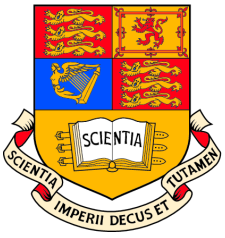
Modelling multiple-material flows on adaptive unstructured meshes

Cian Wilson

Department of Earth Science and Engineering
Imperial College London



Submitted for the degree of Doctor of Philosophy
and Diploma of Imperial College

September 2009

Modelling multiple-material flows on adaptive unstructured meshes

Cian Wilson
Department of Earth Science and Engineering, Imperial College London
September 2009

The ability to distinguish between regions with different material properties is essential when numerically modelling many physical systems. Using a dual control volume mesh that avoids the problem of corner coupling, the HyperC face value scheme is extended to multiple dimensions and applied as a device for material advection on unstructured simplex meshes. The new scheme performs well at maintaining sharp interfaces between materials and is shown to produce small advection errors, comparable to those of standard material advection methods on structured meshes. To further minimise numerical diffusion of material interfaces a total variation bounded flux limiter, UltraC, is defined using a normalised variable diagram.

Combining the material tracking scheme with dynamically adapting meshes, the use of a minimally dissipative bounded projection algorithm for interpolating fields from the old mesh to the new, optimised mesh is demonstrated that conserves the mass of each material. More generally, material conservation during the advection process is ensured through the coupling of the material tracking scheme to the momentum and mass equations. A new element pair for the discretisation of velocity and pressure is proposed that maintains the stability of the system while conserving the mass of materials.

When modelling multiple materials the use of independent advection algorithms for each material can lead to the problem of non-physical material overlap. A novel coupled flux limiter is developed to overcome this problem. This automatically generalises to arbitrary numbers of materials. Using the fully coupled (and rigorously verified) multi-material model, several geophysically relevant simulations are presented examining the generation of waves by landslides. The model is validated by demonstrating close agreement between model predictions and experimental results of wave generation, propagation and run-up. The simulations also showcase the powerful capabilities of an unstructured, adaptive multi-material model in real world scenarios.

Contents

Abstract i

Contents ii

Notation iv

1 Introduction 1
1.1 Modelling Multiple-Material Geophysical Flows 1
1.2 Unstructured Adaptive Meshes for Modelling Multi-Material Flows 3
1.3 Statement of Achievement and Contribution and Structure of Thesis . . . 5

2 Methods 7
2.1 Fundamental Equations 7
2.2 Discretisation 10
2.2.1 Finite Element Method 11
2.2.2 Finite Volume Method 19
2.2.3 Numerical Solutions 21
2.3 Mesh Optimisation 23
2.4 Interpolation 28
2.5 Summary 32

3 Material Tracking 34
3.1 Field Advection 36
3.2 One-Dimensional Limiting 37
3.2.1 Sensor Functions: Detecting Non-Physical Oscillations 39
3.2.2 Face Value Functions: Defining Boundedness 40
3.2.3 Limiters: Correcting the Surface Fluxes 44
3.3 Multiple Dimensions 56
3.3.1 Control Volume Dual Mesh 59
3.3.2 Material Tracking 70
3.4 Adjusting the Boundedness Criteria: A New Compressive Advection Method 95
3.4.1 Multi-Dimensional Analysis 108
3.5 Mesh Adaptivity 116
3.6 Summary 127

4 Flow Model **131**
4.1 Projection Method . . . 132
4.2 Stability . . . 138
4.2.1 Convergence Analysis . . . 143
4.3 Material Conservation . . . 156
4.4 Experimental Validation . . . 169
4.4.1 Two-Dimensional Water Column Collapse . . . 169
4.4.2 Three-Dimensional Water Column Collapse . . . 178
4.5 Summary . . . 187

5 Multiple Materials **188**
5.1 Coupled Flux Limiting . . . 189
5.1.1 Multi-Dimensional Advection Tests . . . 201
5.2 Landslide Generated Waves . . . 217
5.2.1 Wave Run-up on a Sloping Coast . . . 218
5.2.2 Lituya Bay Landslide . . . 222
5.3 Summary . . . 241

6 Conclusions **242**

Acknowledgements **248**

Bibliography **250**

A Model Code **262**

B Advection Tests **273**
B.1 Two Material Solid Body Rotation . . . 273
B.2 Two Material Shearing Flow . . . 273
B.3 Five Material Shearing Flow . . . 277

Notation

x_i	i^{th} component of space
$\cdot_{,i}$	derivative with respect to the i^{th} component of space
t	time
$\cdot_{,t}$	derivative with respect to time
E	error
c_p	order of convergence
u_i	i^{th} velocity component
ρ	density
μ	viscosity
g_i	i^{th} gravity component
u_{0i}	i^{th} velocity component initial condition
p	pressure
u_i^e	exact i^{th} velocity component
p^e	exact pressure
Ω	domain
$\partial\Omega$	domain boundary
$\partial\Omega_{g_i}$	domain boundary with Dirichlet velocity boundary condition
d_i	i^{th} velocity component Dirichlet boundary condition
$\partial\Omega_h$	domain boundary with Neumann velocity boundary condition
$\int_\Omega dV$	body integral over domain, Ω
$\int_{\partial\Omega} ds$	surface integral over domain boundary, $\partial\Omega$
Ω_e	element of domain tessellation
$\partial\Omega_e$	element boundary
N_A	shape function (velocity) for node A
M_A	shape function (pressure) for node A
Ω_v	volume of domain tessellation
$\partial\Omega_v$	volume boundary
n_i	i^{th} component of unit normal to surface
h	mesh spacing
Δt	timestep
$\cdot^n$	variable evaluated at n^{th} time level
θ	implicit (1) or explicit (0) temporal discretisation
θ_i	temporal discretisation of nonlinear momentum terms
θ_p	temporal discretisation of 'low-order' pivot advection terms

