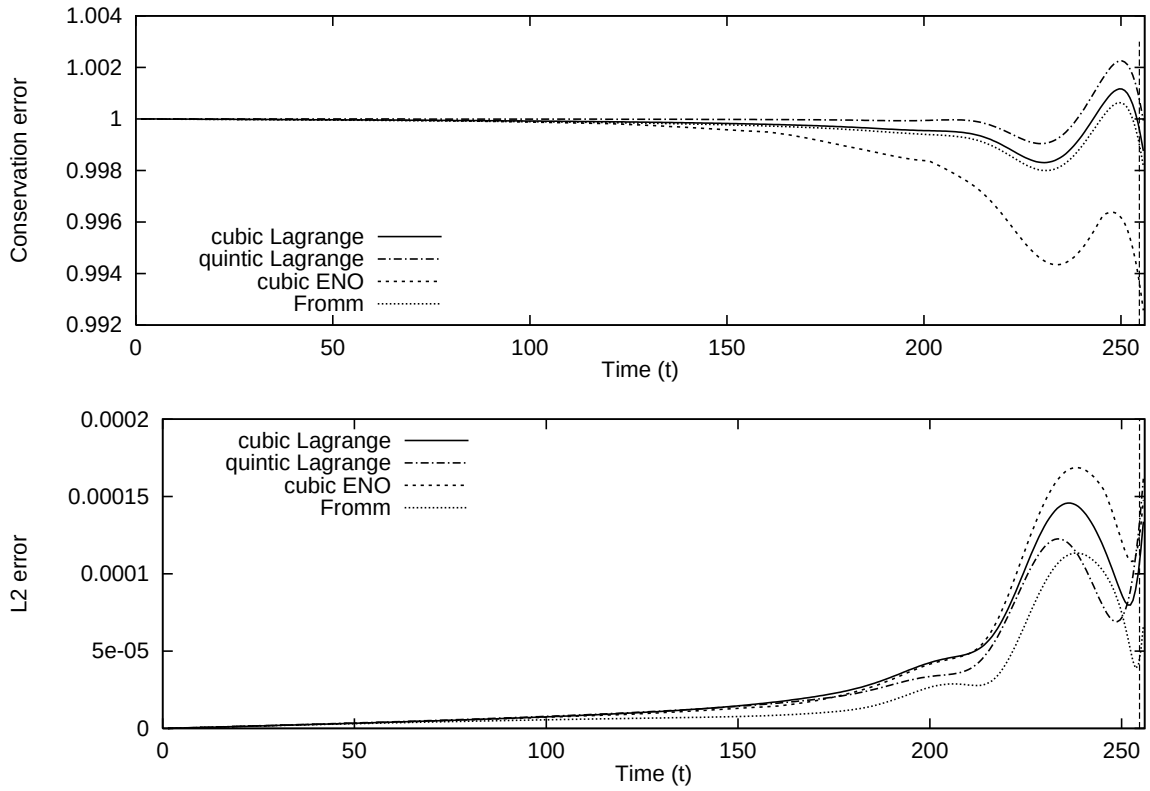

Figure 7.4: Plots of L2 error and conservation error against time for SL solution of inviscid Burgers' equation. Results shown for four different interpolation methods.

Next the integration is run up to time $t = 20$, which is just under eighty percent of the shock time. At this time the spurious behaviour as the shock is approached has not yet begun. This can be seen in Figure 7.4, where each method becomes wildly erratic only after time $t = 200$, which corresponds to $t = 20$ in the current tests. As may be seen in Figure 7.6, the quadratic scheme still performs better than the higher order methods.

7.4 Viscous Burgers' Test Problem

The viscous Burgers' equation (7.1 is discretised using the semi-implicit, semi-Lagrangian discretisation (7.3). This discretisation provides a linear system of equations for the finite difference data at the new time level. Applying index notation to (7.3), this system of

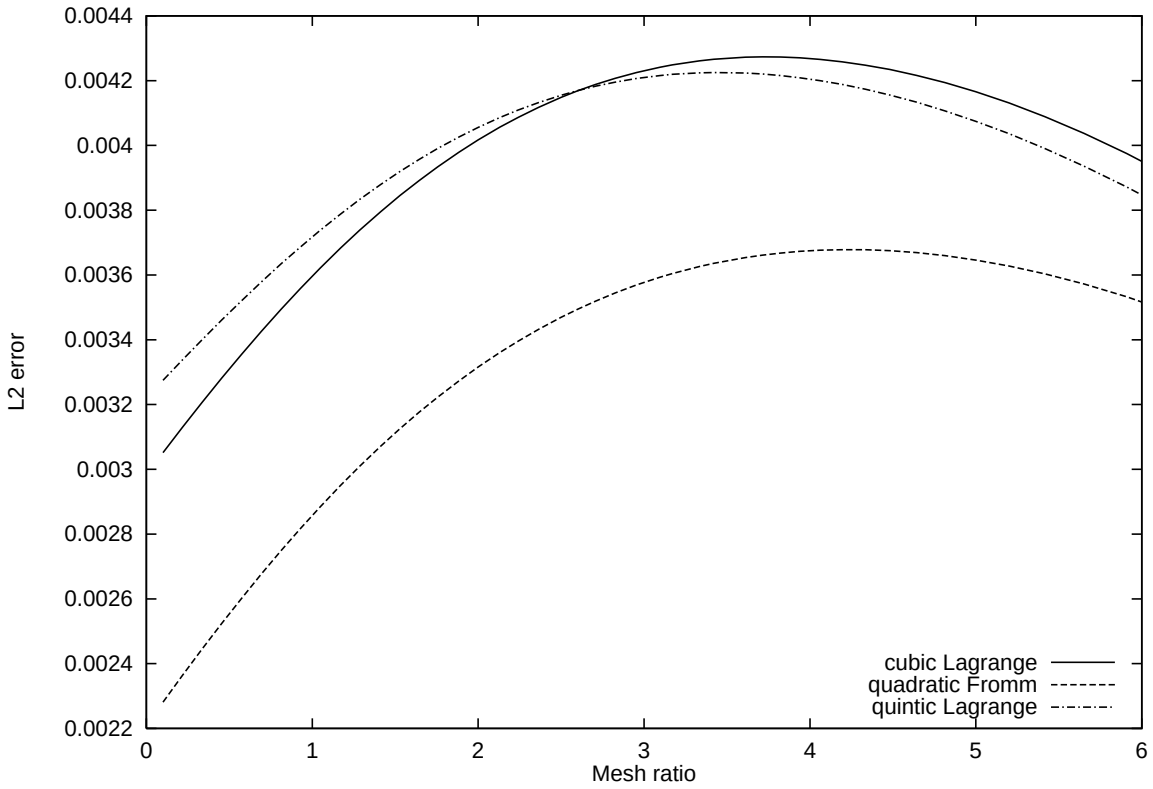

Figure 7.5: Maximum error against mesh ratio parameter. The maximum error is the largest error for inviscid Burgers simulation which runs to just before the time of shock formation.

equations may be written

$$(1+2\alpha\mu)\,u_1^{n+1} - \alpha\mu\,u_2^{n+1} = \alpha\mu\,u_0(x_0) + R_1 \tag{7.35}$$

$$-\alpha\mu\,u_{j-1}^{n+1} + (1+2\alpha\mu)\,u_j^{n+1} - \alpha\mu\,u_{j+1}^{n+1} = R_j, \quad j = 2,\cdots,N-1 \tag{7.36}$$

$$\alpha\mu\,u_{N-1}^{n+1} + (1+2\alpha\mu)\,u_N^{n+1} = \alpha\mu\,u_0(x_{N+1}) + R_N, \tag{7.37}$$

where

$$R_j = \left[u^n + (1-\alpha)\mu\,\delta_x^2 u^n\right]_{d(j)}, \tag{7.38}$$

subscript $d(j)$ denotes interpolation to the departure point of node $x(j)$, and the mesh Peclet number is

$$\mu = \frac{\varepsilon\Delta t}{\Delta x^2}. \tag{7.39}$$

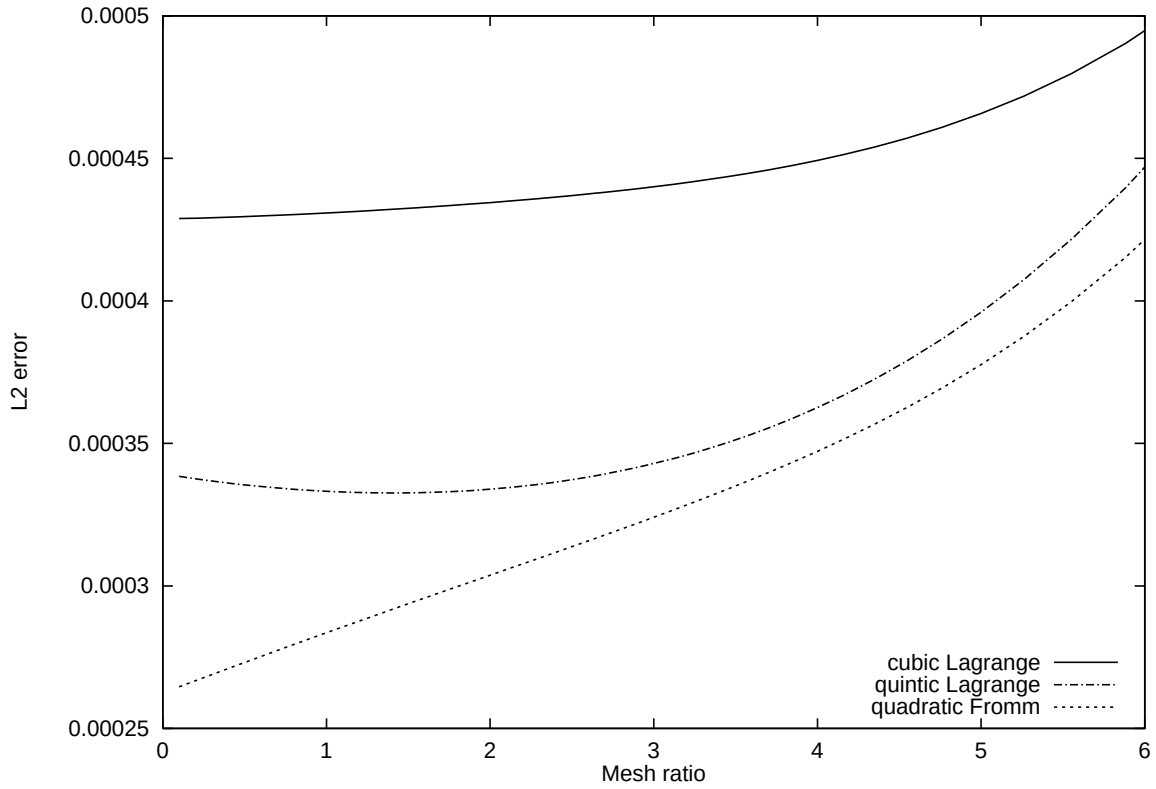

Figure 7.6: Maximum error against mesh ratio, for SL solution of inviscid Burgers' problem. The integration extends only to eighty percent of the shock time.

This is a tridiagonal system of N equations for the N unknowns, u_j^{n+1}, which we solve using the Thomas algorithm.

7.4.1 An Analytical Solution

We shall use an analytical solution of the viscous Burgers' equation as the basis for testing a number of semi-Lagrangian schemes. This analytical result maybe obtained by applying the Cole-Hopf transformation to the viscous Burgers' equation. Here we outline a construction of the same solution.

The solution we seek is a travelling wave front, which connects two regions of differing values of u. Let these two different states be $u \to U_L$ as $x \to -\infty$ and $u \to U_R$ as $x \to \infty$.

For the wave to move in the positive x direction we choose

$$U_L > U_R > 0.$$

Next we look for solutions of (7.1) which are constant velocity, constant profile waves,

$$u(x,t) = f(x - st), \tag{7.40}$$

where $f(y)$ is an arbitrary function and s is the speed of the front. Substituting (7.40) into (7.1) we obtain,

$$\frac{d}{dy}\left[\varepsilon\frac{df}{dy} + sf(y) - \frac{1}{2}f(y)^2\right] = 0. \tag{7.41}$$

Integrating, we have

$$\varepsilon\frac{df}{dy} + sf(y) - \frac{f(y)^2}{2} = C, \tag{7.42}$$

where C is some constant. This constant is determined by applying the two boundary conditions, from which we obtain

$$sU_L - \frac{U_L^2}{2} = C \tag{7.43}$$

$$sU_R - \frac{U_R^2}{2} = C, \tag{7.44}$$

where we have also made the assumption that

$$\frac{\partial u}{\partial x} \to 0 \quad \text{as } x \to \pm\infty. \tag{7.45}$$

Solving (7.43) and (7.44) gives

$$C = \frac{U_L U_R}{2}. \tag{7.46}$$

Next, integrating (7.42), we have

$$f(y) = s - \beta\tanh\left[\frac{\beta}{2\varepsilon}(x - x_0)\right], \tag{7.47}$$

where x_0 is the position of the front, and $\beta^2 = s^2 - U_L U_R$ is a parameter determining the width of the front. Finally, using (7.40), the moving front solution of the viscous Burgers' equation is

$$u(x,t) = s - \beta\tanh\left[\frac{\beta}{2\varepsilon}(x - x_0 - st)\right] \tag{7.48}$$

7.4.2 Test 1: Departure Point Iterations

For the numerical tests the parameter values are as follows. For the initial data,

$$\begin{aligned} s &= 0.6 \\ \beta &= 0.4 \\ \varepsilon &= 0.01 \\ x_0 &= 0.5, \end{aligned}$$

and for the semi-Lagrangian scheme,

$$\begin{aligned} \Delta x &= 0.025 \\ \nu &= 1.70 \\ \alpha &= 0.5. \end{aligned}$$

The computational domain is $0 \leq x \leq 10$ and the grid has 400 equally spaced mesh intervals. With this choice of parameters there are roughly a dozen grid points within the width of the front. The integration is run for 335 timesteps. The total elapsed time is roughly 14 and the front moves through approximately 340 mesh intervals.

We begin by applying the standard semi-Lagrangian scheme, employing cubic Lagrange interpolation. Initially only two iterations are made to solve the implicit mid-point rule for determining trajectories. The result is shown in Figure 7.7. Increasing the number of departure point iterations to four, the result shows marked improvement, Figure 7.8. In view of this result we shall continue to use four iterations in calculating departure points, unless otherwise stated.

7.4.3 Error Measurements

In order to quantify the error in the numerical solution, we define two measures of error.

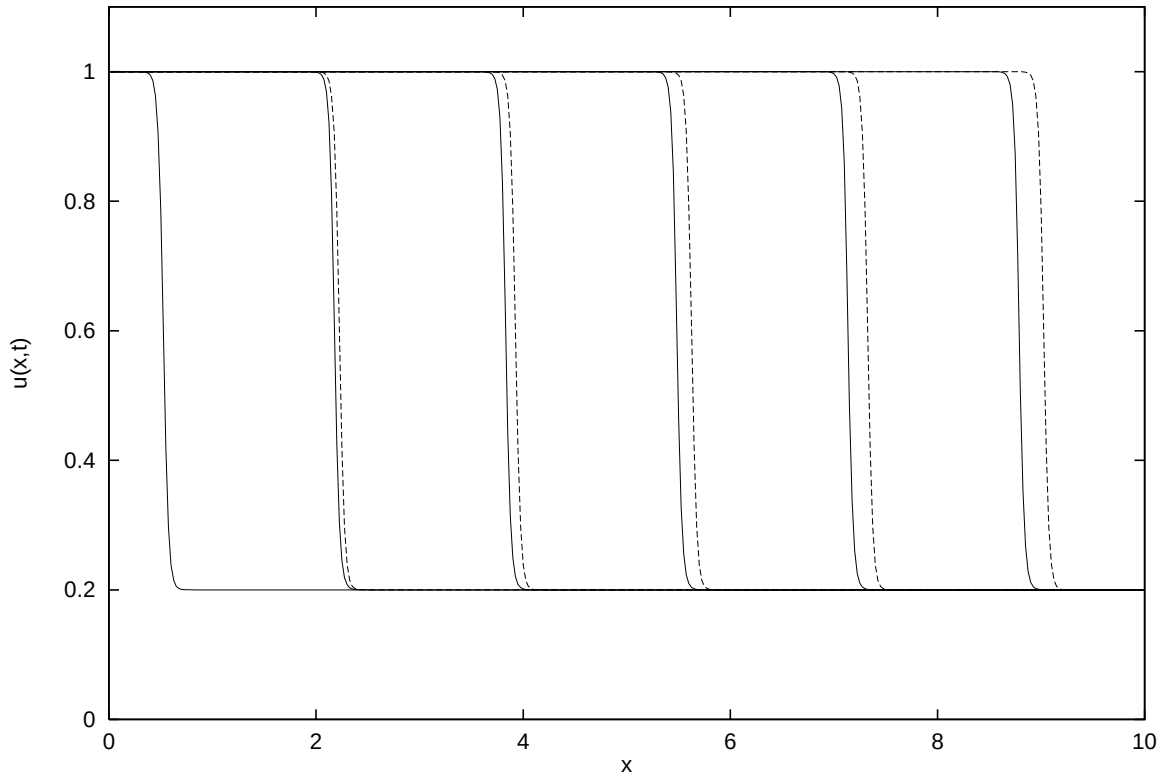

Figure 7.7: Moving front solution of viscous Burgers equation. Analytical solution (dashed lines) and SL solution (solid lines) after 0, 71, 142, 213, 284 and 355 steps. Mesh ratio $\Delta t/\Delta x = 1.7$. Two iterations for calculating departure points.

Position Error

Consider a moving front of the following form,

$$u(x,t) = \begin{cases} u_L, & x < \hat{x}_F(t) \\ \\ u_R, & x > \hat{x}_F(t), \end{cases} \tag{7.49}$$

where $\hat{x}_F(t)$ is the position of the front at time t. At time t, let x_L be a point upstream of the front and x_R be point downstream. Integrating the solution between these points, we have

$$\int_{x_L}^{x_R} u(x,t)\,dx = [\hat{x}_F(t) - x_L]\; u_L + [x_R - \hat{x}_F(t)]\; u_R. \tag{7.50}$$

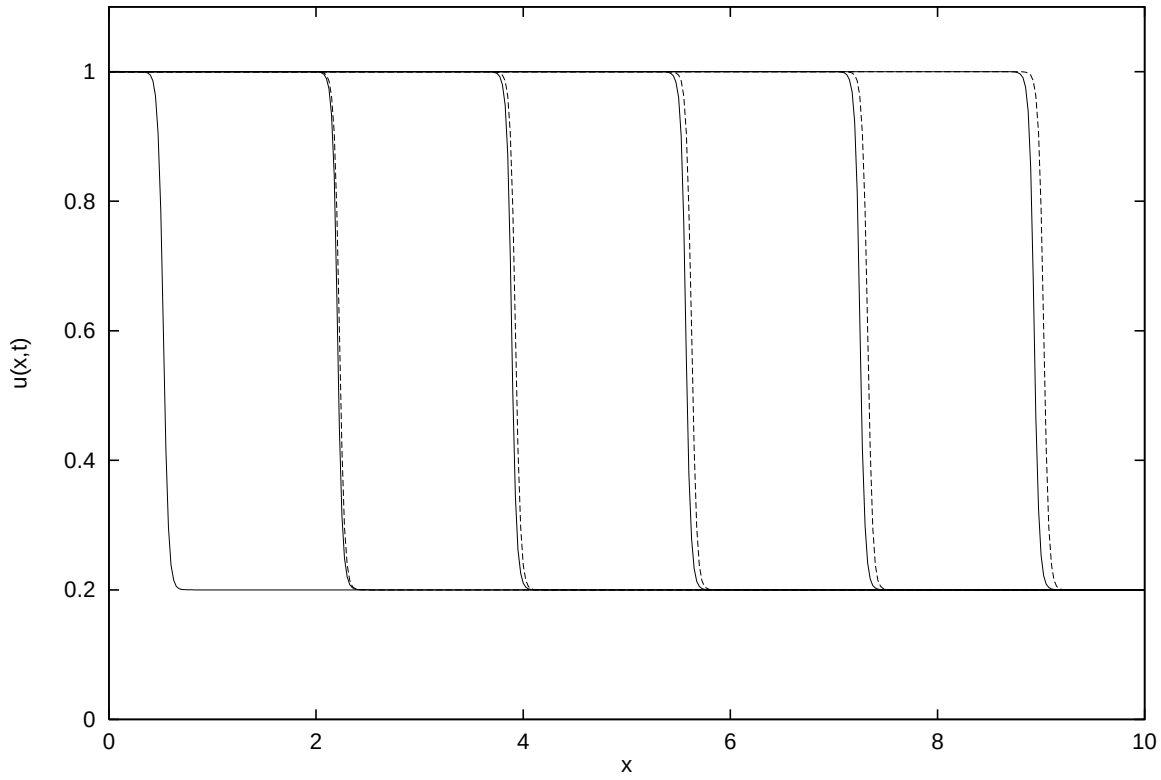

Figure 7.8: As Figure 7.7 but with 4 iterations of departure point scheme.