$\tilde{u}$	nonlinear velocity
u_i^*	best estimate of i^{th} component of velocity at the next time level
p^*	best estimate of pressure at the next time level
$\mathbf{M}$	mass matrix
$\mathbf{M}^L$	lumped mass matrix
$\mathbf{C}_i$	i^{th} component of pressure gradient matrix
$\mathbf{C}_i^T\mathbf{M}^{-1}\mathbf{C}_i$	pressure projection matrix
P1P1	piecewise linear velocity–linear pressure discretisation
P2P1	piecewise quadratic velocity–linear pressure discretisation
P0P1	piecewise constant velocity–linear pressure discretisation
$\text{P1}_{\text{DG}}\text{P2}$	piecewise linear discontinuous velocity–linear pressure discretisation
P0P1_{CV}	piecewise constant velocity–control volume pressure discretisation
α	volume fraction field
α^k	volume fraction for material k
ρ^k	density of material k
$\alpha^{\sum_I}$	sum of the first I volume fractions
α^e	exact volume fraction field
α_0	initial volume fraction field
f_k	face k in the control volume dual mesh
f_{u_k}	face upwind to the face f_k in the control volume dual mesh
$\alpha_{c_{f_k}}$	donor cell value to the face f_k
$\alpha_{d_{f_k}}$	downwind cell value to the face f_k
$\alpha_{u_{f_k}}$	upwind cell value to the face f_k
α_{f_k}	face value at the face f_k
$\alpha\vert_{f_k}^{LO}$	'low-order' face value at the face f_k
$\alpha\vert_{f_k}^{HO}$	'high-order' face value at the face f_k
$\hat{\alpha}_{f_k}$	limited face value at the face f_k
$\tilde{\alpha}$	best estimate of volume fraction at the next time level
$c_{c_{f_k}}$	Courant number for the donor cell to the face f_k
I_{f_k}	flow incoming (1) or outgoing (0) across the face f_k
$\alpha_{td_{f_k}}$	target downwind value to the face f_k
$\alpha_{tu_{f_k}}$	target upwind value to the face f_k
$\bar{\alpha}_{c_{f_k}}$	normalised donor value to the face f_k
$\bar{\alpha}_{f_k}$	normalised face value at the face f_k
$\bar{\hat{\alpha}}_{f_k}$	normalised limited face value at the face f_k
$\check{\alpha}_{c_{f_k}}$	modified normalised donor value to the face f_k
$\check{\alpha}_{f_k}$	modified normalised face value at the face f_k
$\alpha_{\max}$	maximum bound on volume fraction field
$\alpha_{\min}$	minimum bound on volume fraction field
α_{dev}	deviation from boundedness of the volume fraction field
γ	scaling parameter for duration of metric advection

Chapter 1

Introduction

1.1 Modelling Multiple-Material Geophysical Flows

Numerical models are an important tool in the scientific study of many physical phenomena. They are particularly important in the study of geophysical flows that occur on scales that greatly exceed those reproducible in an experimental setting. Numerical modelling may be used to extend data collected from analogue models and other tools to more physical scales. For example, circulation patterns in gases and liquids may be reproduced in small-scale laboratory cells but only numerical modelling can consider planetary-scale flows occuring in the atmosphere, oceans and mantle. Numerical modelling on larger scales also allows different physical regimes to be considered. For example, the behaviour of large rock masses during impact (Melosh, 1989) and avalanching (Collins & Melosh, 2003) is observed to change at a gravity dependent size well beyond the laboratory scale, which is only reproducible numerically. Numerical modelling may also be used to incrementally increase the complexity of a problem. While processes like landslides and tsunamis may be investigated at the laboratory scale, they are highly dependent on bathymetry and topography (Fine *et al.*, 2005; Ward, 2001), which can be more easily adjusted in a numerical rather than an analogue model. Numerical models may also be used to supplement field based data. In a geologic setting the latter tends to only be available at scattered points and modelling is an essential tool to aid interpretation.

Heterogeneities in composition and other physical properties play an important role in many geophysical flows. Numerical models should therefore be capable of simulating more than one material. In some cases the different materials behave passively, simply being transported by the surrounding material. For example, chemical heterogeneities in the mantle may be advected by the thermally-driven overall circulation pattern, yet maintain material properties distinct from the bulk composition. In other cases the material differences drive the flow. For example, density differences between air masses cause atmospheric currents (Yih, 1965). Similarly, impacts between chemically distinct solar system bodies lead to the generation of impact craters while, terrestrially, the impact of landslides into water generates waves.

All of the processes listed above are of great scientific interest and modelling provides significant insights into their behaviour. However, numerically modelling distinct, im-

miscible materials is challenging. Most numerical models consider the behaviour of the physical world as a series of continuous equations. These are approximated or discretised at a finite number of points in space and time, forming a mesh or grid. In Lagrangian models, the mesh follows the movement of material, deforming with it. In this case, multiple materials and the interfaces between them are handled naturally. During highly dynamic fluid processes, however, mesh tangling, as grid points pass each other and overlap, makes the Lagrangian approach unsuitable. Eulerian numerical schemes, where material flows through the mesh, permit simulation of highly dynamic phenomena where large deformations are common; however, transporting multiple materials through a mesh is difficult. If the change in material properties occurs over a length scale less than or equal to the distance between these points then the materials should be considered distinct. In effect they are separated by an interface. As the grid spacing in a numerical model is generally larger than the physical scale on which the material properties change, this interface is numerically poorly defined. Each grid point represents an average of the properties in the region between it and the neighbouring point. In the vicinity of an interface this leads to some grid points representing non-physical mixtures. This problem is exacerbated when attempting to move the material through the mesh. Such transport schemes inherently introduce diffusion at the interface leading to further non-physical mixing.