This may be rearranged to provide a formula for the position of the front,

$$\hat{x}_F(t) = \frac{1}{(u_L - u_R)} \left[x_L u_L - x_R u_R + \int_{x_L}^{x_R} u(x,t)\, dx \right] \tag{7.51}$$

The front defined by (7.48) has a symmetrical form and so the analytical position of the front at time t is simply,

$$\hat{x}_F = x_0 + st. \tag{7.52}$$

In defining the position of a numerically advected front, we may make use of (7.51). Provided points x_L and x_R can be found sufficiently far from the front at all times, then the formula (7.51) provides a definition for the position of a front of any shape. Using piecewise constant quadrature, (7.51) may be discretised to provide a numerical estimate of the front's position,

$$x_F(t_n) = \frac{1}{(u_L - u_R)} \left[x_L u_L - x_R u_R + \frac{\Delta x}{2} (u_L + u_R) + \sum_{j=1}^{N} u_j^n \, \Delta x \right], \tag{7.53}$$

where

$$\begin{aligned} x_L &= x_0 \\ x_R &= x_{N+1} \\ u_L &= u_0^n \\ u_R &= u_{N+1}^n. \end{aligned}$$

Using the numerical and analytical definitions for the position of the front at time t, we may calculate the position error at time t_n,

$$err_{pos}(t_n) = \frac{x_F(t_n) - \hat{x}_F(t_n)}{\Delta x}. \tag{7.54}$$

Shape Error

Once the position of a numerically advected front has been determined, we may calculate the L_2 error in the shape of the front. This we define by,

$$err_{shp}(t_n) = \left[\frac{\sum_{j=1}^{N} \left(u_j^n - u\left(x_j + \hat{x}_F(t_n) - x_F(t_n), t_n\right)\right)^2}{N} \right]^{\frac{1}{2}}. \tag{7.55}$$

This formula shifts the analytical solution to have the same front position as the numerical solution, before calculating the difference of the two solutions. It is, therefore, a measure of the error in the shape of the front and does not reflect phase errors in the numerical solution.

7.4.4 Test 2: Interpolation

In this test we apply three different forms of interpolation: cubic and quintic Lagrange, and the centred quadratic Fromm interpolation. Again running the integration for 355 steps, the resulting errors are shown in Figure 7.9.

All three interpolants give roughly the same position error. The shape error, however, does not reflect the formal accuracy of the interpolant used. Cubic interpolation gives the most accurate results for this particular case. Quintic interpolation, which is a higher

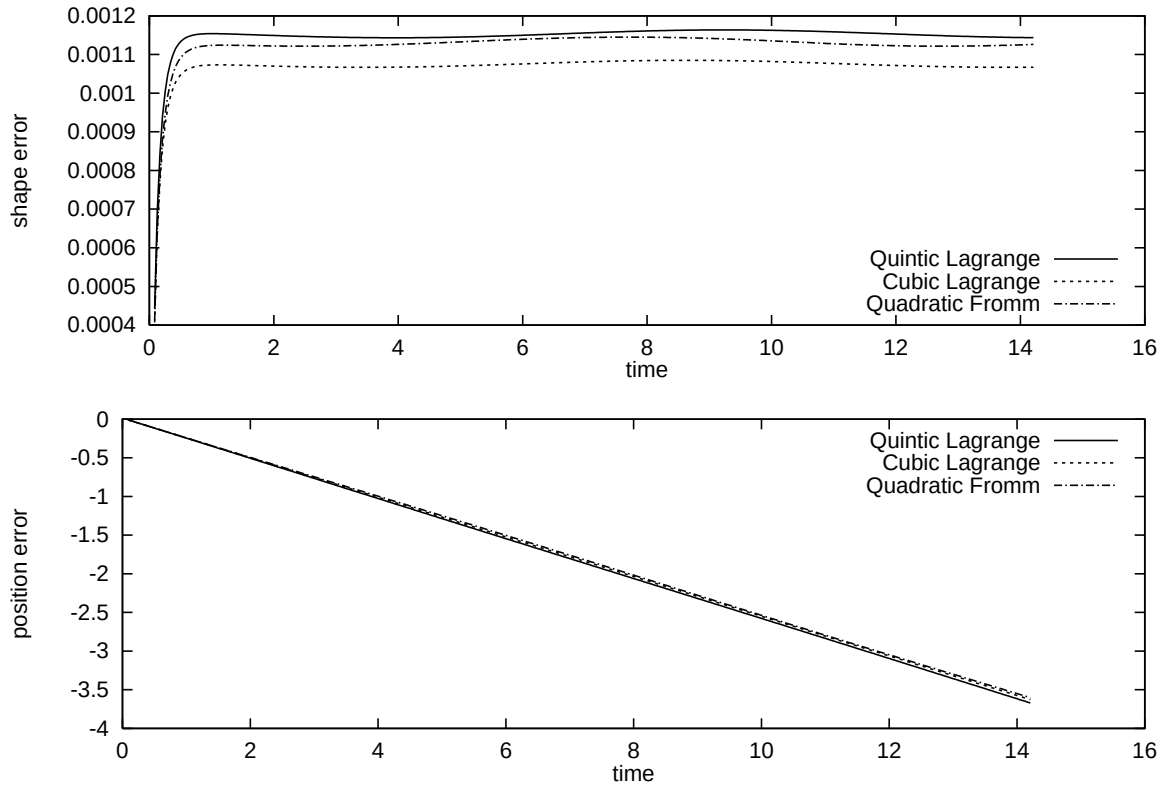

Figure 7.9: Position and shape error for SL integration of viscous Burgers' equation, through 355 timesteps. Results shown for three different interpolants used to evaluate quantities at departure point.

order scheme, gives the worst results. The quadratic scheme performs marginally better than the quintic scheme. In order to verify that the poor performance of the quintic is not due to lack of resolution, the test is repeated using linear advection to displace the front. All parameters remain the same in this test as for the Burgers' equation test. The result is shown in Figure 7.10. The linear advection speed is chosen to be identical to the speed of the front in the analytical solution of the viscous Burgers' equation. For linear semi-Lagrangian advection, the accuracy of the solution now corresponds directly to the order of the interpolation. This confirms that the front is sufficiently well resolved on the grid for each interpolant to give its expected accuracy. This suggests that the error in nonlinear semi-Lagrangian advection is depends on properties of interpolation, other than formal accuracy. To examine this possibility further, additional tests should be made

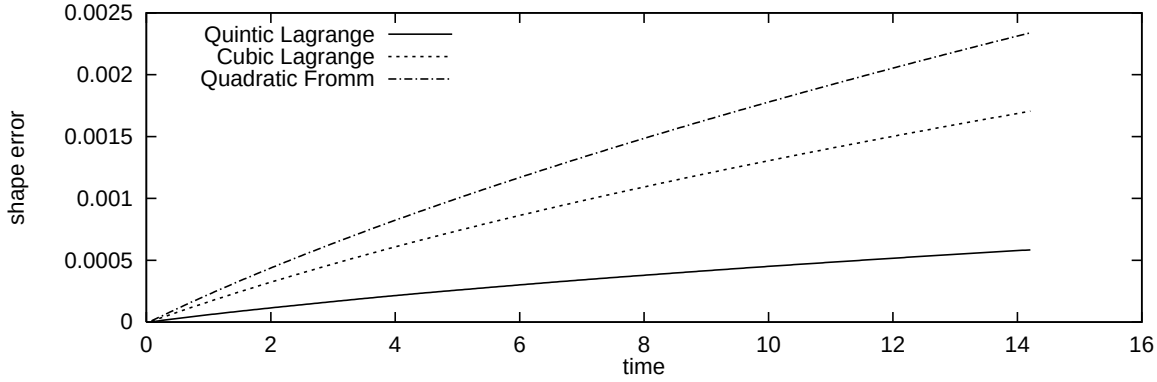

Figure 7.10: Shape error for linear advection of front at constant speed. The accuracy of the solution depends precisely on the order of accuracy of the interpolation used.

where the viscous Burgers' problem is solved for a range of different grid resolutions.

7.4.5 Test 3: Departure Point Method

The results of the interpolation test show that increasing the order of interpolation improves the error in the shape of the front. Increasing the accuracy of interpolation, however, has little effect on the error in the position of the front. In the first test we saw that the number of iterations used in finding departure points did improve the position error. We continue further in this direction by considering alternative methods of calculating departure points.

The first departure point method we define is the method used so far, which we give here for completeness.

Method 1: Time Extrapolation Method

The method is based on the implicit mid-point rule, with linear extrapolation in time of the wind field:

$$x_d = x - \frac{\Delta t}{2}\left[3u^n\left(\frac{x+x_d}{2}\right) - u^{n-1}\left(\frac{x+x_d}{2}\right)\right]. \tag{7.56}$$

This is solved iteratively for x_d. The wind fields are evaluated at the trajectory mid-points using linear interpolation.

The departure point formula (7.56) is based on linear extrapolation of the wind field to time $t_n + \Delta t/2$. We may also consider extrapolation of the wind along trajectories. This provides a second method for calculating departure points:

Method 2: Extrapolation along Trajectories

$$\begin{aligned} x_d &= x - \frac{\Delta t}{2}\left[3u^n(x_d) - u^{n-1}(x_{dd})\right] \\ x_{dd} &= x_d - \frac{\Delta t}{2}\left[u^n(x_d) + u^{n-1}(x_{dd})\right]. \end{aligned} \tag{7.57}$$

This formula calculates the position of a single trajectory, which passes through point x_{dd} at time t_{n-1}, through x_d at time t_n and arrives at the point x at time t_{n+1}. In this respect, the method is closely akin to a standard Adams method for solving the trajectory ODE. Again, this formula is solved iteratively for x_d and the wind is interpolated using linear interpolation. This scheme requires twice the number of interpolations per iteration, compared with the standard scheme. This is because the wind at times t_n and t_{n-1} are to be interpolated to different points. Consequently, for this scheme alone, we reduce the number of iterations to two. In this way each method will expend roughly equivalent amounts of computational effort in calculating departure points.

Both of the methods described above compute departure points solely from the existing wind fields, at times t_n and t_{n-1}. Another approach to calculating departure points is to use the wind data at time t_n only. The differential equation governing the wind field is used to provide an extrapolation of the wind from this time to time $t_n + \Delta t/2$. A second order accurate approximation to the departure points is then made using this extrapolated wind field. Now, since our purpose is to solve the differential equation governing u, we do not want to commit much effort to solving this same equation for the purpose of finding departure points. Our approach is to make a series expansion of u about time-level t_n and then use the inviscid form of Burgers' equation to eliminate time derivatives.

The power series expansion in t of the wind field at time $t_n + \Delta t/2$ is

$$u\left(x, t_n + \frac{\Delta t}{2}\right) = u + \frac{\Delta t}{2}\frac{\partial u}{\partial t} + \frac{\Delta t^2}{8}\frac{\partial^2 u}{\partial t^2} + \mathrm{O}(\Delta x^3), \tag{7.58}$$

where quantities on the right-hand side are evaluated at time t_n. Next we approximate the solution of the viscous Burgers' equation by neglecting the viscosity term,

$$u_t + uu_x = 0, \tag{7.59}$$

where we have used subscript notation to denote derivatives. Differentiating (7.59) with respect to x and with respect to t, we obtain

$$u_{tx} + (uu_x)_x = 0 \tag{7.60}$$

$$u_{tt} + u_t u_x + uu_{tx} = 0. \tag{7.61}$$

Combining these last three equations, we obtain expressions for the time derivatives solely in terms of space derivatives:

$$u_t = -uu_x \tag{7.62}$$

$$u_{tt} = \left(u^2 u_x\right)_x. \tag{7.63}$$

By substituting (7.62) and (7.63) into the power series expansion, (7.58), we obtain a third order accurate extrapolation formula. However, this formula must be discretised in space if it is to be of practical use. If a two-point difference is used to approximate u_x, then an $\mathrm{O}(\Delta x^2)$ error is introduced. As a result no greater accuracy will be obtained by including terms of higher than first order, in the series expansion (7.58). Using the discretisation,

$$(uu_x)_j = u_j\left(\frac{u_{j+1} - u_{j-1}}{2\Delta x}\right), \tag{7.64}$$

we arrive at the extrapolation formula:

Method 3: PDE Extrapolation

$$u_j^{n+\frac{1}{2}} = u_j^n\left[1 - \frac{\Delta t}{2\Delta x}\left(u_{j+1}^n - u_{j-1}^n\right)\right]. \tag{7.65}$$

Departure points are obtained by iterative solution of,

$$x_d = x - \Delta t u^{n+\frac{1}{2}} \left(\frac{x + x_d}{2} \right), \tag{7.66}$$

with $u^{n+\frac{1}{2}}(x)$ being evaluated by linear interpolation of the finite difference data.

Results

In order to compare the three departure point methods, we make a series of model integrations for varying values of the mesh ratio parameter. The timestep and number of timesteps for each model run are chosen so that the front is advected through the same distance in each case. This distance is chosen to be 8.5 with the current choice of model parameters. For each model run the maximum shape error is recorded. Using the numerical definition for the position of the front, x_F as defined by (7.53), we may calculate the average speed of the numerically advecting front. We define the ratio of this quantity to the analytical speed of the front to be the speed error. Figure 7.11 shows the maximum shape error and the speed error plotted against mesh ratio parameter, for the three departure point methods.

In terms of speed error, Method 1 and Method 3 are comparable at all values of the mesh ratio. Both of these methods show a more or less monotonic decline in the front speed, as the mesh ratio is increased. Method 2 shows an acceleration of the front speed, over the analytic speed, for mesh ratios close to one. Thereafter the numerical speed of the front given by this method decreases with increasing mesh ratio. However, the speed remains closer to the analytical speed than for either of the other methods. All the methods give a very accurate front speed for values of mesh parameter below about one half.

For most values of the mesh ratio, the lowest shape error is given by Method 2. For values of mesh ratio below two, all three methods show much the same variation in shape error. Method 1 gives a more rapidly increasing shape error for larger mesh ratios, compared to the other with methods.

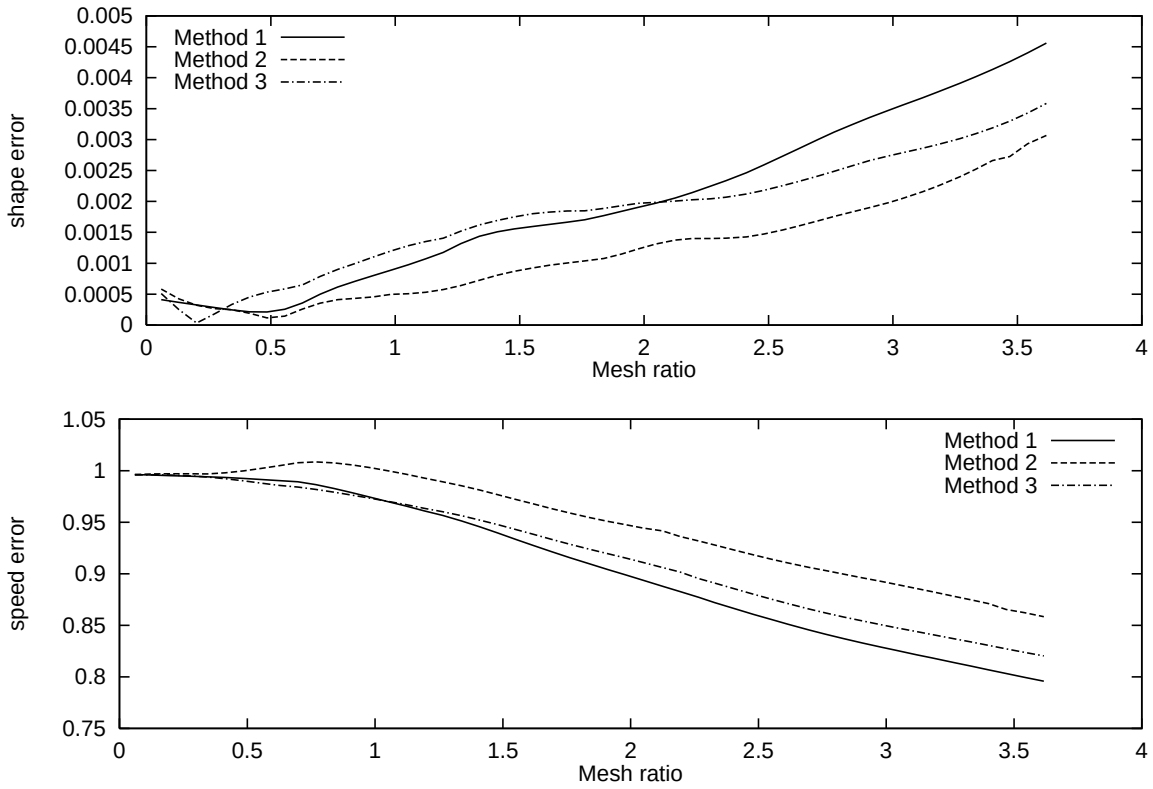

Figure 7.11: Error in speed of front and error in shape of front plotted against mesh ratio, for three different methods of calculating departure points.

7.4.6 Conclusion

We conclude from this test that there may be some advantage in using methods other than the standard method for calculating departure points. This conclusion only holds, however, in the limited setting of the tests conducted here. For calculating departure points in two and three dimensions, the standard method has much in its favour, [59].

7.5 Idealised Atmospheric Flow Results

In order to show that quadratic interpolation may be advantageous in a full atmospheric model, the centred quadratic, q^M, is introduced to a Met. Office NWP model. This model is derived from The Met. Office Unified Model version 5.0 (UM5.0), which is a nonhydrostatic semi-Lagrangian model. The model we consider here is an idealised two-

dimensional vertical slice version of UM5.0. It also lacks the physics parametrisations of the full model, and is a stand-alone code for two-dimensional, nonhydrostatic, inviscid dynamics.

As part of the validation work for UM5.0, idealised tests have been performed with the vertical slice model, in addition to tests with shallow water and vertical column models. One such test involves a neutrally stratified atmosphere, which flows over periodic, sinusoidal terrain. By neutrally stratified, we mean that the vertical profile of temperature is such that no buoyancy forces act, when air is displaced vertically. Under conditions of stable stratification in the atmosphere, buoyancy acts as a restoring force on vertically displaced air-parcels. This mechanism is associated with one of the categories of wave motion that occur in the atmosphere, so-called gravity waves. Unstable stratification leads to unopposed ascent of air-parcels, resulting in convective upwelling. The choice of neutral stratification for our idealised test, therefore, corresponds to a test of the evanescent response of the model. Tests using stable stratification have also been made, which we do not present here.

The domain for the evanescent test has periodic lateral boundaries and free-slip boundary conditions on the vertical boundaries. The top of the model is a rigid lid, set at height $z = H$. Initially the atmosphere is in hydrostatic balance; the vertical wind field is everywhere zero, and the horizontal wind field has the same constant value, U, everywhere. The lower boundary has the sinusoidal form

$$h(x) = h_0 \cos\left[\pi\left(\frac{2x - L}{L}\right)\right], \tag{7.67}$$

where L is the length of the domain (ie. the periodic length), h_0 is the amplitude of the orography and $h(x)$, the orographic height at position x. The parameter values used in the test are:

$$\begin{aligned} U &= 10\text{m/s} \\ h_0 &= 200\text{m} \\ H &= 6000\text{m} \end{aligned}$$

$$L \;=\; 4000\text{m}.$$

The model uses a semi-implicit time integration scheme. In the following, all the implicit weights are set to one half, which corresponds to a fully time-centred Crank-Nicolson scheme [31]. A terrain following grid is used, which relaxes linearly to the flat, horizontal upper boundary. Laterally, the grid has a constant mesh length. Twenty mesh intervals are used in the horizontal and forty in the vertical. A timestep is chosen which gives Courant numbers of approximately 0.15. All fields are interpolated using bi-cubic interpolation.

When the model is integrated forward in time, there is a short period during which the flow adjusts to the orography. After this, a quasi-steady state is reached. Figure 7.12 shows the horizontal and vertical wind components produced by the model after 2500 timesteps. These show the character of this flow problem: the wind slows down in the

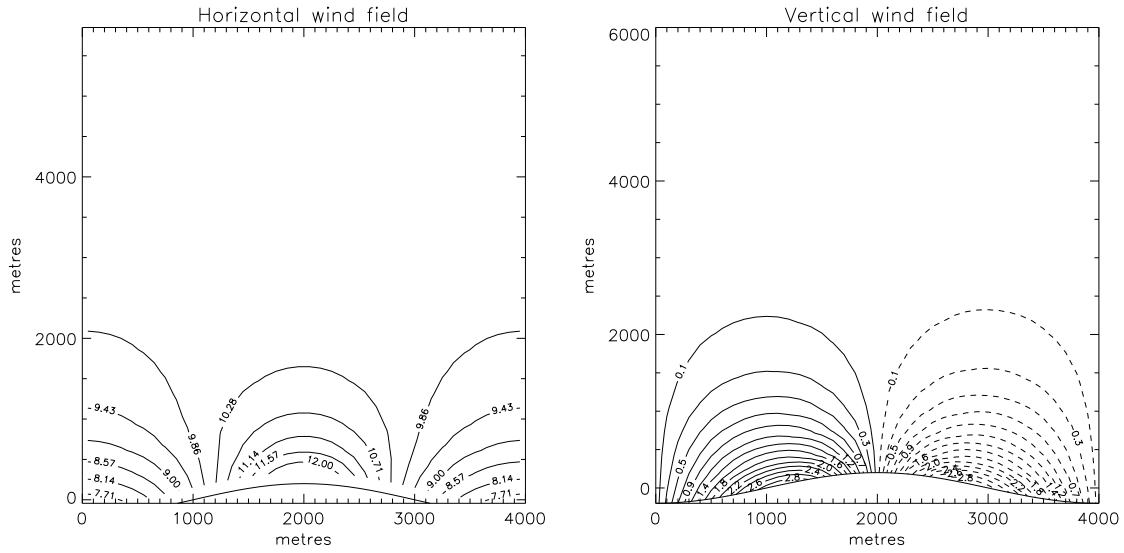

Figure 7.12: Horizontal and vertical wind fields at quasi-steady state. Model represents neutrally stratified and inviscid flow.

valleys and speeds up over the crest of the hill. Both fields are close to the analytical steady state at this point.

During the quasi-steady phase the flow gradually decelerates. The model doesn't

achieve the analytical steady state, due to the dissipation inherent in the semi-Lagrangian scheme. An indication of this slowing down is given by the total kinetic energy of the flow. When the order of interpolation of the model fields is varied, it is found that there is sensitivity only in the wind field interpolation. Figure 7.13 shows the total kinetic energy, normalised against that of the initial data, plotted against time. The time-span

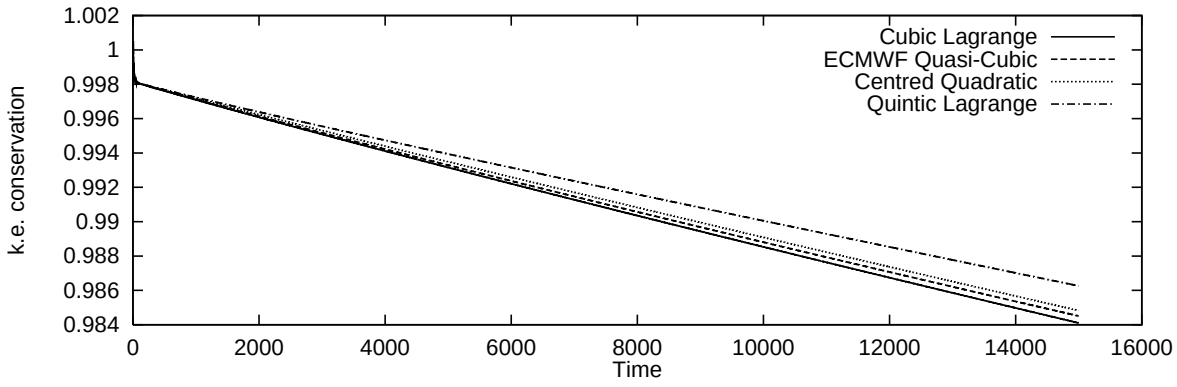

Figure 7.13: Normalised kinetic energy against time, showing deceleration of the quasi-steady state. Results shown for four interpolants of different formal accuracy applied to wind fields.

plotted represents ten thousand timesteps. The kinetic energy is shown for four different interpolants applied to the wind fields. All other fields are interpolated with cubics. The centred quadratic is the mean quadratic, q^M. This is made into a bi-quadratic interpolant through combining five univariate quadratics in a Cartesian product. Similarly, this is how the bi-cubic and bi-quintic interpolants are formed from univariate Lagrange interpolation. The ECMWF (European Centre for Medium Range Weather Forecasting) quasi-cubic, is a method where three univariate cubic and one linear interpolation are used to form a 'quasi-Cartesian' product. In effect it is a slightly degraded bi-cubic interpolant, which is quicker to calculate than the full bi-cubic. We see that quintic interpolation provides the best conservation of kinetic energy, among these four interpolants. The cubic interpolant gives the worst results. This is improved upon slightly by the ECMWF reduced cubic, which in turn is slightly inferior to the quadratic interpolant. A similar trend, for unexpectedly poor performance by cubic interpolation, has also been observed in idealised dynamical

tests with the global three-dimensional version of UM5.0.

7.6 Conclusion

The results presented in this chapter demonstrate a potential advantage which quadratic interpolation may have over cubic, in semi-Lagrangian advection. Truncation error analysis shows cubic interpolation to have a higher formal order of accuracy than has quadratic interpolation. Consequently linear advection using a cubic semi-Lagrangian method is more accurate than with a quadratic SL scheme. This is observed unambiguously in computational results. However, in the case of nonlinear advection the correlation between truncation error and computational accuracy is less clear. Even at moderately high levels of mesh refinement, a low order interpolant may give better results than a high order one, when applied to nonlinear advection of the wind fields. This is likely to hold at the levels of grid resolution currently used in NWP models.

One explanation for this discrepancy, between analysis and computational results, is suggested by the viscous Burgers' equation test. In this test the nonlinear advection acts to increase the steepness of the front. This tendency is balanced by the diffusion term, which acts to smear out the front. In a numerical solution of this problem, the truncation error of the scheme will contribute to the smoothing effect of the diffusion. For cubic interpolation the leading order truncation error depends on the fourth spatial derivative, while for a quadratic it is the third derivative. The balance obtained in a numerical solution, between the steepening and dissipative effects, therefore differs qualitatively as well as quantitatively between cubic and quadratic schemes. This difference in the order of derivatives may play some part in explaining the observed results. Another consideration is the degree to which non-conservative SL schemes correctly model a conservation law. If a quadratic interpolant more accurately meets the conservation requirement, than does a cubic, this could explain the observed results.

We conclude that further analysis is required, in order that nonlinear advection with

semi-Lagrangian schemes be fully understood.

Chapter 8

Global Forecasting

Introduction

In the preceding chapters we have examined the application of SL methods to fluid flows in more than one dimension. Our aim is to identify SL methods particularly suitable for use in weather simulation models. Unless the model covers a very limited geographical region, it must take account of the Earth's curvature. As a first approximation we take the Earth to be a perfect sphere.

In this chapter we shall only consider modelling of atmospheric flows over the whole sphere. The special case of limited area modelling does not raise any further issues which do not arise in a global model.

The advantage of computational speed available when using a semi-Lagrangian scheme is maximised when using a regular computational grid. A regular grid allows for both efficient interpolation and fast location of departure points. For this reason the initial development of SL methods has been to problems in domains covered by a Cartesian coordinate system, where the coordinate directions generate a regular, rectangular grid. Such methods are immediately applicable to limited area atmospheric modelling.

For extending the method for use in global simulations it is necessary to consider the problems posed by spherical geometry. These problems arise from the fact that the sphere

can not be covered by a single coordinate system without singularities. In particular, if we use equal angular spacings along lines of longitude and latitude to generate the grid, then the grid will share the singularities of the coordinate system at the poles. This singularity is reflected in the grid indexing by a degeneracy at the poles: since a line of latitude has zero length at a pole, all the grid- points along that line are coincident with the pole.

Yet for all these difficulties there are significant benefits to be obtained by the use of a latitude-longitude grid. Most importantly, finite difference calculations on such a grid have the simplicity commonly associated with a regular Cartesian grid on a plane rectangular domain. For if (λ, θ) are coordinates of longitude and latitude of points on the sphere, then in the λ-θ plane the grid is regular and rectangular. The singularity at the poles takes the form of an indexing degeneracy: along the lines $\theta = \pi/2, -\pi/2$ all points correspond to the North and South poles respectively. Unfortunately the components of velocity in this coordinate system present particular difficulties for the SL method: one of the components has a polar singularity, which must be taken into account in calculations of trajectories and momentum components near the poles.

Bates et al. [2] present perhaps the best current set of remedies for the problems of spherical geometry in SL schemes. The equations of momentum, which in component form involve polar singularities, are discretised in vector form. This avoids metric terms explicitly occurring in the discrete equations. The picture, however, is less clear for the trajectory calculation. Away from the poles the coordinate lines have low curvature, and we can apply methods of trajectory calculation designed for Euclidean geometry. Closer to the poles, a separate local coordinate system is set up for each grid point. The new coordinates are obtained by rotating the polar axis of the spherical coordinates, so that the near-polar grid point now lies on the coordinate equator. In these coordinates the trajectory equation can once again be solved using methods for flat geometry. Finally at the poles themselves a Fourier method is used. Before looking at these methods in more detail, we shall first review the spherical coordinate system used in meteorology.

8.1 Geophysical Spherical Coordinates

Let λ and θ be the longitude and latitude respectively of a point P, which is a distance r from the Earth's centre. The geophysical spherical coordinates of this point are then (λ, θ, r). It is conventional to denote the unit vectors at P in the coordinate directions λ, θ, r by $\boldsymbol{i}(P)$, $\boldsymbol{j}(P)$, $\boldsymbol{k}(P)$ respectively. For brevity, we shall refer to the unit vectors at any point P as simply $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$, but it is important to remember that these vectors depend on the coordinates of P.

Now let $\boldsymbol{I}$, $\boldsymbol{J}$, $\boldsymbol{K}$ be the Cartesian unit vectors, with $\boldsymbol{J}$ aligned along the polar axis directed from South to North. The vector $\boldsymbol{K}$ is directed from the Earth's centre to the point of zero longitude on the equator (fig. 8.1). The two sets of basis vectors are related by

$$\begin{aligned} \boldsymbol{i} &= \mathcal{M}_{11}\boldsymbol{I} + \mathcal{M}_{12}\boldsymbol{J} + \mathcal{M}_{13}\boldsymbol{K} \\ \boldsymbol{j} &= \mathcal{M}_{21}\boldsymbol{I} + \mathcal{M}_{22}\boldsymbol{J} + \mathcal{M}_{23}\boldsymbol{K} \\ \boldsymbol{k} &= \mathcal{M}_{31}\boldsymbol{I} + \mathcal{M}_{32}\boldsymbol{J} + \mathcal{M}_{33}\boldsymbol{K} \end{aligned} \tag{8.1}$$

where the matrix of coefficients $\mathcal{M} = (\mathcal{M}_{\alpha\beta})$ is

$$\mathcal{M}(\lambda, \theta) = \begin{bmatrix} \cos\lambda & 0 & -\sin\lambda \\ -\sin\theta\sin\lambda & \cos\theta & -\sin\theta\cos\lambda \\ \cos\theta\sin\lambda & \sin\theta & \cos\theta\cos\lambda \end{bmatrix}. \tag{8.2}$$

Use will be made of this result in deriving a discretisation of the momentum equation in vector form. This result also allows us to compute the partial derivatives of the unit vectors $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$ with respect to λ, θ and r. From these partial derivatives vector differential quantities may be computed in coordinate form. Again we shall be making use of this when discussing the semi-Lagrangian treatment of momentum, where the relevant result

is

$$
\begin{aligned}
(\boldsymbol{v}.\boldsymbol{\nabla})\boldsymbol{v} = \quad & \boldsymbol{i}\left[\frac{u}{r\cos\theta}\frac{\partial u}{\partial\lambda} + \frac{v}{r}\frac{\partial u}{\partial\theta} + w\frac{\partial u}{\partial r} + \frac{u}{r}(w - v\tan\theta)\right] \\
+ \quad & \boldsymbol{j}\left[\frac{u}{r\cos\theta}\frac{\partial v}{\partial\lambda} + \frac{v}{r}\frac{\partial v}{\partial\theta} + w\frac{\partial v}{\partial r} + \frac{1}{r}(vw + u^2\tan\theta)\right] \\
+ \quad & \boldsymbol{k}\left[\frac{u}{r\cos\theta}\frac{\partial w}{\partial\lambda} + \frac{v}{r}\frac{\partial w}{\partial\theta} + w\frac{\partial w}{\partial r} - \frac{(u^2+v^2)}{r}\right].
\end{aligned}
\tag{8.3}
$$

8.2 Shallow Water Equations on the Sphere

Many of the important aspects of global atmospheric flow are represented by the shallow water equations [60]. These result from a vertical integration of the primitive flow equations. In vector form, the shallow water equations on the sphere are:

$$
\left(\frac{D\boldsymbol{v}}{Dt}\right)_{\mathrm{H}} = -f\boldsymbol{r}\wedge\boldsymbol{v} - \boldsymbol{\nabla}\phi \tag{8.4}
$$

$$
\left(\frac{D\phi}{Dt}\right)_{\mathrm{H}} = -\phi\boldsymbol{\nabla}.\boldsymbol{v} \tag{8.5}
$$

where ϕ is the geopotential height (ie. the integral of $g\times$density through the depth of the atmosphere); f is the Coriolis parameter; r is the radius of the Earth; $\boldsymbol{v}$ is the horizontal velocity ($\boldsymbol{v} = u\boldsymbol{i} + v\boldsymbol{j}$) and $\left(\frac{D}{Dt}\right)_{\mathrm{H}} \equiv \frac{\partial}{\partial t} + \frac{u}{r\cos\theta}\frac{\partial}{\partial\lambda} + \frac{v}{r}\frac{\partial}{\partial\theta}$. In component form, using (8.3)

$$
\begin{aligned}
\left(\frac{D\boldsymbol{v}}{Dt}\right)_{\mathrm{H}} = \quad & \boldsymbol{i}\left[\frac{u}{r\cos\theta}\frac{\partial u}{\partial\lambda} + \frac{v}{r}\frac{\partial u}{\partial\theta} - \frac{uv}{r}\tan\theta\right] \\
+ \quad & \boldsymbol{j}\left[\frac{u}{r\cos\theta}\frac{\partial v}{\partial\lambda} + \frac{v}{r}\frac{\partial v}{\partial\theta} + \frac{u^2}{r}\tan\theta\right].
\end{aligned}
\tag{8.6}
$$

A discretisation of this, in the λ-θ plane, will have advective velocity components $\left(\frac{u}{r\cos\theta}, \frac{v}{r}\right)$. The first component contains singularities at the poles, where $\theta = \pm\pi/2$. In consequence any numerical scheme, based on this form of discretisation, will require special consideration to be given to the polar regions. This lack of uniformity, in the numerical modelling in different regions of the sphere, is simply due to the approach of generating a scheme through a single coordinate system. A remedy would be to use a different coordinate system in the regions close to the poles: one which didn't contain singularities there.

Such an approach is described by McDonald and Bates [29]. This approach is not entirely satisfactory, and in the case of the momentum equation the difficulty can be avoided by making a vector discretisation.

8.3 Vector Discretisation of Momentum Equation

Consider the nonlinear equation for horizontal advection,

$$\left[\frac{D\boldsymbol{v}}{Dt}\right]_{\mathrm{H}} = 0. \tag{8.7}$$

We have seen that when this equation is written in component form (8.6), singularities are encountered at the poles. A further complication is the fact that we must also take account of the metric terms. These occur due to the dependency of the basis vectors on the coordinates in spherical geometry. An alternative approach is to discretise the equation (8.7) in vector form, giving

$$\frac{1}{\Delta t}(\boldsymbol{v}_i^{n+1} - \boldsymbol{v}_d^n) = 0. \tag{8.8}$$

If the components of a vector, $\boldsymbol{w}$, in the spherical coordinate system are $((\boldsymbol{w})_1, (\boldsymbol{w})_2, (\boldsymbol{w})_3)$ then we can express (8.8) in terms of the basis vectors at the arrival and departure points:

$$\frac{1}{\Delta t}\left[\sum_{\alpha=1}^{3}(\boldsymbol{v}_i^{n+1})_\alpha \boldsymbol{e}_\alpha(\lambda_i,\theta_i) - \sum_{\alpha=1}^{3}(\boldsymbol{v}_d^{n})_\alpha \boldsymbol{e}_\alpha(\lambda_d,\theta_d)\right] = 0 \tag{8.9}$$

where $\boldsymbol{e}_1 = \boldsymbol{i}$, $\boldsymbol{e}_2 = \boldsymbol{j}$, $\boldsymbol{e}_3 = \boldsymbol{k}$ and (λ_i, θ_i), (λ_d, θ_d) are the arrival and departure points of a trajectory on the sphere.

To obtain a discretisation which avoids the need to represent metric terms explicitly, we replace the two sets of basis vectors in (8.9) by a single set. For this purpose, the spherical basis vectors at the mid-point of the trajectory are used. A simple method for achieving this uses the Cartesian basis vectors ($\boldsymbol{I}$, $\boldsymbol{J}$, $\boldsymbol{K}$) in an intermediate step. Using the matrix of transformation coefficients, (8.2), we can relate the two sets of basis functions, at the arrival and departure points, to the set of Cartesian basis functions. The inverse transformation (which exists, since $\det \mathcal{M} = 1$) allows the Cartesian basis to

be expressed in terms of the basis vectors at the trajectory mid-point $\{\boldsymbol{e}_\alpha(\lambda_m, \theta_m) : \alpha = 1, 2, 3\}$.

Consider the basis vectors for the spherical coordinates at the arrival point. These can be expressed relative to the Cartesian basis as follows:

$$\boldsymbol{e}_\alpha(\lambda_i, \theta_i) = \sum_{\beta=1}^{3} \mathcal{M}_{\alpha\beta}(\lambda_i, \theta_i)\boldsymbol{E}_\beta \qquad \alpha = 1, 2, 3 \tag{8.10}$$

where $\mathcal{M}_{\alpha\beta}$ are the components of (8.2) and $\boldsymbol{E}_1 = \boldsymbol{I}$, $\boldsymbol{E}_2 = \boldsymbol{J}$, $\boldsymbol{E}_3 = \boldsymbol{K}$. The $\boldsymbol{E}_\beta$ basis vectors are replaced using the inverse transformation at the trajectory mid-point:

$$\boldsymbol{E}_\beta = \sum_{\gamma=1}^{3} [\mathcal{M}(\lambda_m, \theta_m)^{-1}]_{\beta\gamma}\boldsymbol{e}_\gamma(\lambda_m, \theta_m) \qquad \beta = 1, 2, 3. \tag{8.11}$$

From (8.10) and (8.11) we obtain

$$\boldsymbol{e}_\alpha(\lambda_i, \theta_i) = \sum_{\gamma=1}^{3} \widetilde{\mathcal{M}}(\lambda_i, \theta_i)\boldsymbol{e}_\gamma(\lambda_m, \theta_m) \qquad \alpha = 1, 2, 3$$

where the matrix of coefficients for this transformation is:

$$\widetilde{\mathcal{M}}(\lambda_i, \theta_i) = \mathcal{M}(\lambda_i, \theta_i)\mathcal{M}(\lambda_m, \theta_m)^{-1}. \tag{8.12}$$

A similar result holds for the basis vectors at the departure point. When these results are combined we obtain the following trajectory-centred discretisation:

$$\frac{D\boldsymbol{v}}{Dt} \to \frac{1}{\Delta t}\left\{\sum_{\alpha=1}^{3}[(\boldsymbol{v}_i^{n+1})_\alpha \widetilde{\mathcal{M}}_{\alpha\gamma}(\lambda_i, \theta_i) - (\boldsymbol{v}_d^n)_\alpha \widetilde{\mathcal{M}}_{\alpha\gamma}(\lambda_d, \theta_d)\boldsymbol{e}_\gamma(\lambda_m, \theta_m)\right\}. \tag{8.13}$$

Source terms are treated following the same procedure, and are trajectory centred by taking averages of values at the arrival and departure points.

This method provides a discretisation of the momentum equations which avoids the problems associated with a discretisation of the equations in spherical coordinates. The method is quite independent of whatever grid might be chosen to cover the sphere. In Bates et al. the Arakawa C-grid is employed, but other grids are equally applicable.

8.4 Calculating Departure Points

In order to calculate the departure points used by the semi-Lagrangian method we must solve the trajectory equation. In geospherical coordinates the trajectory equation is

$$\frac{d}{dt}\begin{bmatrix}\lambda\\ \\ \theta\end{bmatrix} = \frac{1}{a\cos\theta}\begin{bmatrix}u\\ \\ v\cos\theta\end{bmatrix}. \tag{8.14}$$

For regions away from the poles it is possible to use the implicit mid-point rule to solve this, just as for the corresponding equation in plane geometry. Hence we make the approximation that the right hand side of (8.14) remains constant throughout a time step, and equal to its value at the trajectory mid-point. Integration of (8.14) then results in the equations

$$\begin{aligned}\lambda_d &= \lambda_i - \left(\frac{u_m^{n+1/2}}{a\cos\theta_m}\right)\Delta t\\ \theta_d &= \theta_i - \left(\frac{v_m^{n+1/2}}{a}\right)\Delta t,\end{aligned} \tag{8.15}$$

where subscript m denotes the mid-point of the great arc from (λ_d, θ_d) to (λ_i, θ_i), and the velocity components are interpolated to this point and extrapolated to the mid-time level. These equations are then solved iteratively for (λ_d, θ_d).

Sufficiently close to the poles, the assumption that the right hand side of (8.14) remains constant through a time step is no longer valid to any degree of accuracy. To deal with this problem, McDonald and Bates [29] introduce a new coordinate system for each grid point within some preset neighbourhood of a pole. Assume that node i, with coordinates (λ_i, θ_i), is such a node. Then the new coordinates, (λ', θ'), are such that node i is at $(\lambda' = 0, \theta' = 0)$ and the $(\boldsymbol{i}, \boldsymbol{j})$ unit vectors of the two coordinate systems are coincident at node i. Consideration of the geometry of this construction reveals that the

two coordinate systems are related through

$$\begin{aligned} \lambda &= \lambda_i + \tan^{-1}\left[\frac{\cos\theta' \sin\lambda'}{\cos\theta'\cos\lambda'\cos\theta_i - \sin\theta'\sin\theta_i}\right] \\ \theta &= \sin^{-1}[\cos\theta'\cos\lambda'\sin\theta_i + \sin\theta'\cos\theta_i] \end{aligned} \tag{8.16}$$

and that the velocity components transform according to

$$\begin{bmatrix} u' \\ \\ v' \end{bmatrix} = \begin{bmatrix} a & -b \\ \\ b & a \end{bmatrix} \begin{bmatrix} u \\ \\ v \end{bmatrix} \tag{8.17}$$

where

$$\begin{aligned} a &= \frac{1}{\cos\theta'}[\cos\theta\cos\theta_i + \sin\theta\sin\theta_i\cos(\lambda - \lambda_i)] \\ b &= \frac{1}{\cos\theta'}\sin\theta_i\sin(\lambda - \lambda_i). \end{aligned}$$

In the (λ', θ') coordinate system, the coordinate lines near node i have low curvature and so it is valid to use the implicit mid-point rule in these coordinates. Once the departure point has been found in the primed coordinates, the (λ, θ) coordinates are obtained from (8.16).