The amount of non-physical mixing is a function of the distance between the grid points. Increasing the number of points decreases the fraction of the domain that each point represents and hence, decreases the amount of numerical mixing between the materials. In essence, increasing the resolution of the model improves the approximation to the real world interface. However, the resolution of the discretisation strongly influences the computational requirements of a simulation. Increasing the number of grid points increases the required computer storage and simulation time, as data must be stored and solutions must be found at more locations. This problem becomes particularly acute in higher dimensions, as the number of points must be spread over a larger domain.

The boundary between two distinct materials is generally the smallest scale feature that must be resolved in a simulation. This means that the grid spacing will be defined by how well the interface needs to be described and must be small enough to achieve the desired level of accuracy in the solution. At the same time, an interface generally takes up only a small fraction of the domain. Therefore, the resolution required to describe it may not be necessary to approximate the internal dynamics of the materials themselves to the same level of accuracy. The computational cost of a simulation may be decreased by reducing the resolution or increasing the grid point spacing away from any interfaces. As a result, the geometry and structure of the computational mesh can have a strong influence on the efficiency and accuracy of modelling multiple material flows.

1.2 Unstructured Adaptive Meshes for Modelling Multi-Material Flows

Structured meshes are chessboard-like, consisting of equally spaced grid points in all directions. They are the easiest way of discretising simple box-like domains and are defined such that all points, except those on the domain boundaries, are linked to their neighbours using a standard formula. If exploited correctly this formula allows each point to find its neighbours in a computationally inexpensive manner. However, it also means that a section of mesh in one part of the domain will look almost identical to that in any other part of the domain.

If a structured mesh is used then the neighbourhood formula must be valid everywhere. This means that variations in resolution are difficult to support. Decreasing the grid spacing in the vicinity of an interface will generally involve increasing the resolution elsewhere as well, where it may not be needed (see Figure 1.1(a)). Such superfluous grid points mean that the full benefits of varying the resolution cannot be achieved, as the computational resources required do not decrease significantly. This additional cost may outweigh the potential advantages of programming on a structured mesh.

On unstructured meshes the relationship between grid points is not governed by a standard formula across the entire domain. In other words, a grid point can be arbitrarily connected to its neighbours. While the lack of a neighbourhood formula means there is an increased cost in determining the relationship between grid points, it also means that the resolution is free to vary rapidly from one region to the next, without imposing additional superfluous resolution in neighbouring areas. The additional cost of arbitrary grid point relationships may therefore be outweighed by the reduction in computational resources required to accurately represent the interface.

The rate at which the resolution can vary on an unstructured mesh is dependent on its isotropy. Isotropic and near-isotropic meshes enforce approximately the same grid point spacing in all directions. An interface, however, is highly anisotropic as material properties vary only in specific directions. The best grid point spacing would be smallest across a material boundary but an isotropic mesh will impose this resolution along the length of the interface as well. This increases the resources required for the simulation (see Figure 1.1(b)).

Allowing the resolution to vary in different directions means that the grid point spacing can be at its smallest across an interface but much larger along its length (see Figure 1.1(c)). When such small scale features occupy only a limited portion of the domain, such unstructured, anisotropic meshes can represent distinct materials with the minimum number of grid points necessary. This allows the computer resource requirements to be reduced and solutions can be found faster to the same degree of accuracy as on structured meshes.

Fixed meshes remain the same throughout a simulation. In Figure 1.1, if the meshes are fixed in time and the position of the interface moves during the simulation, then the solution will not be optimally represented. This arises because the meshes were designed, within certain constraints, to represent only the current position of the two materials with