At the pole itself there is a singularity in the (λ, θ) coordinate system, making the above approach invalid. The method employed by McDonald & Bates to treat the poles depends crucially on the distribution of the variables on the chosen grid. They use the Arakawa C-grid, for which the pole is surrounded by a latitude circle of grid points holding values of v. By constructing a Fourier representation of the velocity field from this data, an approximation for the polar departure point can be found. Full details of all these methods are contained in [29] and [2].

We conclude this chapter by returning to the task of determining departure points in regions away from the poles.

8.5 A Geodesic Method for Calculating Departure Point Coordinates.

In a three-dimensional global NWP model, the time rates of change of longitude, latitude and height are, respectively,

$$\frac{d\lambda}{dt} = \frac{u}{r\cos\phi} \tag{8.18}$$

$$\frac{d\phi}{dt} = \frac{v}{r} \tag{8.19}$$

$$\frac{dr}{dt} = w, \tag{8.20}$$

where u, v and w are the wind components with respect to local unit basis vectors, $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$, (see [13] and Chapter 8). When the semi-Lagrangian method is used to discretise the governing equations, the departure points will be calculated from a discretisation of (8.18)-(8.20). This approach is only valid in regions sufficiently far from the coordinate poles. In polar regions it is necessary to apply a coordinate transformation, in order to avoid the removable singularity in the coordinates at the geographical pole. In the following discussion we consider only regions away from the poles.

It is generally found necessary, in calculating departure points, to treat the horizontal coordinates, λ and ϕ, separately from the vertical coordinate, r. This involves making the "shallow atmosphere" approximation, replacing (8.18) and (8.19) with

$$\frac{d\lambda}{dt} = \frac{u}{a\cos\phi} \tag{8.21}$$

$$\frac{d\phi}{dt} = \frac{v}{a}, \tag{8.22}$$

where a is the radius of the Earth. Equation (8.20) is still used to determine the r coordinate of the departure point. If the grid point (λ, ϕ, r) at time $t_n + \Delta t$ has its time t_n departure point at (λ_d, ϕ_d, r_d), then the implicit mid-point rule, applied to (8.20), provides

$$\frac{r - r_d}{\Delta t} = w\left(\lambda_m, \phi_m, r_m, t_n + \frac{\Delta t}{2}\right), \tag{8.23}$$

where subscript m denotes the mid-point of the trajectory. As always, applying this formula to a finite difference model requires extrapolation of the wind fields forward in time, to $t_n + \Delta t/2$; and wind fields are evaluated at trajectory mid-points using linear interpolation of the data. This equation is solved iteratively for r_d. Similar formulae for the horizontal components must also be iterated simultaneously, together with (8.23), since all the formulae are necessarily coupled through the position of the trajectory mid-point. We next turn to a consideration of the horizontal displacement equations, (8.21) and (8.22).

In solving these two equations, to determine the horizontal coordinates of a departure point, it is necessary to ensure that the trajectory remains on the surface of the sphere. That is, any vertical displacement of a fluid parcel must be due only to the vertical wind field. (Reasons for this requirement are connected with issues of stratification and stability of the atmosphere.) Hence, we must apply an additional constraint in solving (8.21) and (8.22). One such constraint is to assume that fluid parcel trajectories lie along great circles. One way of viewing this procedure is to view the departure point calculation as taking place in a curved, two-dimensional space, the surface of a sphere.

The following assumptions are generally made in SL schemes, when determining the trajectory of a fluid parcel (for purposes of calculating departure points):

- The trajectory is a straight line;
- The speed along the trajectory is constant during the timestep.

In a curved space the natural generalisation of the first of these assumptions is, that the trajectory should be a geodesic. The second assumption then becomes, that the speed along the geodesic should be constant during the timestep. On the sphere, geodesics are great circles. Hence we see that the shallow atmosphere approximation, coupled with the great circle constraint, lead us to consider how departure points might be calculated in an arbitrary curved space, or "Riemannian manifold", [49], [48]. This we do by way of an example, describing the procedure for the case of a sphere.

Before doing so, we consider what further requirements might be needed of a departure point scheme for use with the SL method. Accuracy is one issue. We shall require the method to have a second order truncation error in time. Efficiency is also a consideration, particularly in NWP applications. A departure point scheme requiring many expensive evaluations of trig functions, at each timestep, would not be of much use. Our list of desirable properties for the departure point scheme is:

- Trajectories should closely follow geodesics, with constant speed during each timestep;
- Departure points should be evaluated to at least second order accuracy in time;
- Any quantities dependent only on the geometry of the space, and which remain constant in time, should be evaluated at grid points.

The last of these items addresses the issue of efficiency. Spherical geometry, for instance, inevitably involves the use of trigonometric functions. These are computationally expensive to evaluate. If trig functions are required to be evaluated only at grid points, then they need be calculated just once and stored for use at each timestep. A departure point scheme on the sphere, meeting all these criteria, has been described by Ritchie and Beaudoin [45]. Their approach is to use three-dimensional coordinate geometry to determine great circles. Here we shall re-derive their scheme using the formalism of geodesics.

8.5.1 Geodesic Equations for the Sphere

The equations for a geodesic may be obtained by considering the affect of variations on arc length, over all paths.

Variational Principle

Let the following be defined:

$$a = \text{radius of sphere}$$

$$\lambda = \text{longitude}$$

$$\begin{aligned} \phi &= \text{Latitude} \\ s &= \text{arc length along surface of sphere.} \end{aligned} \tag{8.24}$$

For the longitude-latitude coordinate system, the line element is

$$ds^2 = a^2 \cos^2\phi \, d\lambda^2 + a^2 \, d\phi^2. \tag{8.25}$$

Let $\lambda' = \dfrac{d\lambda}{ds}$ and $\phi' = \dfrac{d\phi}{ds}$, and define

$$L(\lambda(s), \phi(s), \lambda'(s), \phi'(s)) = \frac{1}{2} a^2 \cos^2\phi \, (\lambda')^2 + \frac{1}{2} a^2 (\phi')^2. \tag{8.26}$$

Then geodesics satisfy the variational principle

$$\delta L = 0, \tag{8.27}$$

where variations are made in the path $(\lambda(s), \phi(s))$.

Euler-Lagrange Equations

The Euler-Lagrange equations corresponding to (8.27) are:

(i)

$$\begin{aligned} \frac{d}{ds}\frac{\partial L}{\partial \lambda'} - \frac{\partial L}{\partial \lambda} = 0 \quad &\Rightarrow \quad \frac{d}{ds}(a^2 \cos^2\phi \, \lambda') = 0 \\ &\Rightarrow \quad a^2 \cos^2\phi \, \lambda'' - 2a^2 \cos\phi \sin\phi \, \lambda'\phi' = 0 \\ &\Rightarrow \quad \lambda'' - 2\tan\phi \, \lambda'\phi' = 0. \end{aligned} \tag{8.28}$$

(ii)

$$\begin{aligned} \frac{d}{ds}\frac{\partial L}{\partial \phi'} - \frac{\partial L}{\partial \phi} = 0 \quad &\Rightarrow \quad \frac{d}{ds}(a^2\phi') + a^2 \cos\phi \sin\phi \, (\lambda')^2 = 0 \\ &\Rightarrow \quad \phi'' + \cos\phi \sin\phi \, (\lambda')^2 = 0. \end{aligned} \tag{8.29}$$

Both (8.28) and (8.29) are valid only in regions away from the poles. The polar regions are not considered here, as they play no part in the two-dimensional vertical slice model.

Time Derivatives

Let $\dot{\lambda} = \dfrac{d\lambda}{dt}$ and $\dot{\phi} = \dfrac{d\phi}{dt}$, then

$$\frac{d\lambda}{dt} = \frac{ds}{dt}\frac{d\lambda}{ds} \tag{8.30}$$

and

$$\frac{d^2\lambda}{dt^2} = \frac{d^2s}{dt^2}\frac{d\lambda}{ds} + \left(\frac{ds}{dt}\right)^2 \frac{d^2\lambda}{ds^2}. \tag{8.31}$$

From (8.30) and (8.31) we obtain

$$\lambda' = \frac{\dot{\lambda}}{\dot{s}} \tag{8.32}$$

$$\lambda'' = \frac{\ddot{\lambda}}{\dot{s}^2} - \frac{\ddot{s}}{\dot{s}^3}\dot{\lambda} \tag{8.33}$$

Similarly,

$$\phi' = \frac{\dot{\phi}}{\dot{s}} \tag{8.34}$$

$$\phi'' = \frac{\ddot{\phi}}{\dot{s}^2} - \frac{\ddot{s}}{\dot{s}^3}\dot{\phi} \tag{8.35}$$

Using (8.32) and (8.33) in (8.28), and (8.34) and (8.35) in (8.29), we obtain

$$\ddot{\lambda} - 2\tan\phi\,\dot{\lambda}\dot{\phi} = \frac{\ddot{s}}{\dot{s}}\dot{\lambda} \tag{8.36}$$

$$\ddot{\phi} + \cos\phi\sin\phi\,(\dot{\lambda})^2 = \frac{\ddot{s}}{\dot{s}}\dot{\phi}. \tag{8.37}$$

8.5.2 Departure Point Scheme — Outline

In this section we derive a geodesic scheme for calculating departure points, which is based on using the wind fields

$$u = a\cos\phi\,\dot{\lambda} \tag{8.38}$$

$$v = a\dot{\phi}. \tag{8.39}$$

Trajectories are calculated using time-centred approximations, to provide second order accuracy. Given the position of a particle at time $t + \Delta t$, we wish to find its position at the earlier time t. This must be achieved numerically using only values of

wind components at the intermediate time $t + \Delta t/2$. In describing the scheme it is useful to employ the following notation:

$$\lambda^+ = \lambda(t + \Delta t), \quad \lambda^0 = \lambda(t + \Delta t/2), \quad \lambda^- = \lambda(t) \tag{8.40}$$

and similarly for other variables, so that $u^+ = u(t + \Delta t, \lambda(t + \Delta t), \phi(t + \Delta t))$ etc.

Following the above considerations, we make Taylor series expansions about time $t + \Delta t/2$:

Longitude

$$\lambda^+ = \lambda^0 + \frac{\Delta t}{2}\dot{\lambda}^0 + \frac{\Delta t^2}{8}\ddot{\lambda}^0 + \frac{\Delta t^3}{48}\dddot{\lambda}^0 + \mathrm{O}(\Delta t^4) \tag{8.41}$$

$$\lambda^- = \lambda^0 - \frac{\Delta t}{2}\dot{\lambda}^0 + \frac{\Delta t^2}{8}\ddot{\lambda}^0 - \frac{\Delta t^3}{48}\dddot{\lambda}^0 + \mathrm{O}(\Delta t^4). \tag{8.42}$$

Combining (8.41) and (8.42), we replace (8.42) with

$$\lambda^- = \lambda^+ - \Delta t\dot{\lambda}^0 - \frac{\Delta t^3}{24}\dddot{\lambda}^0 + \mathrm{O}(\Delta t^5). \tag{8.43}$$

Latitude Analogously we have

$$\phi^+ = \phi^0 + \frac{\Delta t}{2}\dot{\phi}^0 + \frac{\Delta t^2}{8}\ddot{\phi}^0 + \frac{\Delta t^3}{48}\dddot{\phi}^0 + \mathrm{O}(\Delta t^4) \tag{8.44}$$

$$\phi^- = \phi^+ - \Delta t\dot{\phi}^0 - \frac{\Delta t^3}{24}\dddot{\phi}^0 + \mathrm{O}(\Delta t^5). \tag{8.45}$$

Equations (8.41) and (8.44) define λ^0 and ϕ^0 implicitly, in terms of the time derivatives of the coordinates at that point. The first stage of a numerical departure point scheme is therefore to calculate the coordinates of the trajectory mid-point by iterative solution of these equations. Once the mid-point (λ^0, ϕ^0) is known, the departure point (λ^-, ϕ^-) is given explicitly by (8.43) and (8.45). So the scheme is as follows:

Scheme: First Stage

Find (λ^0, ϕ^0) by iteration on

$$\lambda^0 = \lambda^+ - \frac{\Delta t}{2}\dot{\lambda}^0 - \frac{\Delta t^2}{8}\ddot{\lambda}^0 - \frac{\Delta t^3}{48}\dddot{\lambda}^0 + \mathrm{O}(\Delta t^4) \tag{8.46}$$

$$\phi^0 = \phi^+ - \frac{\Delta t}{2}\dot{\phi}^0 - \frac{\Delta t^2}{8}\ddot{\phi}^0 - \frac{\Delta t^3}{48}\dddot{\phi}^0 + \mathrm{O}(\Delta t^4). \tag{8.47}$$

Scheme: Second Stage

Once the mid-trajectory coordinates have been found, the time derivatives of λ and ϕ are evaluated at that point. The departure point coordinates are then given by (8.43) and (8.45) to be

$$\begin{aligned} \lambda^- &= \lambda^+ - \Delta t\dot{\lambda}^0 - \frac{\Delta t^3}{24}\dddot{\lambda}^0 + \mathrm{O}(\Delta t^5) \\ \phi^- &= \phi^+ - \Delta t\dot{\phi}^0 - \frac{\Delta t^3}{24}\dddot{\phi}^0 + \mathrm{O}(\Delta t^5). \end{aligned}$$

8.5.3 Departure Point Scheme — Practical Details

So far we have developed the scheme in terms of time derivatives of the coordinates λ and ϕ. For the scheme to be practical it should use only the wind components u and w evaluated at time $t + \Delta t/2$. The wind field at this mid-time level is obtained by extrapolation of the data already calculated at time levels t and $t - \Delta t$. In order to obtain expressions for all the time derivatives required in the above outlined scheme, extra assumptions about the trajectory are required. These are outlined in the next section.

8.5.4 Great Circle Constant Speed Approximation

Departure point schemes in Cartesian geometry typically make the approximation that each trajectory follows a straight line, and that the speed along a trajectory is constant during each time step. Such a scheme is second order accurate in time if the linear trajectory is constructed from velocity data valid mid-way through the timestep. A natural generalisation of this scheme to other geometries is to assume that a trajectory follows a geodesic during each time step. This is equivalent to assuming a straight-line trajectory in Cartesian coordinates. The constant speed approximation generalises to the assumption that the time rate of change of arc-length along a trajectory is constant. In spherical geometry geodesics are great circles, so a suitable scheme may be derived with a little three-dimensional trigonometry [45]. Here we use the equations for a geodesic on a sphere (8.28) and (8.29).

Applying $\dot{s} = \text{constant}$ and $\ddot{s} = 0$ to (8.28) and (8.29), we obtain

$$\ddot{\lambda} = 2\tan\phi\,\dot{\lambda}\dot{\phi} \tag{8.48}$$

$$\ddot{\phi} = -\cos\phi\sin\phi\,(\dot{\lambda})^2. \tag{8.49}$$

These are the constant speed geodesic equations, from which we observe that the trajectory construction of the current method does not assume $\dot{\lambda}$ and $\dot{\phi}$ are constant during a time step. We also require the third time derivatives of λ and ϕ, obtained by differentiating the last two formulae:

$$\dddot{\lambda} = 2(1+3\tan^2\phi)\dot{\lambda}\dot{\phi}^2 - 2\sin^2\phi\,\dot{\lambda}^3 \tag{8.50}$$

$$\dddot{\phi} = -(1+2\sin^2\phi)\dot{\lambda}^2\dot{\phi}. \tag{8.51}$$

All the higher time derivatives of the coordinates may be expressed in terms of the first derivatives. At time $t + \Delta t/2$ the first derivatives are related to the wind components by

$$\dot{\lambda}^0 = \frac{u^0}{a\cos\phi^0} \tag{8.52}$$

$$\dot{\phi}^0 = \frac{v^0}{a}. \tag{8.53}$$

Using (8.48)-(8.53) the first stage of the scheme now takes the form,

$$\begin{aligned}\lambda^0 = {} & \lambda^+ - \frac{\Delta t}{2}\frac{u^0\sec\phi^0}{a} - \frac{\Delta t^2}{4}\frac{u^0v^0\tan\phi^0\sec\phi^0}{a^2} \\ & -\frac{\Delta t^3}{24}\frac{u^0\sec\phi^0}{a^3}[(v^0)^2(3\sec^2\phi^0-2) - (u^0)^2\tan^2\phi^0] + \mathrm{O}(\Delta t^4) \qquad (8.54)\\ \phi^0 = {} & \phi^+ - \frac{\Delta t}{2}\frac{v^0}{a} + \frac{\Delta t^2}{8}\frac{(u^0)^2\tan\phi^0}{a^2} \\ & +\frac{\Delta t^3}{48}\frac{(u^0)^2v^0}{a^3}(3\sec^2\phi^0-2) + \mathrm{O}(\Delta t^4), \qquad (8.55)\end{aligned}$$

where we have made use of the identity

$$1 + \tan^2\phi^0 = \sec^2\phi^0.$$

This scheme would be computationally expensive to implement, due to the requirement of calculating trig functions at trajectory mid-points. In the next section we consider a modification which avoids this problem.

8.5.5 Efficient use of Trig Calculations

To solve (8.54) and (8.55) iteratively would require repeated calculations of $\tan\phi^0$ and $\sec\phi^0$ for each iteration, at every grid point, at every time step. A considerable gain in efficiency is made by calculating trig functions just once and storing their values for every grid point. We therefore wish to replace the trig functions in (8.54) and (8.55) with functions of ϕ^+, rather than of ϕ^0. Our approach is to adapt the procedure described in [45].

We begin by considering

$$\phi^+ = \phi^0 + \frac{\Delta t}{2}\dot{\phi}^0 + \frac{\Delta t^2}{8}\ddot{\phi}^0 + \frac{\Delta t^3}{48}\dddot{\phi}^0 + \mathrm{O}(\Delta t^4).$$

For a constant speed trajectory along a great circle we may make use of (8.55) to obtain

$$\phi^+ = \phi^0 + \frac{\Delta t}{2}\frac{v^0}{a} - \frac{\Delta t^2}{8}\frac{(u^0)^2}{a^2}\tan\phi^0 - \frac{\Delta t^3}{48}\frac{(u^0)^2 v^0}{a^3}(3\sec^2\phi^0 - 2) + \mathrm{O}(\Delta t^4). \tag{8.56}$$

The quantity we require, ϕ^0, appears implicitly in this equation. To obtain a solution we assume ϕ^0 has an expansion of the form

$$\phi^0 = \phi^+ + p_1\Delta t + p_2\Delta t^2 + p_3\Delta t^3 + \mathrm{O}(\Delta t^4). \tag{8.57}$$

Substituting (8.57) into (8.56),

$$\begin{aligned}\phi^+ &= \phi^+ + p_1\Delta t + p_2\Delta t^2 + p_3\Delta t^3 - \frac{\Delta t}{2}\frac{v^0}{a} \\ &\quad + \frac{\Delta t^2}{8}\frac{(u^0)^2}{a^2}\tan(\phi^+ + p_1\Delta t + \mathrm{O}(\Delta t^2)) + \mathrm{O}(\Delta t^4).\end{aligned} \tag{8.58}$$

Using

$$\tan(\phi^+ + p_1\Delta t + \mathrm{O}(\Delta t^2)) = \tan\phi^+ + p_1\sec^2\phi^+\,\Delta t + \mathrm{O}(\Delta t^2)$$

and equating powers of Δt, we find

$$p_1 = -\frac{v^0}{2a} \tag{8.59}$$

$$p_2 = \frac{(u^0)^2}{8a^2}\tan\phi^+ \tag{8.60}$$

$$p_3 = -\frac{(u^0)^2 v^0}{24a^3}. \tag{8.61}$$

Hence

$$\phi^0 = \phi^+ - \frac{v^0}{2a}\Delta t + \frac{(u^0)^2}{8a^2}\tan\phi^+\Delta t^2 - \frac{(u^0)^2 v^0}{24a^3}\Delta t^3 + \mathrm{O}(\Delta t^4). \tag{8.62}$$

This is now used to evaluate the various trig functions required in the departure point scheme. The Taylor expansions of tan and sec for a small displacement ϵ about the point x are

$$\begin{aligned}
\tan(x+\epsilon) &= \tan x + \epsilon\sec^2 x + \epsilon^2\sec^2 x\tan x + \mathrm{O}(\epsilon^3) && (8.63)\\
\sec(x+\epsilon) &= \sec x + \epsilon\sec x\tan x + \frac{\epsilon^2}{2}\sec x(\sec^2 x + \tan^2 x) + \mathrm{O}(\epsilon^3). && (8.64)
\end{aligned}$$

These expansions give

$$\begin{aligned}
\tan\phi^0 &= \tan\phi^+ - \frac{v^0}{2a}\sec^2\phi^+\Delta t + \mathrm{O}(\Delta t^2) && (8.65)\\
\sec\phi^0 &= \sec\phi^+ - \frac{v^0}{2a}\sec\phi^+\tan\phi^+\Delta t \\
&\quad + \left[\frac{(u^0)^2\tan^2\phi^+ + (v^0)^2(2\sec^2\phi^+ - 1)}{8a^2}\right]\sec\phi^+\Delta t^2 + \mathrm{O}(\Delta t^3). && (8.66)
\end{aligned}$$

Combining these last two, we also have

$$\tan\phi^0\sec\phi^0 = \tan\phi^+\sec\phi^+ - \frac{v^0}{2a}\sec\phi^+(2\sec^2\phi^+ - 1)\Delta t + \mathrm{O}(\Delta t^2). \tag{8.67}$$

8.5.6 Departure Point Scheme — Final Form

With the above expansions, the first stage of the departure point scheme becomes

$$\begin{aligned}
\lambda^0 &= \lambda^+ - \frac{u^0}{2a}\sec\phi^+\,\Delta t\left\{1 + \left[\frac{(u^0)^2\tan^2\phi^+ - (v^0)^2}{24a^2}\right]\Delta t^2\right\} + \mathrm{O}(\Delta t^4) && (8.68)\\
\phi^0 &= \phi^+ - \frac{v^0}{2a}\Delta t + \frac{(u^0)^2}{8a^2}\tan\phi^+\,\Delta t^2 - \frac{(u^0)^2 v^0}{24a^3}\Delta t^3 + \mathrm{O}(\Delta t^4). && (8.69)
\end{aligned}$$

And the second stage becomes

$$\begin{aligned}
\lambda^- &= \lambda^+ - \Delta t\frac{u^0}{a}\sec\phi^+\left(1 - \frac{v^0}{2a}\tan\phi^+\Delta t\right) \\
&\quad + \frac{u^0\sec\phi^+}{4a^3}\left[(v^0)^2\left(\sec^2\phi^+ - \frac{2}{3}\right) - (u^0)^2\tan^2\phi^+\right]\Delta t^3 + \mathrm{O}(\Delta t^4) && (8.70)\\
\phi^- &= \phi^+ - \Delta t\frac{v^0}{a} - \frac{(u^0)^2 v^0}{8a^3}\left(\sec^2\phi^+ - \frac{2}{3}\right)\Delta t^3 + \mathrm{O}(\Delta t^4). && (8.71)
\end{aligned}$$

This departure point scheme is identical to the Ritchie and Beaudoin scheme, when applied in the context of a two time-level SL advection scheme. The procedure which we have followed here, however, has the potential to be applied in curved spaces other than the surface of the sphere. Possible applications for the method may be found, for instance, in computational general relativity.

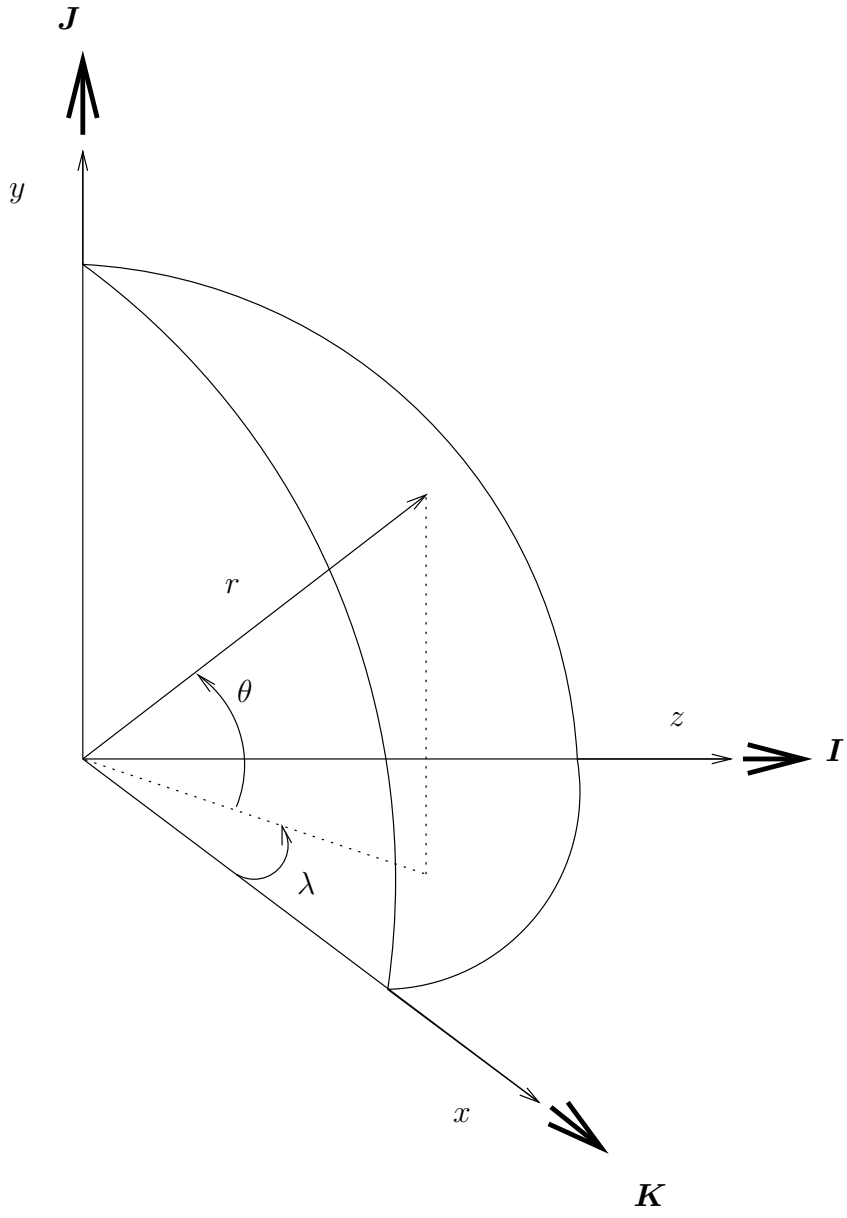

Figure 8.1: Geophysical spherical coordinates

Chapter 9

Mass Conserving Schemes

Before considering conservation of mass in numerical schemes it is first helpful to consider what we require of a numerical simulation. An issue of central importance in any simulation of fluid flow is the degree to which the simulation exhibits the true physics of the flow. Many of the physical principles we expect to be obeyed by a fluid, though not all, can be expressed as the conservation of some quantity. For instance, in the regime of classical physics, we would expect an isolated fluid system to have constant values of total mass, momentum and energy.

In constructing a mathematical model of a fluid flow, we introduce further principles to the description of the flow. These are empirically determined relationships between the various flow variables such as velocity, pressure and temperature. Examples are numerous and include equations of state, Fourier's law of heat flow and Darcy's law for flow in porous media. These phenomenological principles are referred to as constitutive relations, and are required to produce a closed set of equations in the fluid variables. The closed set of equations, which frequently consist of differential and algebraic equations, forms the mathematical model.

Once we have obtained a mathematical model, we can often identify further relationships between variables, or principles involving quantities derived from the basic variables. These principles belong properly to the mathematical equation set alone. Examples of

such dynamical principles, or constraints, which arise in meteorology are the conservation of potential vorticity and geostrophic balance.

Often the picture is not so clear, and it is somewhat arbitrary how we distinguish the mathematical from the physical. What is clear, however, is that the whole picture can change radically when we move on to the stage of solution by numerical methods. Ill-chosen numerics can destroy all the deep properties to be found in the mathematical model. This is a topic of much current interest, see for instance Sanz-Serna & Calvo on numerical methods for Hamiltonian dynamical systems [47].

To obtain a numerical solution of a mathematical model we transform the equations into finite form. This yields a set of algebraic equations in a finite number of unknowns. Together with whatever methods we choose to apply to obtain their solution, these equations constitute the numerical model. No matter how the above discretisation is done, we expect the loss of some, if not all, of the exact balances which might exist in the mathematical model. The extent to which this has a significant effect on predictions obtained from the model depends on what kind of prediction we wish to make.

Lorenz ([25]) identifies two categories of prediction. In the first kind the prediction depends on both the initial data and the boundary conditions. The second kind of prediction depends on the boundary conditions alone. Weather prediction is an example of the first kind. Given an initial state of the atmosphere we wish to predict its development over a period of a few days. For these purposes the boundary conditions can be largely taken as being constant.

Climate prediction falls into the second category of predictions. For the extended simulation periods of climate modelling all memory of the initial conditions is lost. The statistical properties, that is climate, of the prediction depend on the boundary conditions alone. Correct prediction of climate requires comprehensive modelling of all the atmosphere's boundary interactions. This will include coupling to the ocean, land and ice surfaces, interactions with the biota and driving by solar heat. In climate modelling these are the factors of primary concern. Ideally a numerical model would not introduce

spurious physical processes when representing atmospheric dynamics. Such errors could obscure the essential interactions to be represented by the climate model. In practice, however, this ideal can never be reached. But the manner in which we approximate to the ideal will have a strong qualitative effect on the end result.

A given property of a mathematical model, such as a conservation principle, may translate across to the numerical model in one of two ways. It may be the case that a direct numerical analogue of the original property is found to hold for the discrete equations, at all levels of mesh refinement. Alternatively it could be the case that no such property holds, except in the asymptotic limit of refinement. In weather prediction short term accuracy is the desired goal. Under such a regime it may be sufficient to approach certain properties of the flow only in the asymptotic limit, provided that this allows for an advantage in computational speed. The situation is entirely different for climate prediction, where a strong case can be made for the numerical model to possess as many exact analogue properties as possible.

We are now in a position to ask how does the semi-Lagrangian method fare in these respects. At present the answer to this question is still largely incomplete. Until recently no SL method existed which preserved total advected mass. Analysis of deeper dynamical properties is wholly lacking.

For the remainder of this chapter, we shall turn our attention specifically to conservation of mass, and other advected quantities, in numerical schemes.

9.1 Conservation and Continuity

Conservation is a global property. For an isolated fluid system we expect a global quantity, such as total mass, to be conserved. The total mass after any period of time should be precisely the same as it was initially.

If the system is not isolated then quantities which are free to be exchanged with the fluid's surroundings will not be conserved within the system itself. For instance, to a

good level of approximation the atmosphere does not lose or gain mass on a short time scale, so its total mass is a conserved variable. Total heat energy, however, has a greater freedom to fluctuate through radiation processes. Factors such as planetary albedo and the chemical constitution of the atmosphere determine the flow of heat energy from the sun, into the atmosphere and back to space. A mathematical framework is required for describing balance processes such as these.

Variables of state for a macroscopic system fall into two classes [33]. Extensive variables are ones which refer to the system as a whole. Total mass and total energy are extensive variables. Variables of the other class are called intensive and describe local properties of the fluid, such as pressure and temperature.

Consider a volume of fluid, V. Let $M(t)$ be the extensive variable corresponding to some property, such as mass, of the fluid volume V. To every such extensive variable there is associated an intensive variable, $m(\boldsymbol{x}, t)$, where the two variables are related by

$$M(t) = \int_V m(\boldsymbol{x}, t)\ d\boldsymbol{x}. \tag{9.1}$$

Next we analyse the time evolution of M into two parts. Internal processes generate a change in M from within V. For instance chemical reactions between atmospheric components can generate new amounts of one substance, while the amounts of the reactants decrease. Secondly the value of M may vary due to exchanges between V and its surroundings. The time rate of change of M is thus reduced to the sum of contributions from internal and external processes :

$$\frac{dM}{dt} = \frac{d_i}{dt}M + \frac{d_e}{dt}M. \tag{9.2}$$

These two processes can be modelled in terms of the intensive variable m, as follows. First, exchanges between V and its environment are represented by defining the flux, $\boldsymbol{F}(\boldsymbol{x}, t)$, which represents the flow of m. The net flow of m out of V is then obtained by integrating the flux over the surface (∂V) of V,

$$\frac{d_e}{dt}M = -\oint_{\partial V} \boldsymbol{F}.d\boldsymbol{S}. \tag{9.3}$$

Secondly, internal processes are modelled by the generation of m through a source term s defined at all points in V,

$$\frac{d_i}{dt}M = \int_V s \; d\boldsymbol{x}. \tag{9.4}$$

Using (9.1), (9.3) and (9.4) equation (9.2) becomes

$$\frac{d}{dt}\int_V m \; d\boldsymbol{x} = -\oint_{\partial V} \boldsymbol{F}.d\boldsymbol{S} + \int_V s \; d\boldsymbol{x}. \tag{9.5}$$

Applying Gauss' Theorem to the first term on the right-hand side of (9.5), and exploiting the arbitrariness of V, we obtain the balance equation for m in differential form :

$$\frac{\partial m}{\partial t} = -\boldsymbol{\nabla}.\boldsymbol{F} + s. \tag{9.6}$$

The system is conservative if the source term s is identically zero. In this case changes in M depend solely on the flow across the boundary ∂V. If in addition the system is isolated then we have

$$\frac{dM}{dt} = 0. \tag{9.7}$$

It is important to realise that the conservation equation (9.7) by itself does not specify the way in which M is conserved. For this we require the modelling in terms of the intensive variable m, which led to equation (9.6). This equation expresses the continuity of physical quantities such as mass and electric charge.

9.2 Eulerian Conservative Schemes

Conservative Eulerian schemes can be produced by using the control-volume method of discretisation. The region occupied by the fluid is partitioned into discrete computational cells. Fluid variables are represented by their cell-average values in each cell. The development of the numerical scheme follows closely the derivation of the Eulerian balance equation (9.6) in the previous section. That equation is quite general, and could be applied equally to any extensive fluid variable. For clarity we shall now restrict our attention to the case where this variable represents the mass distribution of some quantity, which

is being carried along within the fluid. For instance this could be a gas being released into the air-stream from an industrial outlet. As a further restriction we assume that this substance is merely passively advected, without any other processes affecting its motion. Towards the end of the chapter we shall examine the complications which arise in the case of transport of momentum, in which case nonlinearities appearing in (9.6) need to be dealt with appropriately.

Consider a fluid of density $\rho(\boldsymbol{x}, t)$ which occupies a region partitioned into cells $\{\mathcal{C}_j\}$ with corresponding volumes $\{V_j\}$. Define the cell-average value of ρ,

$$\overline{\rho}_j(t) = \frac{1}{V_j} \int \rho(\boldsymbol{x}, t)\ d\boldsymbol{x} \tag{9.8}$$

where

$$V_j = \int_{\mathcal{C}_j} d\boldsymbol{x}.$$

Since mass is a conserved variable, equation (9.6) provides

$$\frac{d}{dt} \int_{\mathcal{C}_j} \rho\ d\boldsymbol{x} = - \oint_{\partial \mathcal{C}_j} \boldsymbol{F}.d\boldsymbol{S} \tag{9.9}$$

where the mass flux is $\boldsymbol{F} = \rho \boldsymbol{u}$ and $\boldsymbol{u}$ is the fluid velocity.

Combining (9.8) and (9.9) we obtain the evolution equation for the cell-average density,

$$\frac{d\overline{\rho}_j}{dt} = -\frac{1}{V_j} \oint_{\partial \mathcal{C}_j} \boldsymbol{F}.d\boldsymbol{S}. \tag{9.10}$$

This equation indicates that the cell-average densities change according to exchanges of mass between each cell and its surroundings. By discretising this equation in such a way that exchanges of mass between cells are balanced, we obtain a mass-conserving scheme.

A particularly simple class of schemes with this conservation property take the form

$$\frac{\overline{\rho}_j^{n+1} - \overline{\rho}_j^n}{\Delta t} = -\frac{1}{V_j} \sum_{i \neq j} \mathcal{F}_{ij} \tag{9.11}$$

where $\mathcal{F}_{ij}$ is a numerical flux function representing the flow of mass from cell i to cell j. Define total mass at time t_n to be

$$M_n = \sum_j \overline{\rho}_j^n V_j. \tag{9.12}$$

Following (9.5), with $s = 0$, we require

$$M_{n+1} = M_n + \text{boundary terms.} \tag{9.13}$$

This balance is obtained by a natural constraint on the flux function

$$\mathcal{F}_{ji} = -\mathcal{F}_{ij}. \tag{9.14}$$

From (9.11) and (9.12) we obtain

$$M_{n+1} = M_n - \Delta t \sum_j \sum_{i \neq j} \mathcal{F}_{ij}.$$

Interchanging the order of summations,

$$\sum_j \sum_{i \neq j} \mathcal{F}_{ij} = \sum_i \sum_{j \neq i} \mathcal{F}_{ij}.$$

However, if we merely swap the roles of the indices and then apply (9.14),

$$\sum_j \sum_{i \neq j} \mathcal{F}_{ij} = \sum_i \sum_{j \neq i} \mathcal{F}_{ji} = -\sum_i \sum_{j \neq i} \mathcal{F}_{ij}.$$

Hence

$$\sum_j \sum_{i \neq j} \mathcal{F}_{ij} = -\sum_j \sum_{i \neq j} \mathcal{F}_{ij}.$$

So we find that the sum of flux terms is zero, and the mass M_n remains constant at its initial value. If the system is not closed there will be extra contributions to the sum from cells at the boundary of the region of flow. The local nature of mass balance can also be built into this numerical framework. For this purpose a distance function must first be defined for the separation between cell centres or centroids :

$$d(\mathcal{C}_i, \mathcal{C}_j) = 0 \qquad \text{if} \qquad \mathcal{C}_i = \mathcal{C}_j$$

$$d(\mathcal{C}_i, \mathcal{C}_j) > 0 \quad \text{otherwise.}$$

The difference scheme is then said to be local if the numerical flux function satisfies

$$\mathcal{F}_{ij} = 0 \quad \text{whenever} \quad d(\mathcal{C}_i, \mathcal{C}_j) > R, \tag{9.15}$$

where R is some specified finite radius.

In conclusion we see that the cancellation property of the numerical fluxes provides conservation of discrete mass. And that the conservative Eulerian discretisation (9.10), (9.14), (9.15) models the local property of continuity (9.6).

9.3 Lagrangian Conservative Scheme

We shall attempt to construct a conservative Lagrangian scheme by using once again the control-volume approach. Yet we must also employ a Lagrangian description of the fluid, and attach our attention to the motion of individual fluid particles. To this end, let $\mathcal{L}$ be an indexing set for a given collection of fluid particles. The control-volume is identified with the region occupied at any instant by these particles. The density of the fluid is $\rho(\boldsymbol{a}, t)$, where $\boldsymbol{a}$ is the indexing coordinate for the fluid particles. That is, $(\boldsymbol{a}, t)$ are Lagrangian coordinates for the fluid. (For instance we could use a particle's position at some reference time t_0 as an identifying label : $\boldsymbol{a} \equiv \boldsymbol{x}(t_0)$.)

To obtain the total mass contained within the control-volume, we must first associate a volume element, $d\tau(\boldsymbol{a}, t)$, to each particle. By construction no fluid particles enter or leave the control-volume, so the total mass within the volume remains constant. This is precisely the statement of mass conservation in Lagrangian form,

$$\frac{d}{dt} \int_{\boldsymbol{a} \in \mathcal{L}} \rho(\boldsymbol{a}, t)\ d\tau(\boldsymbol{a}, t) = 0. \tag{9.16}$$

The notation D/Dt for the time derivative is not used in this section, since only Lagrangian variables will be used here.

Following the procedure used in deriving a conservative Eulerian scheme, let

$$\overline{\rho} = \frac{1}{V(t)} \int_{\mathcal{L}} \rho\ d\tau \tag{9.17}$$

where

$$V(t) = \int_{a \in \mathcal{L}} d\tau(\boldsymbol{a}, t). \tag{9.18}$$

It is clear from (9.18) that the Lagrangian control-volume is time dependent. This must be the case since it is comoving with the fluid in a Lagrangian description of the flow. A complication arises when we wish to base numerical schemes on (9.17) and (9.18), owing to this time dependency. For if we differentiate (9.17) with respect to time to obtain an evolution equation for $\overline{\rho}$, the resulting equation will contain a term involving the time derivative of V. Any semi-Lagrangian discretisation would then be obliged to to

handle these volume derivatives as source terms. A scheme involving more conventional flow variables can be obtained by using the equivalency of volume change and velocity divergence,

$$\frac{1}{V}\frac{dV}{dt} \equiv \overline{\boldsymbol{\nabla}.\boldsymbol{u}}.$$

However such schemes do not possess the exact numerical conservation property (9.13).

A Lagrangian control-volume formulation of the mass conservation equation, (9.16) better suited to semi-Lagrangian discretisation, is obtained by working with the extensive quantity $\overline{\rho}V$. Using the Lagrangian conservation equation (9.16) together with the definition (9.17), the evolution equation for $\overline{\rho}V$ is found to be simply

$$\frac{d}{dt}(\overline{\rho}V) = 0. \tag{9.19}$$

An approach to conservative schemes along these lines has been pursued by Rančić[42] and more recently by Plante and Laprise [20].

9.4 The Rezoning Method of Rančić, Plante and Laprise

Using subscript a to denote arrival quantities and d for departure, a straightforward semi-Lagrangian discretisation of (9.19) yields

$$\overline{\rho}_a V_a = \overline{\rho}_d V_d. \tag{9.20}$$

In any semi-Lagrangian scheme there is a fixed (Eulerian) grid on which the numerical solution is to be held as grid-point data. Identifying V_a with a cell of the Eulerian grid will result in an upstream scheme. Alternatively a downstream scheme results from taking V_d to be a grid cell.

9.4.1 Upstream Scheme

In the following we shall use ρ_j^n to denote the cell-average value ($\overline{\rho}$) of ρ in cell j, at time t_n. Let V_j be the control-volume associated with cell j. Quantities associated with the

upstream departure region of cell j are labelled subscript $d(j)$. The scheme (9.20) now takes the form

$$\rho_j^{n+1} V_j = \rho_{d(j)}^n V_{d(j)}^n. \tag{9.21}$$

We shall use $V_{d(j)}^n$ to refer to both the departure region and its volume, allowing context to dictate the intended meaning.