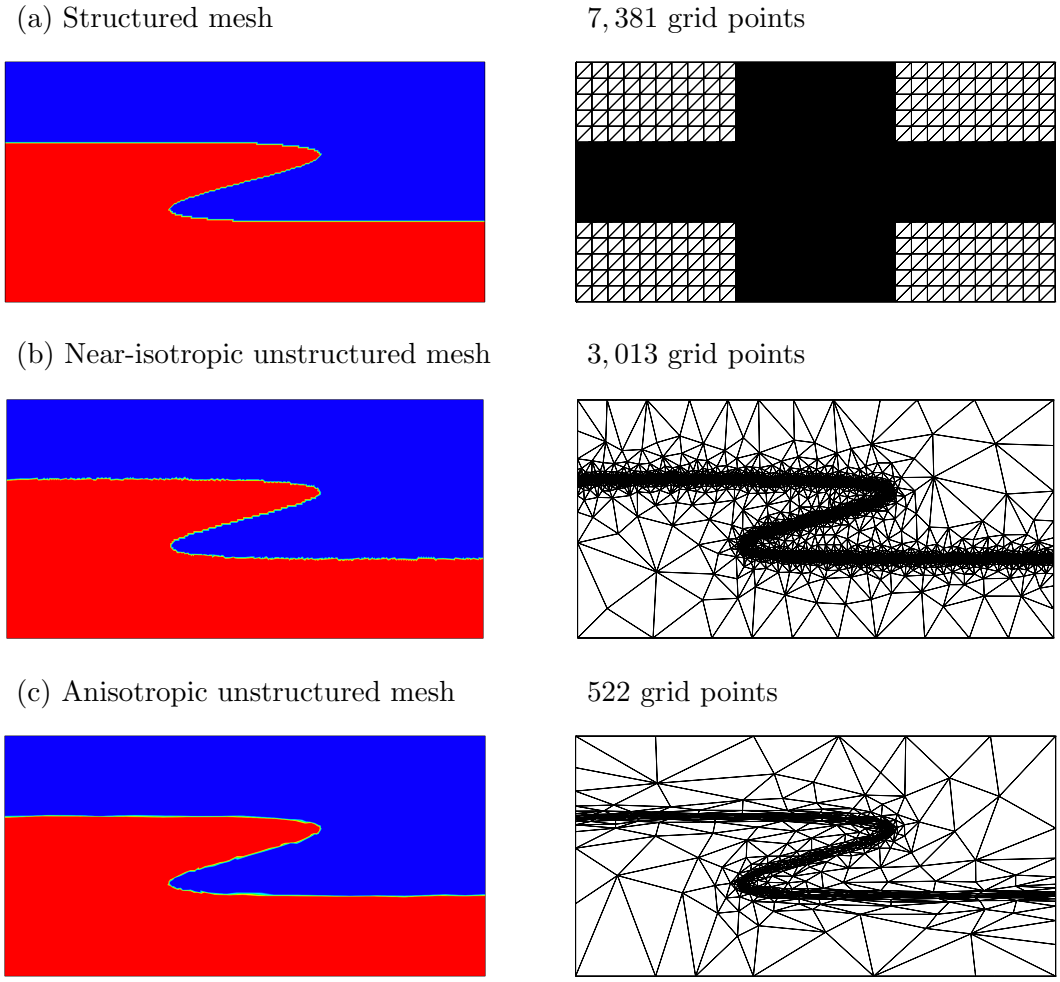


Figure 1.1: A rapidly varying field represented on different meshes. All meshes have the same minimum distance between grid points so that the mixed area around the interface remains approximately the same width. Relaxing various constraints leads to a large variation in the number of grid points required, lowering the computational cost of the mesh. (a) A structured mesh, even with variable resolution, requires large numbers of nodes away from the interface in order to maintain a constant local stencil. This leads to $7,381$ grid points over the domain. (b) An unstructured mesh substantially reduces this number to $3,013$ by allowing the resolution to vary more rapidly away from the interface. An aspect ratio bound on the cells prevents the resolution in the highly resolved region from varying along and across the interface. (c) Removing this requirement means that an anisotropic unstructured mesh can represent the interface with only 522 grid points.

the fewest grid points possible. Therefore, if the positions of the materials change, the mesh is no longer suitable. If a fixed mesh is used then it is again necessary to increase the resolution in areas away from the interface in order to pre-empt its possible movement. Alternatively, a refined mesh may be used so long as it is periodically adjusted to ensure it is optimised for the current state of the materials. Such an *adaptive* mesh allows the resolution to be focussed around interfaces throughout the simulation. This can be an extremely efficient approach, as only the number of grid points required at the current time are ever used and the number of points may increase or decrease as necessary to achieve a certain accuracy.

Unstructured, anisotropic, adaptive meshes offer a method of efficiently modelling multi-scale processes, which can be represented in different regions by varying levels of resolution. Furthermore, such meshes reduce the cost of increasing the dimensionality of the problems considered, as the grid point scaling that applied to structured meshes is no longer relevant. The principal factor influencing the number of grid points on an unstructured mesh is the degree of accuracy desired and the complexity of the feature being represented. However, the issue of interface smearing during transport remains, even on an optimised mesh. Standard methods of overcoming this involve explicitly reconstructing the interface between the materials. Using the position of this interface to guide the advection of materials between grid points minimises the numerical diffusion inherent in this process.

Interface reconstruction is computationally challenging and expensive. Generally, geophysical flow models that use it have been implemented on structured fixed meshes. This considerably simplifies the construction as only a single set of possible relationships exist between the interface and the mesh around a grid point. For arbitrarily connected meshes the number of possible interface intersections with the mesh increases and, as a result, so does the computational cost. Periodically changing the mesh, to optimally represent the current solution, further increases the cost of interface reconstruction. After each step of mesh optimisation a new set of potential intersections must be considered. To avoid these costs, this thesis investigates novel methods of advecting immiscible materials on unstructured, anisotropic, adaptive meshes, that allow realistic geophysical flows to be simulated.

1.3 Statement of Achievement and Contribution and Structure of Thesis

In this thesis I describe a new method for simulating multi-material fluid flow on unstructured adaptive simplex meshes. Chapter 2 contains a brief overview of the equations that are to be solved and the simplifications made to them. Standard methods of discretising these equations on unstructured meshes are then discussed before considering their numerical assembly and solution. An overview is also given of the computational libraries used to perform this assembly and to optimise the mesh. The purpose of this chapter is to provide the reader with a concise summary of important background material for

later chapters.

While Chapter 2 reviews standard methods, Chapters 3–5 describe the results of my own research, except where other sources are cited. In particular, in Sections 3.3–3.5 I describe new methods of transporting materials in multiple dimensions and demonstrate their use on unstructured and adaptive meshes. The results in Section 3.5 were published in Farrell *et al.* (2009) but were not discussed in the context of modelling multiple materials. A further paper from Chapter 3 is in preparation, where the applicability of these new methods to material tracking will be discussed in comparison to some standard structured methods.