Just as in the standard semi-Lagrangian framework, the data required by this scheme at the departure time level (t_n) are not immediately available. From the representation of ρ on the Eulerian grid, $\{\rho_j^n, V_j\}$, a Lagrangian representation $\{\rho_{d(j)}^n, V_{d(j)}^n\}$ must be found. For general fluid simulations this can only be achieved numerically. Interpolation is used to approximate $\rho_{d(j)}^n$, and $V_{d(j)}^n$ must be approximated by some suitable geometric construction. The way in which these two quantities are represented numerically will determine both the accuracy and conservation properties of the scheme. Rančić, and Plante and Laprise incorporate conservation of mass into (9.21) in the following way.

Neglecting any flow across the boundaries of the model domain, the numerical statement of mass conservation (9.13), using the variables currently defined, becomes

$$\sum_j \rho_j^{n+1} V_j = \sum_j \rho_j^n V_j. \tag{9.22}$$

The treatment of inflow and outflow boundaries does not introduce any particular problems in the following. So for the present, we assume the velocity field of the flow maps the model domain onto itself. This is the case, for example, in global simulations on the sphere.

From (9.21) we see that the conservation requirement (9.22) is satisfied if

$$\sum_j \rho_{d(j)}^n V_{d(j)}^n = \sum_j \rho_j^n V_j. \tag{9.23}$$

Equation (9.23) is a statement of mass conservation for the transformation from an Eulerian to a Lagrangian representation of the fluid density, ρ. It shows that conservation in this scheme depends on the numerical treatment of local volume changes, represented by the quantities $V_{d(j)}^n$.

If the total volume of fluid to be modelled remains constant, then this ought to be built into the scheme as a constraint :

$$\sum_j V_j = \sum_j V^n_{d(j)} \text{ for all } n.$$

However, we may go beyond this global constraint and identify a natural local constraint. From the theory of differential equations we know that fluid particle trajectories do not intersect one another. This can be represented numerically by requiring the approximations of the regions $V^n_{d(j)}$ to form a partition of the whole domain. That is, they should cover the domain without gaps or overlaps. The result is a collection of control-volumes whose structure may be associated with a (usually irregular) grid. For instance, such a grid is formed by the edges of control-volumes. This grid approximates to the true Lagrangian grid, formed between the regions which flow into separate Eulerian cells after one time-step, figure (9.1).

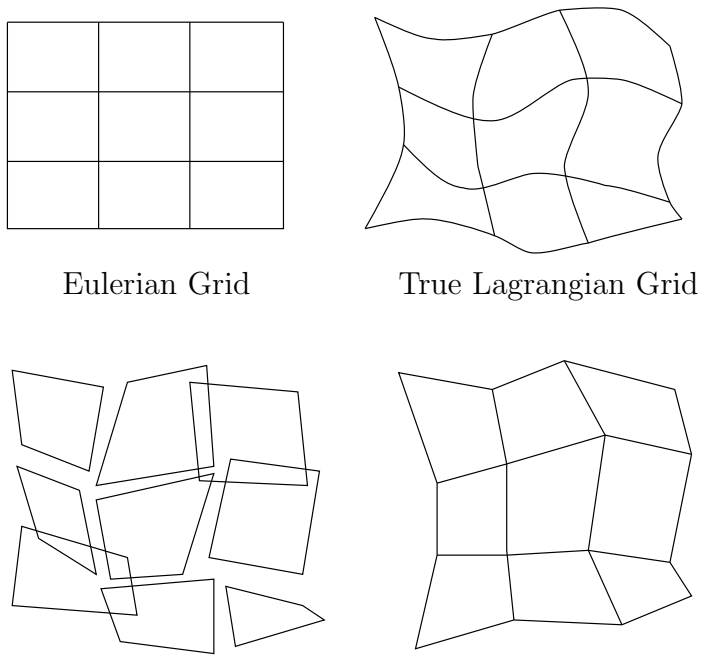

Figure 9.1: Control Volumes and Grids

The process of numerically transferring a discrete representation of a function from one grid to another, both occupying the same domain, is known as *rezoning*. The rezoning

constraint, described above, allows the right-hand side of (9.21) to be split in two parts for efficient numerical evaluation. The splitting results from identifying the regions of intersection between the Lagrangian and Eulerian control-volumes/grid cells. We shall next describe how this allows the right-hand side of (9.21) to be discretised in such a way that (9.23) is satisfied.

Since the whole rezoning process involves data only at the departure time level t_n, we shall drop the n superscript for the remainder of this description.

In general $V_{d(j)}$ will overlap with more than one grid cell, perhaps containing some entirely. Let C_{ij} be the region of intersection between V_i and $V_{d(j)}$. By identifying the overlap regions, we split the right-hand side of (9.21),

$$\rho_{d(j)}V_{d(j)} = \sum_{\substack{i \\ C_{ij}=V_i}} \rho_i V_i + \sum_{\substack{i \\ C_{ij}\neq\emptyset \\ C_{ij}\neq V_i}} \int_{V_i\cap V_{d(j)}} \rho_i(\boldsymbol{x})\ d\boldsymbol{x}. \tag{9.24}$$

The first summation in (9.24) comprises contributions from all those grid cells V_i which lie entirely within $V_{d(j)}$. It can be evaluated directly. Regions of partial overlap are accounted for in the second summation. To evaluate this second group of terms numerically we must make an assumption about the way in which ρ varies within a single cell. In order to do this we introduce the functions $\rho_i(\boldsymbol{x})$, to represent sub-cell variation of the numerical solution within each cell $\mathcal{C}_i$ The only data available, within a scheme, from which to construct such representations are the cell average values $\{\rho_j\}$. From these averaged values the piecewise function $\rho(\boldsymbol{x})$ (defined as $\rho_i(\boldsymbol{x})$ on each cell $\mathcal{C}_i$) must somehow be recovered.

Conservation of mass now becomes a constraint on the reconstruction, or recovery, of $\rho(\boldsymbol{x})$. Since $\{V_j\}$ and $\{V_{d(j)}\}$ partition the same domain (9.24) will satisfy (9.23) if

$$\int_{V_i} \rho_i(\boldsymbol{x})\ d\boldsymbol{x} = \rho_i V_i. \tag{9.25}$$

Given the above design requirements many schemes are possible. Plante and Laprise follow Rančić in using piecewise parabolic recovery for $\rho(\boldsymbol{x})$. With constant advection

speed this is equivalent to using cubic interpolation in a standard semi-Lagrangian scheme, Plante [35].

A practical rezoning strategy, to obtain the departure regions $V_{d(j)}$, is given by connecting departure points of cell vertices with straight edges. This produces departure zones which are general quadrilaterals in two dimensions. With this geometry analytic evaluation of the integrals appearing in the second term of (9.24) is possible, though cumbersome. This suggests that greater sophistication in the representation of departure zones would be computationally inefficient.

In three dimensions the corresponding departure regions are three-dimensional figures with bilinear faces. Identifying regions of intersection with Eulerian grid cells — and performing exact integration — in this geometry would appear to be a serious challenge. Further geometric simplifications are likely to be needed.

9.4.2 Downstream Scheme

It is possible to invert the procedure outlined for the upstream scheme, and obtain a downstream discretisation for the $\overline{\rho}$ conservation equation (9.19). And again this scheme takes the basic form (9.20). In such a scheme the departure region, V_d, is taken to be a grid-cell. From this beginning, the aim is to construct a numerical procedure for distributing the mass contained in this cell to grid-cells at the arrival time. Just as before, the transfer of mass between time levels depends on how we model local volume changes. But in the place of interpolation of the field variable we now have redistribution between grid-cells. The numerical scheme takes the form

$$\rho_j^{n+1} V_j = \sum_j m_{ij}^n, \tag{9.26}$$

where m_{ij} is the mass of fluid which flows from cell i, at time t_n, to cell j, at time t_{n+1}.

The mass distribution, m_{ij}, is calculated in two stages. First the region within cell i which flows to cell j must be identified. Secondly the mass within this region is calculated by integration of the recovered sub-grid distribution, $\rho(\boldsymbol{x})$.

A computational barrier is presented by the first stage of the scheme. If the geometry used in the upstream scheme were to be used to approximate the arrival region in the downstream scheme, the computational cost would be prohibitive. Plante [35] proposes a simplified geometry, in which the arrival region is approximated by a region whose sides remain aligned with the grid.

9.4.3 Performance of Schemes

The schemes developed by Rančić, Plante and Laprise offer simulations comparable in accuracy to standard semi-Lagrangian methods. In addition conservation of advected quantities is maintained to within machine precision. This is achieved at the expense of computational speed. The upstream scheme incurs the largest overheads, requiring three times the cpu usage of a standard scheme for one test case reported by Laprise & Plante [20]. The downstream scheme only increases cpu usage by a factor of 1.5, compared to a standard semi-Lagrangian scheme, in the same test. However the downstream scheme is less accurate than the upstream scheme.

sectionQuasi-Conservative Schemes

Finally in this chapter, we describe an elegant method for introducing conservation into semi-Lagrangian schemes. This method, due to Priestley [38], builds on the monotone interpolation method of Bermejo and Staniforth, described in section 4.3. In their QMSL method, correction terms are calculated which improve the accuracy of an existing monotone SL scheme. Priestley's method calculates these correction terms under the further constraint of conservation of mass. The result is a scheme termed quasi-conservative semi-Lagrange (QCSL).

We begin by assuming that a monotone and conservative SL solution has been computed up to time t_n, and the solution is next to be advanced to time t_{n+1}. A purely monotone solution is computed using QMSL. As described in section 4.3, this solution is found by adding an accuracy correction to a low order monotone solution. In the notation

of section 4.3, the new solution is to take the form

$$u_j^{n+1} = u_{Lj}^{n+1} + \delta_j^{n+1}, \tag{9.27}$$

where the correction term is derived from a high order non-monotone solution,

$$\delta_j^{n+1} = \alpha_j (u_{Hj}^{n+1} - u_{Lj}^{n+1}). \tag{9.28}$$

The factors α_j are generated by QMSL to be as large as possible, subject to the monotonicity constraint (4.2). In this section the factors produced by QMSL will be denoted $\alpha_j^{\max}$. The QCSL algorithm seeks sub-optimal values of α_j, satisfying the monotonicity requirement $0 \leq \alpha_j \leq \alpha_j^{\max}$ together with some further constraint for conservation of mass.

To make these ideas precise, let V_j be the volume associated with node j of the grid. The mass of the initial distribution may then be defined as

$$M_0 = \sum_j u_j^0 V_j.$$

It is now possible to state conservation of mass as a constraint on the new solution u_j^{n+1} :

$$\sum_j u_j^{n+1} V_j = M_0. \tag{9.29}$$

Next we shall use (9.27) and (9.28) to derive from (9.29) a conservation constraint for α_j.

Let the mass of the low order solution be

$$M_L = \sum_j u_{Lj}^{n+1} V_j.$$

The correction terms δ_j^{n+1} must be chosen so that they exactly meet the mass discrepancy

$$\delta M = M_0 - M_L.$$

That is, we require

$$\sum_j \delta_j^{n+1} V_j = \delta M.$$

Using (9.28) this last balance equation provides the desired constraint for α_j :

$$\sum_j \alpha_j \beta_j = \delta M, \tag{9.30}$$

where

$$\beta_j = (u_{Hj}^{n+1} - u_{Lj}^{n+1})V_j.$$

The task is now to find the largest values of α_j, in the range $0 \leq \alpha_j \leq \alpha_j^{\max}$, which satisfy the conservation condition (9.30). It may be assumed that the QMSL algorithm produces correction terms with a total mass larger than the discrepancy δM. If these two masses were the same, then the QMSL algorithm would, fortuitously, be conservative. In such a case further adjustment to the solution would be unnecessary. Alternatively, if the QMSL corrections have a mass too low to meet the conservation requirement, then a simple change of sign reverses this ordering :

$$\begin{aligned} \beta_j &= (u_{Lj}^{n+1} - u_{Hj}^{n+1})V_j \\ \delta M &= M_L - M_0. \end{aligned}$$

So, it may be assumed that the QMSL algorithm has produced correction factors $\alpha_j^{\max}$, which are such that

$$\sum_j \alpha_j^{\max} \beta_j > \delta M.$$

values i

Priestley's approach to solving the above problem is a four stage algorithm :

1: $\left[\begin{array}{llll} \text{if } \beta_j < 0 & \alpha_j = \alpha_j^{\max} & \text{and } flag(j) = 1 \\ \text{else} & \alpha_j = 0 & \text{and } flag(j) = 0 \end{array} \right.$

2: $M^+ = \delta M - \sum_{flag(j)=1} \alpha_j \beta_j$

3: $\overline{\alpha} = \dfrac{M^+}{\sum_{flag(j)=0} \beta_j}$

4: $\left[\begin{array}{lll} \text{if } \overline{\alpha} < \alpha_j^{\max} & \forall j \text{s.t. } flag(j) = 0 & \\ \text{then} & \alpha_j = \overline{\alpha} & \forall j \text{s.t. } flag(j) = 0 : \text{END} \\ \text{else} & \forall j \text{ s.t. } flag(j) = 0 \text{ and } \overline{\alpha} > \alpha_j^{\max} & \\ & \text{put } \alpha_j = \alpha_j^{\max} \text{ and } flag(j) = 1 & \\ & \text{go to 2:} & \end{array}\right.$

This scheme is termed quasi-conservative, since it may often give slight conservation errors. In practice this doesn't appear to be a significant problem [9]. In Chapter 10 we shall compare this scheme against a fully conservative SL scheme, in the context of linear advection.

Chapter 10

Numerical Experiments with Conservative Schemes

10.1 Introduction

In this chapter we describe a two-dimensional scheme of the form outlined in the previous chapter. This is a downstream scheme, which identifies overlap regions between the Eulerian and Lagrangian grids. These regions, which are polygons, are broken down into triangular regions. Polynomial recovery is applied to the finite difference data on the Eulerian grid. The polynomials are then integrated over each triangle to transfer the representation of the data to the Lagrangian grid. Rather than using some approximation technique for the integration, we shall integrate the polynomials exactly. The resulting scheme conserves mass to machine accuracy.

A comparison is made between this scheme and the quasi-conservative scheme of Priestley, described in Chapter 5. Both these schemes are compared against two non-conservative schemes. The first of these is a basic semi-Lagrangian scheme without any control for monotonicity. The second scheme is the same as the first, but with the addition of the quasi-monotone method of Bermejo and Staniforth, described in Chapter 2. All of the schemes use Lagrange polynomials for interpolation, and as basis functions for the

remapping scheme. The same method of trajectory calculation is used for all the schemes.

10.2 Remapping Scheme

A remapping scheme has two requirements :

- a method for calculating overlap regions of Eulerian and Lagrangian grids
- integration of a representation of the solution in the overlap regions.

Schemes other than the semi-Lagrangian schemes described here employ algorithms for these or similar functions. For instance the Lagrange-Galerkin method [18], [17].

10.2.1 Exact Integration

The integration method to be described here is based on a technique for exact integration of polynomials over arbitrary triangles. An extension of this method to three-dimensional geometry is possible.

As described in the introduction to this chapter, the overlap regions between the two grids are analyzed into triangles. A polynomial recovery of the finite difference data must be integrated over each triangle. In this section we describe a technique for the exact integration of a bi-variate polynomial over an arbitrary triangle. This technique consists of five stages :

[**1**] Express the polynomial as a product of linear factors

[**2**] Find the transformation for mapping the unit right-angled triangle onto the given triangle

[**3**] Apply the transformation found in [**2**] to each linear factor of the polynomial

[**4**] Collect terms of the same power within the polynomial

[**5**] Integrate over unit triangle.

In stage [**1**] the polynomial is to be expressed in the form

$$p_n(x,y) = \prod_{r=1}^{n}(a_r x + b_r y + c_r). \tag{10.1}$$

If the polynomial we wish to integrate is known in terms of Lagrange basis functions, then this first stage is trivial since each basis function is of the form (10.1). The integration method may therefore be applied to each basis function separately. Summation of the integrated basis functions, multiplied by the appropriate expansion coefficients, then yields the required integral.

Assuming the polynomial has been expressed in the form required by stage [**1**], stage [**2**] is to calculate the transformation. Let the vertices of the triangle be labeled 1,2,3. And let the coordinates of the vertices be (x_1, y_1), (x_2, y_2) and (x_3, y_3) respectively. The transformation maps the unit triangle in a canonical space with coordinates (ξ, η), to the triangle 123 in physical space with coordinates (x, y). The canonical triangle has vertices at $(0,0)$, $(1,0)$ and $(0,1)$. A simple transformation maps from the ξ-η triangle to the x-y triangle :

$$\begin{bmatrix} x \\ \\ y \end{bmatrix} = \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix} \begin{bmatrix} \xi \\ \\ \eta \end{bmatrix} + \begin{bmatrix} x_1 \\ \\ y_1 \end{bmatrix}.$$

Since we shall be evaluating the integral of the polynomial in the transformed coordinate system, we also require the Jacobian (J) of the transformation. This is simply the magnitude of the determinant of the above matrix and also the area of the triangle,

$$J = \left|(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)\right|.$$

In the following we use a more compact notation, with the transformation represented by

$$\begin{bmatrix} x \\ \\ y \end{bmatrix} = \begin{bmatrix} J_{xx} & J_{xy} \\ \\ J_{yx} & J_{yy} \end{bmatrix} \begin{bmatrix} \xi \\ \\ \eta \end{bmatrix} + \begin{bmatrix} J_x \\ \\ J_y \end{bmatrix}. \tag{10.2}$$

Step [**3**] requires the transformation to be applied to each linear factor of the polynomial, in order to find the polynomial in the transformed coordinates. Let $\tilde{p}(\xi,\eta)$ be the transformed polynomial. From (10.1) and (10.2) we obtain

$$\begin{aligned}\tilde{p}_n(\xi,\eta) &= \prod_{r=1}^{n}\left[a_r(J_{xx}\xi+J_{xy}\eta)+b_r(J_{yx}\xi+J_{yy}\eta)+J_x a_r+J_y b_r+c_r\right] \\ &\equiv \prod_{r=1}^{n}(A_r\xi+B_r\eta+C_r) \qquad (10.3)\end{aligned}$$

where

$$\begin{aligned}A_r &= J_{xx}a_r+J_{yx}b_r \\ B_r &= J_{xy}a_r+J_{yy}b_r \\ C_r &= J_x a_r+J_y b_r+c_r.\end{aligned}$$

The final step will involve integrating this transformed polynomial over the unit triangle. This task is considerably simplified by step [**4**], in which terms of the same power are collected together. So, we wish to find the coefficients Q_{rs} for which

$$\tilde{p}_n(\xi,\eta)=\sum_{r=0}^{n}\sum_{s=0}^{k-r}Q_{rs}\xi^r\eta^s$$

A recurrence procedure offers a simple means of calculating these coefficients.