In Chapter 4 I show how the advection algorithms developed in Chapter 3 are coupled to a flow model, which describes the velocity throughout the simulation. This allows the previously passive materials to influence the dynamics through their properties, like density and viscosity. In essence, I describe in Chapter 4 a geophysical two-material flow model and demonstrate its ability to reproduce real world experiments through several example problems. A paper is in preparation describing the discretisation selected to couple the velocity and material fields together and demonstrating its capabilities.

In Chapter 5 I broaden the range of scenarios that can be considered by extending the model to incorporate arbitrary numbers of materials. Here I describe the development of a novel coupling mechanism between the materials that creates a truly multiple material geophysical flow model. In addition I demonstrate the model's successful application to real world geophysical flow problems by simulating the generation of a tsunami by the impact of a landslide with water and the subsequent run-up of the wave against a shoreline. Two papers are in preparation based on this work. One detailing the material coupling method and the other describing the model's application to landslide generated waves.

Finally, in Chapter 6 I summarise the main developments of the thesis and discuss possible future developments and applications of the model. These include investigating other geophysical flow problems, such as thermo-chemical mantle convection, and the extension of the work on landslide generated waves to more complicated scenarios. The main conclusions and findings of the work are also summarised.

Throughout this project, I incorporated the developments described in this thesis into the code base of Fluidity. Fluidity is an open source, fluid dynamics model that uses adaptive, unstructured, anisotropic meshes to solve the Navier-Stokes and related field equations. It has been developed principally by the Applied Modelling and Computation Group at Imperial College London. During this project I have made substantial modifications and contributions to Fluidity (see Ham *et al.* (2009), for example). My implementations, being specific to Fluidity, are not discussed at length; however, Appendix A gives details on how to access the code base and summarises the principal files I have written, contributed to or modified significantly.

Chapter 2

Methods

This chapter reviews some of the fundamental methods used throughout this thesis. The basic equations used in geophysical flow models are introduced in Section 2.1. These are derived from the basic principals of conservation of momentum, mass and energy which govern many physical systems. For the purposes of demonstrating material advection methods these equations are then simplified so that only incompressible, isothermal flows are considered.

These equations must then be represented at a finite number of points so that their solutions can be approximated numerically. This discretisation process must allow the use of unstructured meshes to describe the distribution of these points. Two related discretisation schemes that satisfy this requirement, the finite element and finite volume methods, are demonstrated on the principal equations in Section 2.2. The methods and libraries used to assemble and solve these discretised systems are then briefly discussed.

Finally the method used to define the optimal mesh for a given set of fields is described in Section 2.3. These methods, already incorporated into Fluidity, allow the mesh to be repeatedly improved as the computed solutions to the equations change.

2.1 Fundamental Equations

Time-dependent geophysical flow models are generally based on the Navier-Stokes equations (2.1), which describe the dynamics of continuous media. In an inertial frame of reference of dimension d these are the conservation of linear (Equation 2.1a) and angular (Equation 2.1b) momentum, mass (Equation 2.1c) and energy (Equation 2.1d):

$$(\rho u_i)_{,t} + (\rho u_i u_j)_{,j} - \sigma_{ij,j} = f_i \tag{2.1a}$$

$$\sigma_{ij} = \sigma_{ji} \tag{2.1b}$$

$$\rho_{,t} + (\rho u_i)_{,i} = 0 \tag{2.1c}$$

$$\rho \iota_{,t} + \rho u_i \iota_{,i} = \sigma_{ij} \dot{\epsilon}_{ij} \tag{2.1d}$$

in $\Omega \times [0, T]$, $i, j = 1, \ldots, d$, where ρ is the density, u_i is the i^{th} component of velocity, σ_{ij} is the Cauchy stress tensor, $f_i = \rho g_i + S_i$ is the i^{th} component of the body force vector (including a buoyancy term, ρg_i, where g_i is the gravity vector, and a momentum source, S_i), ι is the specific internal energy and $\dot{\epsilon}_{ij}$ is the strain rate tensor. Subscripted commas

indicate differentiation with respect to the i^{th} component of space, $\cdot_{,i}$, or time, $\cdot_{,t}$, and summation is assumed over repeated indices. A dynamic solution is sought through time, $0 \leq t \leq T$, in the bounded domain $\Omega \in \mathbb{R}^d$, with boundary $\partial\Omega$.

Some of these variables are further related to each other through two auxiliary equations: the constitutive or strength model (Equation 2.2) and the equation of state (Equation 2.3):

$$\sigma_{ij} = \varsigma(\dot{\epsilon}_{ij}, \dots) \tag{2.2}$$

$$\rho = \rho(\iota, \dots). \tag{2.3}$$

For the purpose of demonstrating material advection methods on unstructured meshes it is sufficient to simplify Equations 2.1. Firstly, Equation 2.1a can be rewritten:

$$\rho u_{i,t} + \rho \left(u_i u_j\right)_{,j} - \rho u_i u_{j,j} - \sigma_{ij,j} = f_i$$

by extracting the conservation of mass equation (2.1c) and rewriting the advection term:

$$u_j u_{i,j} = \left(u_i u_j\right)_{,j} - u_i u_{j,j}.$$

Furthermore, the stress term, σ_{ij}, can be split into isotropic, $p\delta_{ij}$, and non-isotropic, τ_{ij}, parts:

$$\sigma_{ij} = -p\delta_{ij} + \tau_{ij}. \tag{2.4}$$

Here p $(= \sigma_{ii}/d)$ is the mechanical pressure, representing the stress when the medium is at rest, while τ_{ij} is the deviatoric stress tensor, which arises due to motion. While these steps do not restrict the physical regimes that can be modelled using Equations 2.1, Chapter 4 will show that they considerably simplify the numerical implementation of their discrete counterparts. To further simplify the system, however, it is necessary to limit the scenarios which can be simulated.

This process is started by only considering Newtonian fluid constitutive models (Equation 2.2) where the deviatoric stress is linearly proportional to the gradient of velocity:

$$\tau_{ij} = c_{ijkl} u_{k,l},$$

and c_{ijkl} is a tensor containing the constants of proportionality. This is the most general Newtonian constitutive model. Most fluids, however, behave at a molecular level in a similar manner in all directions (Batchelor, 2000). This isotropy, combined with the symmetry properties of the conservation of angular momentum (Equation 2.1b) means that the generalised Newtonian fluid model becomes:

$$\tau_{ij} = 2\mu \left(u_{(i,j)} - \frac{1}{3}\delta_{ij} u_{k,k} \right) \tag{2.5}$$

where μ is the molecular viscosity and $u_{(i,j)}$ is the symmetric part of the velocity gradient tensor $(= \left(u_{i,j} + u_{j,i}\right)/2)$ which is the definition of the strain rate tensor $(= \dot{\epsilon}_{ij})$.

The remaining auxiliary equation (2.3) can also be simplified by considering an incompressible isothermal fluid of constant density, ρ. In which case the conservation of mass constraint (Equation 2.1c) reduces to:

$$u_{i,i} = 0, \tag{2.6}$$

so that the velocity field has zero divergence. Combining Equations 2.4, 2.5 and 2.6, the stress term in the momentum equation becomes:

$$\sigma_{ij,j} = -p_{,i} + \mu u_{i,jj}.$$

Hence, for an isotropic, isothermal Newtonian fluid Equations 2.1 become:

$$\rho u_{i,t} + \rho \left(u_i u_j\right)_{,j} - \rho u_i u_{j,j} - \mu u_{i,jj} = -p_{,i} + f_i, \tag{2.7a}$$

$$u_{i,i} = 0. \tag{2.7b}$$

In addition the following boundary and initial conditions are required:

$$u_i = d_i(x_j, t) \text{ on } \partial\Omega_{g_i} \times [0, T] \tag{2.8}$$

$$\mu u_{i,j} n_j = h_i(x_k, t) \text{ on } \partial\Omega_h \times [0, T] \tag{2.9}$$

$$u_i(x_j, 0) = u_{0i}(x_j) \quad \text{in } \Omega \tag{2.10}$$

where $(\bigcup_i \partial\Omega_{g_i}) \bigcap \partial\Omega_h = \emptyset$, $(\bigcup_i \partial\Omega_{g_i}) \bigcup \partial\Omega_h = \partial\Omega$ and n_j is the unit normal vector to $\partial\Omega$.

Equation 2.8 describes the imposition of a specified velocity, d_i, on some subset of the domain boundary, $\partial\Omega_{g_i}$. This often takes the form of a no slip boundary condition, where $d_i = 0 \,\forall\, i$, preventing flow at the boundary in all directions. Similarly, a no normal flow boundary condition specifies that $d_i n_i = 0$, preventing flow into or out of the domain.

Through the constitutive law (Equation 2.5) Equation 2.9 corresponds to the imposition of a stress normal to the boundary. This effectively means that the gradient of velocity normal to the boundary can be specified, though only the homogeneous case of zero normal stress, $h_i = 0$, is considered in this project.

Finally, Equation 2.10 describes the initial state of the velocity field at $t = 0$, u_{0i}. In general this should be in agreement with the boundary conditions and in its simplest form takes the form of a stationary initial condition, $u_{0i} = 0 \,\forall\, i$.

Equations 2.7 combined with suitable Dirichlet (Equation 2.8) and Neumann (Equation 2.9) boundary conditions and an initial condition (Equation 2.10) provide a well posed system for the velocity field. However, additional equations are required in order to track different fields around the domain subject to this velocity.

For some arbitrary field, α, a linear advection equation:

$$\alpha_{,t} + u_i \alpha_{,i} = 0, \tag{2.11}$$

is used to transport the field subject to velocity u_i and the boundary and initial conditions:

$$\alpha = \alpha_{\text{in}}(x_j, t) \text{ on } \partial\Omega_{\text{in}} \times [0, T], \tag{2.12}$$

$$\alpha(x_j, 0) = \alpha_0(x_j) \quad \text{in } \Omega, \tag{2.13}$$

where $\partial\Omega_{\text{in}} \subset \partial\Omega$ represents any inflowing boundary where $u_i n_i|_{\partial\Omega_{\text{in}}} < 0$. Effectively Equation 2.12 tells Equation 2.11 what information to propagate into the domain when

it would otherwise be undefined at the boundary. Similarly Equation 2.13 provides the initial distribution of the field prior to advection.

Despite all these simplifications analytical solutions to the coupled system of Equations 2.7–2.13 are not generally available, except in very simple scenarios. However, the complexity can be reduced further by approximating their solutions at a finite number of points in space and time. Such numerically discretised equations provide an approximation to the true behaviour of the system, which, given a suitable numerical method, converges towards the correct answer as the number of points increases.

2.2 Discretisation

The discretisation process begins by assuming that a finite set of n_p points is distributed arbitrarily throughout the domain, Ω. These points are all connected to each other in a simple manner so that no lines connecting two points cross each other. They therefore form vertices in a tessellation which fills the spatial domain with no gaps or overlaps.

In a dynamic simulation the solution can vary throughout the time domain, $0 \leq t \leq T$. As for the discretisation of space this is also divided into a finite set of discrete time levels. However, whereas the spatial position of a vertex is arbitrary, all nodes are assumed to be discretised at the same finite set of n_t times.