The recurrence builds through polynomials of increasing order $k=1,\ldots,n$, by successively multiplying by linear factors, until the polynomial (10.3) is reached. The first level of the recurrence has coefficients $Q_{rs}^{[0]}$, where $r=0,1$ and $s=0,r$. These coefficients define the first-level polynomial, which is simply equal to the first linear factor of (10.3) :

$$Q_{10}^{[0]}\xi+Q_{01}^{[0]}\eta+Q_{00}^{[0]}\equiv A_1\xi+B_1\eta+C_1,$$

from which we obtain

$$\begin{aligned}Q_{00}^{[0]} &= C_1 \\ Q_{01}^{[0]} &= B_1 \\ Q_{10}^{[0]} &= A_1.\end{aligned}$$

Next, assume the recurrence has been completed up to level k, and we wish to add level $k+1$. If the level k polynomial is

$$\tilde{p}_k(\xi,\eta) = \prod_{r=1}^{k}(A_r\xi + B_r\eta + C_r) \equiv \sum_{r=0}^{k}\sum_{s=0}^{k-r} Q_{rs}^{[k]}\xi^r\eta^s,$$

then the polynomial at the next level of the recurrence is

$$\begin{aligned}
\tilde{p}_{k+1}(\xi,\eta) &= (A_{k+1}\xi + B_{k+1}\eta + C_{k+1})\tilde{p}_k(\xi,\eta) \\
&= (A_{k+1}\xi + B_{k+1}\eta + C_{k+1})\sum_{r=0}^{k}\sum_{s=0}^{k-r} Q_{rs}^{[k]}\xi^r\eta^s \\
&\equiv \sum_{r=0}^{k+1}\sum_{s=0}^{r} Q_{rs}^{[k+1]}\xi^r\eta^s.
\end{aligned}$$

To find the new coefficients $Q_{rs}^{[k+1]}$, we work with $\tilde{p}_k$ written in the form of a univariate polynomial in ξ :

$$\tilde{p}_k(\xi,\eta) = \sum_{r=0}^{k} q_r^{[k]}(\eta)\xi^r,$$

where the coefficients are polynomials in η,

$$q_r^{[k]}(\eta) = \sum_{s=0}^{k-r} Q_{rs}^{[k]}\eta^s. \tag{10.4}$$

Using this form we calculate $\tilde{p}_{k+1}$:

$$\begin{aligned}
\tilde{p}_{k+1}(\xi,\eta) &= (A_{k+1}\xi + B_{k+1}\eta + C_{k+1})\sum_{r=0}^{k} q_r^{[k]}(\eta)\xi^r \\
&= A_{k+1}q_k^{[k]}(\eta)\xi^{k+1} + \sum_{r=1}^{k}\left[A_{k+1}q_{r-1}^{[k]}(\eta) + B_{k+1}\eta q_r^{[k]}(\eta)\right]\xi^r + \\
&\quad B_{k+1}\eta q_0^{[k]}(\eta) + C_{k+1}\sum_{r=0}^{k} q_r^{[k]}(\eta)\xi^r,
\end{aligned}$$

from which we obtain recurrence relations for the coefficients of ξ^r :

$$\begin{aligned}
q_{k+1}^{[k+1]}(\eta) &= A_{k+1}q_k^{[k]}(\eta) \\
q_r^{[k+1]}(\eta) &= A_{k+1}q_{r-1}^{[k]}(\eta) + B_{k+1}\eta q_r^{[k]}(\eta) + C_{k+1}q_r^{[k]}(\eta) \quad r = 1,\ldots,k \\
q_0^{[k+1]}(\eta) &= B_{k+1}\eta q_0^{[k]}(\eta) + C_{k+1}q_0^{[k]}(\eta).
\end{aligned}$$

Applying the definition (10.4) of the $\tilde{q}_k(\eta)$ polynomials in terms of the coefficients Q_{rs}, we obtain recurrence relationships for the coefficients :

$$
\begin{aligned}
Q^{[k+1]}_{k+1\,0} &= A_{k+1} Q^{[k]}_{k\,0} \\
Q^{[k+1]}_{r\,0} &= A_{k+1} Q^{[k]}_{r-1\,0} + C_{k+1} Q^{[k]}_{r\,0} && r = 1, \ldots, k \\
Q^{[k+1]}_{rs} &= A_{k+1} Q^{[k]}_{r-1\,s} + B_{k+1} Q^{[k]}_{r\,s-1} + C_{k+1} Q^{[k]}_{rs} && r = 1, \ldots, k; \;\; s = 1, \ldots, k-r \\
Q^{[k+1]}_{r\,k+1-r} &= A_{k+1} Q^{[k]}_{r-1\,k+1-r} + B_{k+1} Q^{[k]}_{r\,k-r} && r = 1, \ldots, k \\
Q^{[k+1]}_{0\,k+1} &= B_{k+1} Q^{[k]}_{0\,k} \\
Q^{[k+1]}_{0\,s} &= B_{k+1} Q^{[k]}_{0\,s-1} + C_{k+1} Q^{[k]}_{0\,s} \\
Q^{[k+1]}_{0\,0} &= C_{k+1} Q^{[k]}_{0\,0}.
\end{aligned}
$$

Once the coefficients Q_{rs} have been found, stage [**5**] of the algorithm is to perform the integration,

$$I = \int_0^1 \int_0^{1-\xi} \sum_{r=0}^{n} \sum_{s=0}^{n-r} Q_{rs} \xi^r \eta^s \, d\eta \, d\xi.$$

Simply rearranging this, we have

$$I = \sum_{r=0}^{n} \sum_{s=0}^{n-r} Q_{rs} I_{rs},$$

where we now have to evaluate the integrals

$$I_{rs} = \int_0^1 \int_0^{1-\xi} \xi^r \eta^s \, d\eta \, d\xi, \;\; r = 0, \ldots, n; \; s = 0, \ldots, n-r.$$

The following provides a reasonably efficient way to evaluate these integrals :

$$\begin{aligned} I_{rs} &= \int_0^1 \xi_r \int_0^{1-\xi} \eta^s \, d\eta \, d\xi \\ &= \frac{1}{s+1} \int_0^1 \xi^r (1-\xi)^{s+1} \, d\xi \\ &= \frac{1}{s+1} \int_0^1 \xi^{s+1} (1-\xi)^r \, d\xi \\ &= \frac{1}{s+1} \sum_{k=0}^{r} (-1)^k \begin{pmatrix} r \\ \\ k \end{pmatrix} \frac{1}{s+k+2}. \end{aligned}$$

The calculations required to evaluate the final addition may be reduced by careful computer programming. Finally, multiplication by the Jacobian of the transformation provides the integral of the original polynomial over the triangle 123,

$$\int_{123} p_n(x,y) \, dx \, dy = JI.$$

This completes the integration algorithm.

It should be noted that this method may be extended to any number of dimensions. In three dimensions, stages [**1**] to [**5**] may be used to integrate polynomials of the form

$$p_n = \prod_{r=1}^{n} (a_r x + b_r y + c_r z + d_r)$$

over arbitrary tetrahedra. If such a tetrahedron has vertices at the points

$$(x_i, y_i, z_i), \quad i = 1, 2, 3, 4,$$

then the appropriate transformation at stage [**2**] is

$$\begin{bmatrix} x \\ \\ y \\ \\ z \end{bmatrix} = \begin{bmatrix} x_2 - x_1 & x_3 - x_1 & x_4 - x_1 \\ \\ y_2 - y_1 & y_3 - y_1 & y_4 - y_1 \\ \\ z_2 - z_1 & z_3 - z_1 & z_4 - z_1 \end{bmatrix} \begin{bmatrix} \xi \\ \\ \eta \\ \\ \omega \end{bmatrix} + \begin{bmatrix} x_1 \\ \\ y_1 \\ \\ z_1 \end{bmatrix},$$

where the canonical space now has coordinates (ξ, η, ω). For a three dimensional scheme the determination of the overlap regions between the Eulerian and Lagrangian grids will be a substantial problem. The three dimensional scheme will not be considered further here.

In the next section we present the numerical results of applying conservative semi-Lagrangian algorithms to an advection test.

10.3 Results of Comparison Tests

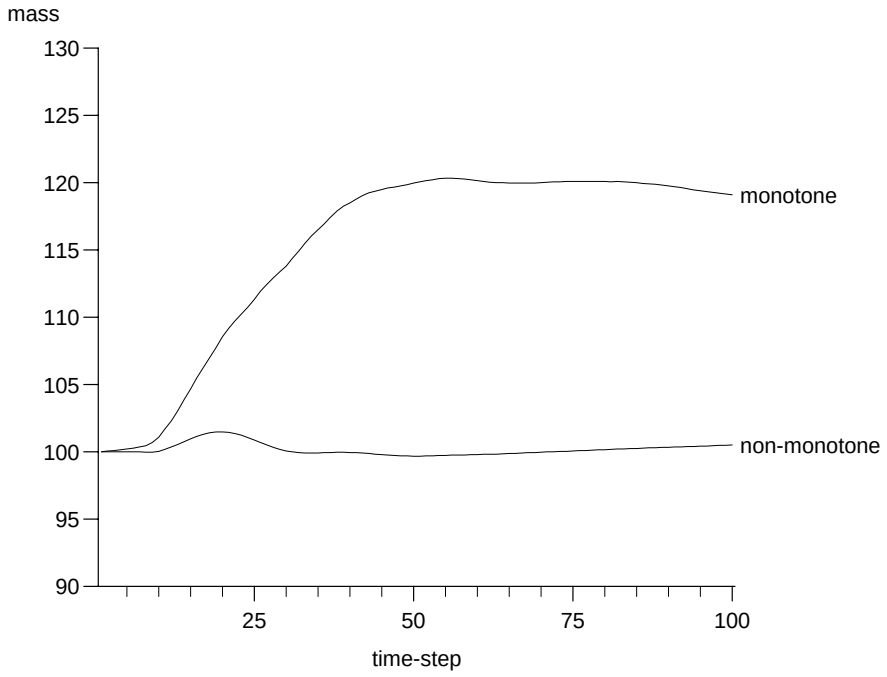

Figure 10.1: Mass histories for non-conservative schemes

Figure 10.1 shows the loss of mass conservation which results when two SL schemes are applied to the Smolarkiewicz deformational test [50]. The problem is two dimensional, and consists of an initial mass distribution, which is purely advected by a highly deformational velocity field. The domain is a square of side $L = 100$ mesh units, and the initial scalar distribution is a cone with base radius 15 units, centred on the domain. The flow

field is obtained from a stream-function,

$$\psi(x, y) = A \sin(kx) \cos(ky),$$

where $A = 8$ and $k = 4\pi/L$, with the origin located at the bottom left corner of the square. The analytical solution for this problem [56], consists of spiral distributions which become wound tighter and tighter within square vortex cells. This provides a challenge for schemes without variable resolution. As the motion progresses, filaments are produced within the distribution which are too thin to be resolved by the grid. This may provide a source for nonlinear instabilities in some schemes [51]. Our present interest is in how such a flow may affect conservation of mass in SL schemes.

Since the only process present in this problem is advection, the total mass remains constant for all time. Figure 10.1 plots the percentage of the initial mass present, against the time-step number. One hundred time-steps are made, with step length $\Delta t = 2.6376$. After about 20 time-steps, filaments begin to become too narrow to be adequately represented on the grid. After this point no finite difference scheme is able to follow accurately the exact evolution of the distribution. Nevertheless, we would require any numerical scheme to maintain a physically meaningful representation. In particular, the total mass should remain constant. The monotone scheme uses cubic Lagrange interpolations both for interpolating velocity values in the trajectory calculations, and for the advected distribution itself. Three iterations of the implicit mid-point rule were used to determine departure points. The scheme is made monotone by use of the QMSL algorithm described in section 4.3. The non-monotone scheme is the same as the above, except without any extra processing for monotonicity. It can be seen that for this particular problem, the introduction of monotonicity causes a large error in conservation.

Figure 10.2 shows the final field, after 100 time-steps, when the non-monotone scheme is used. It shows a good deal of noise. The field should be non-zero except within the six central cells. Figure 10.3 is the final field from the monotone scheme. It shows a marked reduction in noise, and there is minimal spreading of the disturbance to regions where there should be none. We see that the monotone scheme has desirable properties, except

for its severe lack of conservation.

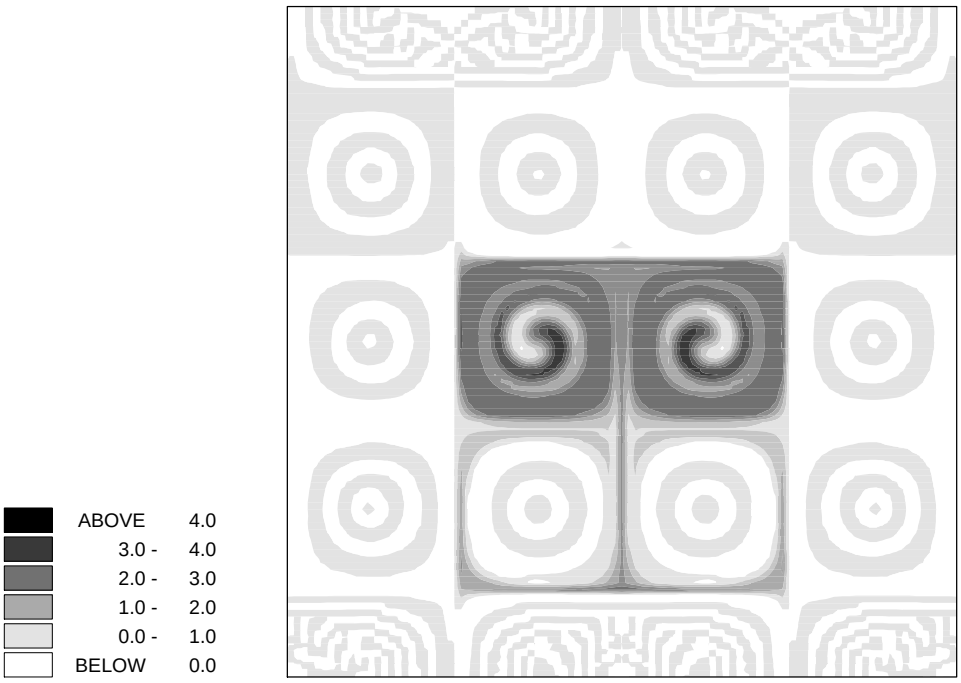

Figure 10.2: Final distribution for non-monotone scheme

Next we apply the QCSL algorithm, described in section 9.4.3, in an attempt to recover conservation in the monotone solution. The solution produced by this quasi-conservative scheme is compared with that obtained using the remapping scheme described in section 10.2. The particular remapping scheme uses quadratic basis functions, and trajectory calculations of the same accuracy as above. As expected, the remapping scheme is mass conserving to machine accuracy, while the QCSL scheme is almost conserving, figure 10.4. However, the solution produced by the remapping method (figure 10.5), appears far more diffusive than the QCSL solution (figure 10.6). In addition the computational time for the remapping solution is almost an order of magnitude greater than for the QCSL algorithm. However, the speed of the remapping scheme could be significantly improved, through the use of more efficient algorithms to calculate the grid overlap regions.

In conclusion, we note that the QCSL scheme offers a simple and efficient means of calculating nearly-conservative numerical solutions. In addition this method may be

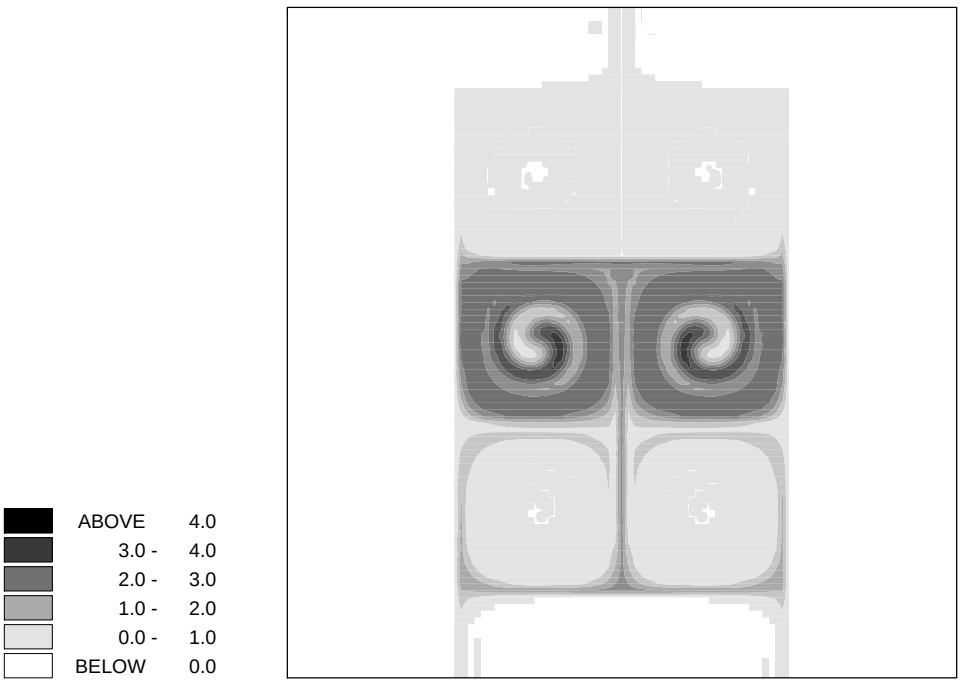

Figure 10.3: Final distribution for monotone scheme

applied directly to any existing SL scheme working on the sphere. This is not true of remapping schemes. The added complexity of identifying overlap regions, between Eulerian and Lagrangian grids on the sphere, poses a significant disadvantage for such schemes.

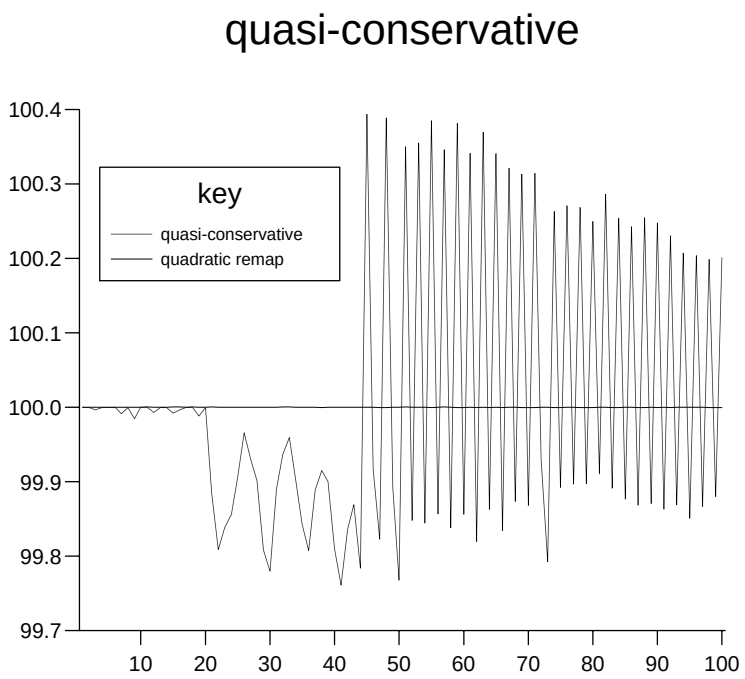

Figure 10.4: Mass histories for conservative SL schemes

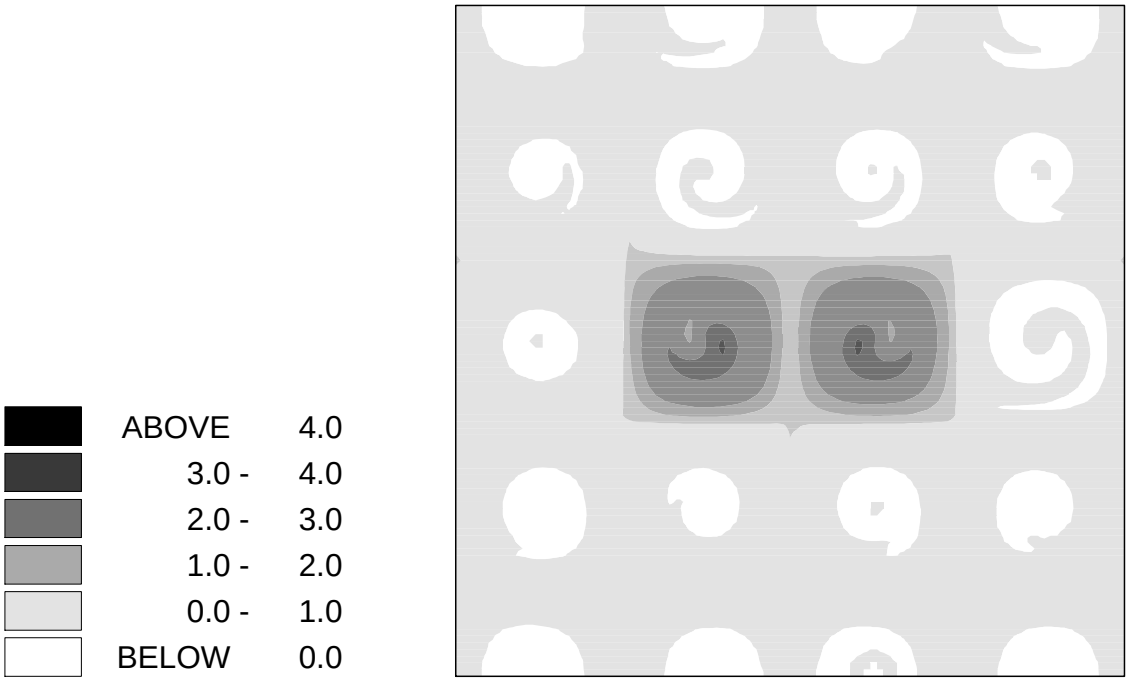

Figure 10.5: Final distribution for remapping scheme

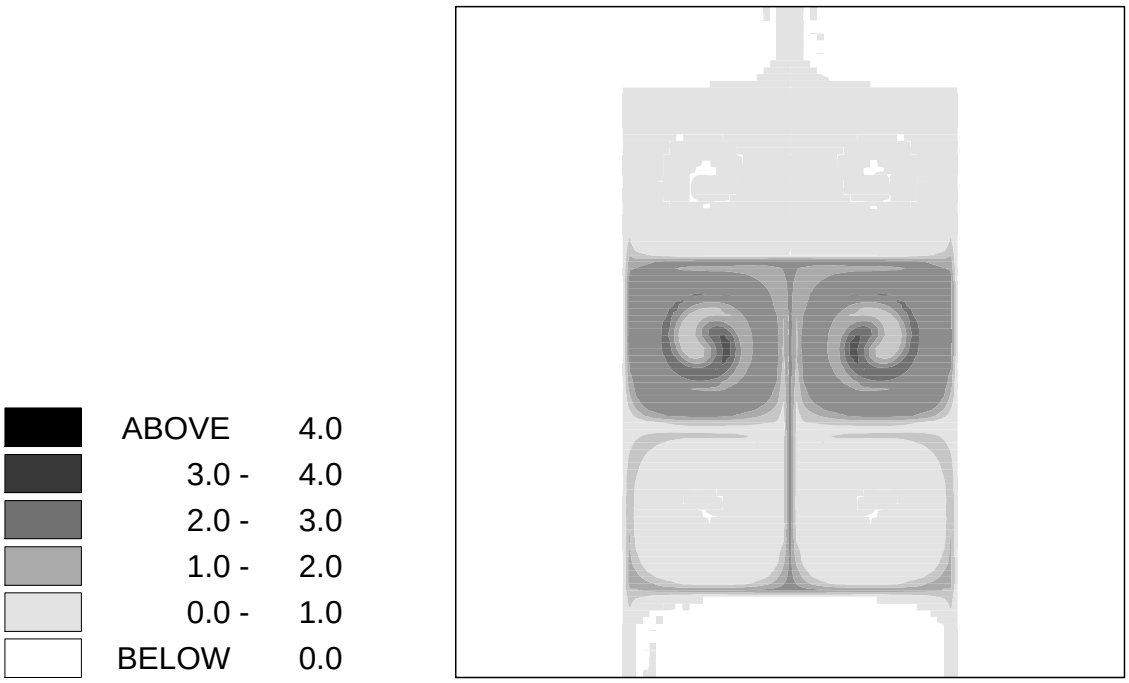

Figure 10.6: Final distribution for QCSL scheme

Chapter 11

Spurious Orographic Resonance

In their 1991 review, Côté and Staniforth highlight the problem of spurious orographic resonance. It has been found that weather simulations employing semi-implicit, semi-Lagrangian discretisations frequently exhibit spurious numerical resonance, triggered by surface topography. Downstream of major mountain ranges, large scale standing wave patterns can become established . Wavelengths of these features are of the order 10^2 or 10^3 kilometres.