Numerous schemes exist for the numerical discretisation of partial differential equations. The most common methods that use meshes can be broadly categorised into three groups:

- finite difference methods, which approximate derivatives by the divided difference between grid points;
- finite and control volume methods, which consider fluxes between discrete cells; and
- finite element methods, which reduce the continuum to a series of patches with limited overlapping support.

Finite difference methods are easy to implement but best suited to structured grids with points typically aligned in the same directions as the coordinate system. Clearly the arbitrarily located set of n_p points do not satisfy this condition making the implementation of a finite difference spatial discretisation complex.

Finite element and finite or control volume methods make very few prior assumptions about the structure of the underlying mesh. Instead the arbitrary alignment of the grid points is incorporated into their formulation making them highly suited to unstructured meshes. Hence, as discussed below, finite element and finite volume spatial discretisations were implemented as part of this thesis. However, as the n_t time levels are structured and aligned with the time axis the temporal discretisation was performed using a point based finite difference method.

2.2.1 Finite Element Method

The finite element method is a member of the Galerkin family of discretisation methods (Zienkiewicz, 1977). By varying the mesh connectivity and the smoothness requirements of the discrete solution it encompasses a broad range of discretisations, a number of which will be investigated in this project. However, despite this flexibility the derivation of a discretised equation is quite systematic.

Taking the momentum equation as an example, first the smoothness requirements on the solution, u_i, are lowered and the boundary conditions imposed. This is achieved by multiplying Equation 2.7a throughout by a test function, ω_i, integrating over the domain, Ω, and integrating the advection and stress terms by parts:

$$\int_\Omega \omega_i \rho u_{i,t} \mathrm{d}V - \int_\Omega \omega_{i,j} \rho u_i u_j \mathrm{d}V + \int_{\partial\Omega_h} \omega_i \rho u_i u_j n_j \mathrm{d}s - \int_\Omega \omega_i \rho u_i u_{j,j} \mathrm{d}V + \int_\Omega \omega_{i,j} \mu u_{i,j} \mathrm{d}V = -\int_\Omega \omega_i p_{,i} \mathrm{d}V + \int_\Omega \omega_i f_i \mathrm{d}V - \int_{\partial\Omega_{g_i}} \omega_i \rho u_j n_j d_i \mathrm{d}s + \int_{\partial\Omega_{g_i}} \omega_i \mu d_{i,j} n_j \mathrm{d}s + \int_{\partial\Omega_h} \omega_i h_i \mathrm{d}s, \quad (2.14)$$

where use has been made of the boundary conditions (Equations 2.8–2.9).

Equation 2.14 is referred to as the weak form of the momentum equation (Zienkiewicz, 1977). It has the advantage that it naturally admits the boundary conditions into the formulation. Furthermore its solution need only be differentiable once ($u_i \in C^1$ for Equation 2.14) rather than twice ($u_i \in C^2$ for Equation 2.7a).

In fact the smoothness requirements can be relaxed further. It is only necessary to require that the weak solution, u_i, and its derivative, $u_{i,j}$, be real valued functions such that $\int_\Omega |u|^2 \mathrm{d}V$ and $\int_\Omega |u_{i,j}|^2 \mathrm{d}V$ are bounded (Zienkiewicz, 1977). In other words, $u_i \in H^1(\Omega)$, where $H^1(\Omega)$ is the first Hilbertian Sobolev space.[1] The set of solutions satisfying these requirements is referred to as the trial space.

Similar requirements exist on ω_i, referred to as the test space. Hence, a solution, $u_i \in H^1(\Omega)$, is sought to Equation 2.14 which is valid for all $\omega_i \in H^1(\Omega)$. However, if the solution to the weak equation (2.14) satisfies extra smoothness conditions (i.e. if $u_i \in C^2$) then it is also a solution to the strong equation (2.7a).

The second step in the finite element discretisation is to replace the test and trial spaces with finite dimensional subspaces, $H^1_h(\Omega) \subset H^1(\Omega)$. In other words the test and trial spaces cease to be continuous and can instead be represented on the nodes of the mesh.

As different discretisations are being used in space and time it is essential that the temporal and spatial variabilities are separated. Hence, potentially time dependent nodal values, $u_{iA}(t)$ and w_{iA}, are associated with each solution node, A. These have no individual spatial variability. Instead, to take account of the spatial variation between these values interpolating shape functions, $N_A(x_j)$, are also introduced at each node. Thus,

[1] The first Hilbertian Sobolev space is defined as: $H^1(\Omega) = \{w \in L_2(\Omega) : w_{,j} \in L_2(\Omega), j = 1, \ldots, d\}$, where $L_2(\Omega)$ denotes the set of real valued functions defined in Ω such that $\int_\Omega |w(x_j)|^2 \mathrm{d}V < \infty$.

the discrete test and trial spaces may be represented as:

$$\omega_i = N_A(x_j)\,\omega_{iA}(t) \tag{2.15}$$

$$u_i = N_A(x_j)\,u_{iA}(t)\,. \tag{2.16}$$

Due to their role as interpolants, certain restrictions on the choice of $N_A(x_j)$ are imposed. They must be piecewise polynomial functions over some small subdivision, or element, Ω_e, of the computational domain. Additionally, their support must be limited such that $N_A(x_j) = 1$ at node A and 0 at all other grid points. An element is then defined as a region of overlapping support between two of the shape functions, within which N_A is piecewise polynomial. This corresponds to a tile or volume in the domain tessellation.