In a paper which addresses this problem, Rivest et al. [46] apply the techniques of Fourier analysis to a one-dimensional linearised shallow-water model. We now present the content of this paper.

The same Fourier analysis is applied to both the mathematical model, and to a standard semi-implicit, semi-Lagrangian discretisation of the model. It is important to note that this discretisation is fully centred in time, with the orographic forcing term averaged along trajectories. Analysis of the mathematical model shows that it does indeed support large wavelength standing wave patterns, driven by orography. However these waves occur only when the flow speed is supersonic. Consequently the phenomenon is far from likely to be realised under terrestrial conditions. When the analysis is repeated for the discrete equation set, it too is found to support orographic standing wave patterns. However, in contrast to the analytical case, the conditions under which this numerical

behaviour occurs fall well within the operational range of weather simulation.

The solution proposed in Rivest et al. is to off-centre the weighting between explicit and implicit parts, in the semi-implicit discretisation. For an equation of the form

$$\frac{DF}{Dt} = G, \tag{11.1}$$

a one parameter family of discretisations is considered :

$$\frac{DF}{Dt} \rightarrow \frac{1}{\Delta t}\left(F^n - F^{n-1}\right) \tag{11.2}$$

$$G \rightarrow \left(\frac{1+\delta}{2}\right) G^n + \left(\frac{1-2\delta}{2}\right) G^{n-1} + \left(\frac{\delta}{2}\right) G^{n-2} \tag{11.3}$$

where $H^{n-r} \equiv H(\boldsymbol{x}(t_n - r\Delta t), t_n - r\Delta t)$. This form of discretisation retains the $\mathrm{O}(\Delta t^2)$ accuracy of the centred approximation ($\delta = 0$) by introducing a third time-level. At this new time-level, departure points must be found and interpolations performed there, just as for the $t - \Delta t$ time-level.

Repeating the Fourier analysis of orographically forced standing waves, it is found that for this new discretisation resonance will not occur whenever δ is non-zero. Values of δ away from zero actively inhibit the formation of persistent standing wave patterns. Stability analysis for the new discretisation shows that a necessary condition for stability is

$$\delta \geq 0.$$

The choice $\delta = 1/2$ is recommended by Rivest et al., since this produces some simplification when applied to (11.3) while eliminating orographic resonance.

Before considering further developments of this scheme, we shall outline the various options for calculating departure points at the extra time-level introduced in (11.3).

Two methods for calculating departure points at the new time-level are examined by Rivest et al. In the first method it is assumed that a departure point at time $t - \Delta t$ has already been found using standard methods. The trajectory thus found is extrapolated, along a great-circle arc, back to the $t - 2\Delta t$ time-level to give a departure point there.

The second method requires considerable extra computational time compared to the first. In this approach the extra level of departure points is found using the same

techniques as for the first level. For instance if the departure points at time $t - \Delta t$ are calculated using the implicit mid-point rule,

$$\boldsymbol{x}(t - \Delta t) = \boldsymbol{x}(t) - \Delta t \boldsymbol{u} \left(\frac{1}{2}[\boldsymbol{x}(t) + \boldsymbol{x}(t - \Delta t)], t - \frac{1}{2}\Delta t \right), \tag{11.4}$$

then the same rule is applied to calculating the departure points at time $t - 2\Delta t$. In (11.4) the velocity field at the intermediate time-level is not immediately available, but must be obtained by extrapolation of the field at time-levels $t - \Delta t$ and $t - 2\Delta t$.

For calculating departure points at $t - 2\Delta t$ there are two ways in which the implicit mid-point rule might be applied. In one, a piecewise trajectory between times t and $t - 2\Delta t$ is constructed. In the other, two separate trajectories are calculated.

Considering the piecewise method first, the part of the trajectory between t and $t - \Delta t$ is simply that found by solving (11.4). The part between $t - \Delta t$ and $t - 2\Delta t$ is obtained from

$$\boldsymbol{x}(t - 2\Delta t) = \boldsymbol{x}(t - \Delta t) - \Delta t \boldsymbol{u} \left(\frac{1}{2}[(\boldsymbol{x}(t - \Delta t) + \boldsymbol{x}(t - 2\Delta t)], t - \frac{3}{2}\Delta t \right). \tag{11.5}$$

This formula too involves evaluation of velocities at an intermediate time-level. Rather than extrapolating from previous time-levels, interpolation may be used here between data at times $t - \Delta t$ and $t - 2\Delta t$.

However, an alternative formulation of the implicit mid-point rule avoids the complication of extrapolation or interpolation of data. Since the velocity field is known at time-level $t - \Delta t$, this can be used in an approximation of a separate trajectory across two time-levels :

$$\boldsymbol{x}(t - 2\Delta t) = \boldsymbol{x}(t) - 2\Delta t \boldsymbol{u} \left(\frac{1}{2}[\boldsymbol{x}(t) + \boldsymbol{x}(t - 2\Delta t)], t - \Delta t \right).$$

Rivest et al. compare weather simulations employing the great-circle extrapolation method with those obtained by using double trajectory calculations. (The piecewise trajectory method is not considered.) The extrapolation method exhibits slightly inferior accuracy, but has the advantage of computational speed.

11.1 Three-Time-Level Schemes

We shall now take the solution to the orographic resonance problem, described above, as a starting point for a fuller investigation of three time-level semi-implicit, semi-Lagrangian discretisations of (11.1). We have seen that off-centring the semi-implicit discretisation of G is an effective approach to removing spurious numerical oscillations. A centred in time, two level discretisation (Crank-Nicolson) has second order accuracy. However, off-centring a two level scheme will reduce its accuracy to first order. Second order may be regained by introducing a third time-level to the discretisation. A paper by Côté , Gravel and Staniforth [5] builds on the results of Rivest et al. by applying three time-level discretisations to both sides of (11.1). Their scheme has two parameters and takes the form

$$\frac{DF}{Dt} \rightarrow \frac{1}{\Delta t}\left[(1+\epsilon)F^n - (1+2\epsilon)F^{n-1} + \epsilon F^{n-2}\right], \tag{11.6}$$

$$G \rightarrow \left(\frac{1+\delta+\epsilon}{2}\right)G^n + \left(\frac{1-2\delta}{2}\right)G^{n-1} + \left(\frac{\delta-\epsilon}{2}\right)G^{n-2}. \tag{11.7}$$

In this scheme δ decentres the semi-implicit discretisation of G to control spurious oscillations. The second parameter, ϵ, specifies a three time-level semi-Lagrangian discretisation along trajectories, allowing reduction of temporal truncation errors.

To identify suitable ranges for the parameter values the Fourier analysis of Rivest et al. is repeated. Fourier analysis of the traveling wave solutions supported by the discrete equation set (11.6,11.7) reveals that a necessary condition for stability is

$$\delta > 0,$$

whenever $1 + 2\epsilon > 0$.

Fourier analysis of stationary modes shows they are non-resonant provided

$$1 + 2\epsilon > 0.$$

For negative values of ϵ, although the spurious stationary mode may be stable it will alternate sign with successive time-steps. A positive but small value of ϵ will damp the resonant mode and remove its temporal sign-switching.

Through a series of trial integrations, on a global shallow-water model, optimal values of ϵ and δ are deduced. A value of δ above 1/6 will sufficiently dampen resonance, but a value larger than 1/2 will lead to increased temporal truncation errors. It would appear that the choice of ϵ is less critical, except that it should be small and positive.

Summary

In this thesis we have examined recent developments in semi-Lagrangian schemes. The earliest uses of SL schemes in meteorological applications indicated deficiencies which have lately been addressed [55]. Among these we mention the computational cost of interpolation, the lack of formal conservation and the problem of orographic resonance.

Two approaches to reducing the computational cost of interpolation have been examined in this thesis. The noninterpolating methods discussed in Chapter 6 remove interpolation completely from SL schemes. Eulerian advection schemes are used in place of interpolation. The noninterpolating formalism therefore offers a framework for the development of many new schemes.

An alternative to more traditional forms of interpolation has been introduced by Purser and Lesley, as discussed in Chapter 5. This method significantly reduces the number of univariate interpolations required to produce a single multivariate interpolation, in comparison to standard tensor product methods. A further development of this method has incorporated conservation of mass. To date, it is not clear how this latter scheme may be implemented in spherical geometry. For this reason the method is not yet applicable to global atmospheric modelling.

Besides the issue of computational speed, further requirements of interpolation for SL schemes have been discussed. In Chapter 4 we saw how the introduction of non-physical errors may be avoided through the use of monotone, or shape-preserving interpolation. The QMSL algorithm offers a means of rendering any existing SL scheme monotone, through minimal extra computational effort. In addition, this algorithm has led to a

method for making SL schemes conservative, or nearly so. This method, QCSL, was described in Chapter 9, where we also discussed other approaches to making conservative SL schemes.

The most frequently used interpolation in semi-Lagrangian schemes is cubic. This is generally found to give the best compromise between accuracy and computational effort. In Chapter 3 we examined the largely neglected use of quadratic interpolation in SL schemes. A possible application where quadratic interpolation might be superior to cubic interpolation is examined in Chapter 7. This is in the modelling of nonlinear advection, which we approach by calculating SL solutions of the Burgers' equation. The results of this Chapter show that quadratic interpolation does have an advantage over cubic in this context. This finding is also replicated in idealized tests with a nonhydrostatic atmospheric model.

In Chapter 10 a comparison was made between a QCSL scheme and a second form of conservative scheme described in that chapter. Non-monotone schemes were also applied to the same test problem. The results indicated first the advantages of using a monotone scheme, and secondly the degree to which conservation may be lost in such a monotone scheme. The QCSL algorithm successfully retains the shape-preserving properties of the monotone scheme, while significantly improving conservation. In comparison, a more sophisticated approach to conservation attains conservation of mass to the significant detriment of the accuracy of the solution.

The development of SL schemes has taken place almost entirely within the context of atmospheric modelling [32]. Chapter 8 examined an approach to implementing SL on the sphere. It was seen that the semi-Lagrangian method may be applied fully to the linear, and nonlinear, advection terms of the shallow-water equations in spherical geometry. Also in this Chapter, we presented a formalism for deriving departure point schemes, for use in curved spaces. This approach is based on the method of geodesics and provides an alternative derivation of a technique currently used in numerical weather prediction. However, the generality of the geodesic method has the potential to extend

the range of possible applications of the SL method.

Lastly, in Chapter 11 we examined the problem of spurious resonance caused by orography. In early SL models, where mountain topography was included at the lower boundary, spurious standing wave patterns presented a particular problem. A sequence of papers has lately addressed this problem, demonstrating further the suitability of semi-Lagrangian methods in atmospheric modelling.

Future Developments

The semi-Lagrangian method is still an active area for development. Many new algorithms for the three components of SL (interpolation, trajectory tracing and time discretisation) may yet be possible. In addition to the development of new schemes, much work still remains to be done on the error analysis of SL schemes. In particular, analysis may show the optimum balance between the components of an SL scheme required to achieve a given level of predictive accuracy.

Bibliography

[1] H. Akima. A new method of interpolation and smooth curve fitting based on local procedures. *J. Assoc. Comput. Mach.*, 17:589–602, 1970.

[2] J.R. Bates, F.H.M. Semazzi, R.W. Higgins, and S.R.M. Barros. Integration of the shallow water equations on the sphere using a vector semi-Lagrangian scheme with a multigrid solver. *MWR*, 118:1615–1627, 1990.

[3] R. Bermejo and A. Staniforth. The conversion of semi-Lagrangian schemes to quasi-monotone schemes. *Monthly Weather Review*, 120:2622–2632, 1992.

[4] J.P. Boris and D.L. Book. Flux corrected transport, I, SHASTA, a fluid transport algorithm that works. *Journal of Computational Physics*, 11:38–69, 1973.

[5] J.C. Côté, S. Gravel, and A. Staniforth. A generalized family of schemes tht eliminate the spurious resonant response of semi-Lagrangian schemes to orographic forcing. *Monthly Weather Review*, 123:3605–3613, 1995.

[6] R. Courant, K.O. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *Math. Annalen*, 100:32–74, 1928. English translation by Phyllis Fox, IBM Jnl, March 1967.

[7] R.L. Dougherty, A. Edelman, and J.M. Hyman. Nonnegativity-, monotonicity-, or convexity-preserving cubic and quintic Hermite interpolation. *Mathematics of Computation*, 52:471–494, 1989.

[8] F.N. Fritsch and R.E. Carlson. Monotone piecewise cubic interpolation. *SIAM J. Numer. Anal.*, 17:238–246, 1980.

[9] S. Gravel and A. Staniforth. A mass-conserving semi-Lagrangian scheme for the shallow-water equations. *Monthly Weather Review*, 122:243–248, 1994.

[10] A. Harten. High resolution schemes for hyperbolic conservation laws. *Journal of Computational Physics*, 49:357–393, 1983.

[11] A. Harten, B. Engquist, S. Osher, and S.R. Chakravarthy. Uniformly high-order accurate essentially non-oscillatory schemes .3. *Journal of Computational Physics*, 71:231–303, 1987.

[12] A. Harten, S. Osher, B. Engquist, and S.R. Chakravarthy. Some results on uniformly high order accurate essentially non-oscillatory schemes. *Applied. Num. Math.*, 2:347–377, 1986.

[13] J.T. Houghton. *The Physics of Atmospheres*. Cambridge University Press, 1986.

[14] J.M. Hyman and B. Larrouturou. The numerical differentiation of discrete functions using polynomial interpolation methods. In J.F. Thompson, editor, *Numerical Grid Generation for Numerical Solution of Partial Differential Equations*, pages 487–506. Elsevier North-Holland, 1982.

[15] I. Jacques and C. Judd. *Numerical Analysis*. Chapman and Hall, 1987.

[16] L.W. Johnson and R.D. Riess. *Numerical Analysis*. Addison-Wesley, 1977.

[17] J. Douglas Jr. and T.F. Russell. Numerical methods for convection-dominated diffusion problems based on combining the method of characteristics with finite element or finite difference procedures. *SIAM J. Numer. Anal.*, 19:871–885, 1982.

[18] T. Kawai, editor. *Characteristics and the Finite Element Method*, number 4 in Int. Symp. on Finite Element Methods in Flow Problems. North-Holland, 1982.

[19] Hung-Chi Kuo and R.T. Williams. Semi-Lagrangian solutions to the inviscid Burgers' equation. *Monthly Weather Review*, 118:1278–1288, 1990.

[20] J.P.R. Laprise and A. Plante. A class of semi-Lagrangian integrated mass (SLIM) numerical transport algorithms. *Monthly Weather Review*, 123:553–565, 1995.

[21] P. Lax and B. Wendroff. Systems of conservation laws. *Communications on Pure and Applied Mathematics*, 13:217–237, 1960.

[22] B.P. Leonard. Positivity-preserving numerical schemes for multidimensional advection. Technical Report ICOMP-93-05, Institute for Computational Mechanics in Propulsion, 1993.

[23] L.M. Leslie and R.J. Purser. Three-dimensional mass-conserving semi-Lagrangian scheme employing forward trajectories. *Monthly Weather Review*, 123:2551–2566, 1995.

[24] R.J. LeVeque. *Numerical Methods for Conservation Laws*. Birkhäuser Verlag, 1992.

[25] E. Lorenz. Climate Predictability. GARP Publication Series No. 16, World Meteorological Organisation, Geneva, 1975.

[26] J.E. Lovelock and L.R. Kump. Failure of climate regulation in a geophysiological model. *Nature*, 369:732–734, 1994.

[27] J. Maddox. Clouds and global warming. *Nature*, 247:329, 1990.

[28] P.A. Makar and S.R. Karpik. Basis-spline interpolation on the sphere : Applications to semi-Lagrangian advection. *Monthly Weather Review*, 124:182–199, 1996.

[29] A. McDonald and J.R. Bates. Semi-Lagrangian integration of a gridpoint shallow water model on the sphere. *Monthly Weather Review*, 117:130–137, 1989.

[30] J.L. McGregor. Economical determination of departure points for semi-Lagrangian models. *Monthly Weather Review*, 121:221–230, 1993.

[31] K.W. Morton and D.F. Mayers. *Numerical Solution of Partial Differential Equations*. Cambridge University Press, 1994.

[32] P. Garcia Navarro and A. Priestley. A conservative and shape-preserving semi-Lagrangian method for the solution of the shallow-water equations. *International Journal for Numerical Methods in Fluids*, 18:273–294, 1994.

[33] G. Nicolis. *Introduction to Nonlinear Science*. Cambridge University Press, 1995.

[34] M. Olim. A truly noninterpolating semi-Lagrangian Lax-Wendroff method. *Journal of Computational Physics*, 112:253–266, 1994.

[35] A. Plante. *Transport de Substances par Schémas Numériques Semi-Lagrangiens Intégrés par Cellules*. PhD thesis, Physics Department, University of Québec at Montréal, 1993.

[36] P.M. Prenter. *Spline and Variational Methods*. 1975.

[37] A. Priestley. Private communication.

[38] A. Priestley. A quasi-conservative version of the semi-Lagrangian advection scheme. *Monthly Weather Review*, 121:621–629, 1993.

[39] R.J. Purser and L.M. Leslie. An efficient interpolation procedure for high-order three-dimensional semi-Lagrangian models. *Monthly Weather Review*, pages 2492–2498, 1991.

[40] R.J. Purser and L.M. Leslie. An efficient semi-Lagrangian scheme using third-order semi-implicit time integration and forward trajectories. *Monthly Weather Review*, 122:745–756, 1994.

[41] C.S. Ramage. Forecasting in meteorology. *Bulletin of the American Meteorological Society*, 74(10):1863,1871, 1993.

[42] M. Rančić. An efficient conservative, monotonic remapping for semi-Lagrangian transport algorithms. *Monthly Weather Review*, 123:1213–1217, 1995.

[43] P.J. Rasch and D.L. Williamson. On shape-preserving interpolation and semi-Lagrangian transport. *SIAM J. Sci. Stat. Comp.*, 11, 1990.

[44] H. Ritchie. Eliminating the interpolation associated with the semi-Lagrangian scheme. *Monthly Weather Review*, 114:135–146, 1986.

[45] H. Ritchie and C. Beaudoin. Approximations and sensitivity experiments with a baroclinic semi-Lagrangian spectral model. *Monthly Weather Review*, 122:2391–2399, 1994.

[46] C. Rivest, A. Staniforth, and A. Robert. Spurious resonant response of semi-Lagrangian discretizations to orographic forcing : Diagnosis and solution. *Monthly Weather Review*, 122:366–376, 1994.

[47] J.M. Sanz-Serna and M.P. Calvo. *Numerical Hamiltonian Problems*. Chapman & Hall, London, 1994.

[48] B.F. Schutz. *Geometrical Methods of Mathematical Physics*. Cambridge University Press, 1980.

[49] B.F. Schutz. *A First Course in General Relativity*. Cambridge University Press, 1985.

[50] P.K. Smolarkiewicz. The multi-dimensional Crowley advection scheme. *Monthly Weather Review*, 113:1050–1065, 1982.

[51] P.K. Smolarkiewicz. Reply. *Monthly Weather Review*, 115:901–902, 1987. in notes and correspondence.

[52] P.K. Smolarkiewicz and G.A. Grell. A class of monotone interpolation schemes. *Journal of Computational Physics*, 101:431–440, 1992.

[53] P.K. Smolarkiewicz and J.A. Pudykiewicz. A class of semi-Lagrangian approximations for fluids. *Journal of the Atmospheric Sciences*, 49:2082–2096, 1992.

[54] P.K. Smolarkiewicz and P.J. Rasch. Monotone advection on the sphere : An Eulerian versus semi-Lagrangian approach. *Journal of the Atmospheric Sciences*, 48:793–810, 1991.

[55] A. Staniforth and J. Côté. Semi-Lagrangian integration schemes for atmospheric models. *Monthly Weather Review*, 119:2206–2223, 1991.

[56] A. Staniforth, J. Côté, and J. Pudykiewicz. Comments on "Smolarkiewicz's deformational flow". *Monthly Weather Review*, 115:894–900, 1987.

[57] P.K. Sweby. High-resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM J. Numerical Analysis*, 121:995–1011, 1984.

[58] M.C. Tapp and P.W. White. A non-hydrostatic mesoscale model. *Quarterly Journal of the Royal Meteorological Society*, 102:277–296, 1976.

[59] C. Temperton and A. Staniforth. An efficient two-time-level semi-Lagrangian semi-implicit integration scheme. *Quarterly Journal of the Royal Meteorological Society*, 113:1025–1039, 1987.

[60] D.L. Williamson, J.B. Drake, J.J. Hack, R. Jakob, and P.N. Swarztrauber. A standard test set for numerical approximations to the shallow-water equations in spherical geometry. *Journal of Computational Physics*, 102:211–224, 1992.

[61] D.L. Williamson and P.J. Rasch. Two-dimensional semi-Lagrangian transport with shape-preserving interpolation. *Monthly Weather Review*, 117:102–129, 1989.

[62] S.T. Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of Computational Physics*, 31:335–362, 1979.