The relationship between the set of n_p vertices of this tessellation and the set of n_s solution nodes, A, is dependent on a number of factors. For instance, the number of solution nodes in an element, n_e, is dictated by the polynomial order of the shape function while the number of vertices is dictated by the shape of the element itself and the polynomial order required to describe its boundaries.

Furthermore, while all vertices must be connected to each other through the tessellation and hence be shared between neighbouring elements, shape functions may be continuous (see Figure 2.1) or discontinuous (see Figure 2.2) across element boundaries. If the support of a shape function is limited to a single element (i.e. discontinuous) the solution nodes must also be limited to that element. Hence in a discontinuous element the solution nodes are not shared across elements and multiple solution nodes, belonging to different elements, may coexist at the same point in space.

In general the vertices define the elements which, given a particular interpolating shape function, define the locations and number of solution nodes. These may be the same (as in Figure 2.1(a, c)), co-located (as in Figure 2.2(a, c)) or more generally distributed (as in Figures 2.1(b, d) and 2.2(b, d)).

Continuous Galerkin

Initially the case of continuous Lagrange (C^1 at the vertices, Zienkiewicz & Taylor, 2000a, see Figure 2.1) shape functions is investigated. Substituting Equations 2.15 and 2.16 into 2.14:

$$\begin{aligned}\int_\Omega N_A\rho N_B \mathrm{d}V\, u_{iB,t} - \int_\Omega N_{A,j}\rho u_j N_B \mathrm{d}V\, u_{iB} + \int_{\partial\Omega_h} N_A\rho u_j n_j N_B \mathrm{d}s\, u_{iB} \\ - \int_\Omega N_A\rho u_{j,j} N_B \mathrm{d}V\, u_{iB} + \int_\Omega N_{A,j}\mu N_{B,j}\mathrm{d}V\, u_{iB} = -\int_\Omega N_A p_{,i}\mathrm{d}V + \int_\Omega N_A f_i \mathrm{d}V \\ - \int_{\partial\Omega_{g_i}} N_A\rho u_j n_j d_i \mathrm{d}s + \int_{\partial\Omega_{g_i}} N_A d_{i,j} n_j \mathrm{d}s + \int_{\partial\Omega_h} N_A h_i \mathrm{d}s, \end{aligned} \tag{2.17}$$

the finite element semi-discrete equation is produced, having been discretised in space and not time. Since ω_{iA} is arbitrary, the trivial solution, $\omega_{iA} = 0\,\forall\, A, i$, has been discarded.

It is now necessary to turn Equation 2.17 into a matrix equation of the form:

$$\mathbf{A}u_{iB} = b, \tag{2.18}$$

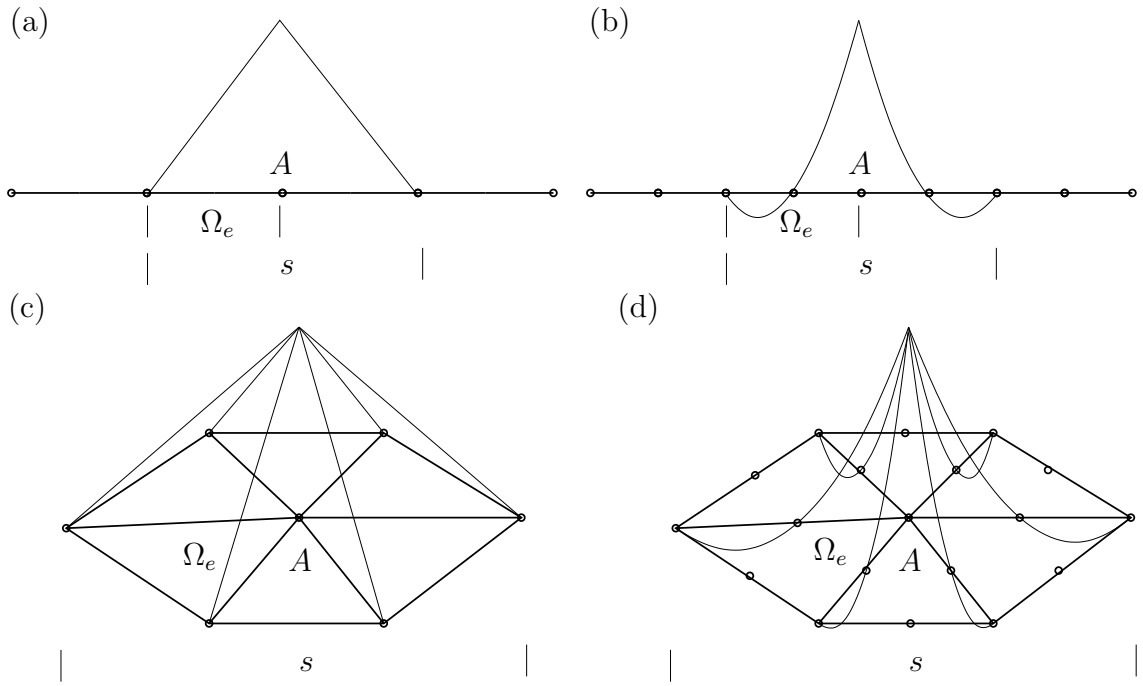


Figure 2.1: One-dimensional (a, b) and two-dimensional (c, d) schematics of piecewise linear (a, c) and piecewise quadratic (b, d) continuous Lagrange shape functions. The shape function has value 1 at node A descending to 0 at all surrounding nodes. The number of nodes per element, Ω_e, depends on the polynomial order while the support, s, extends to all the elements surrounding node A.

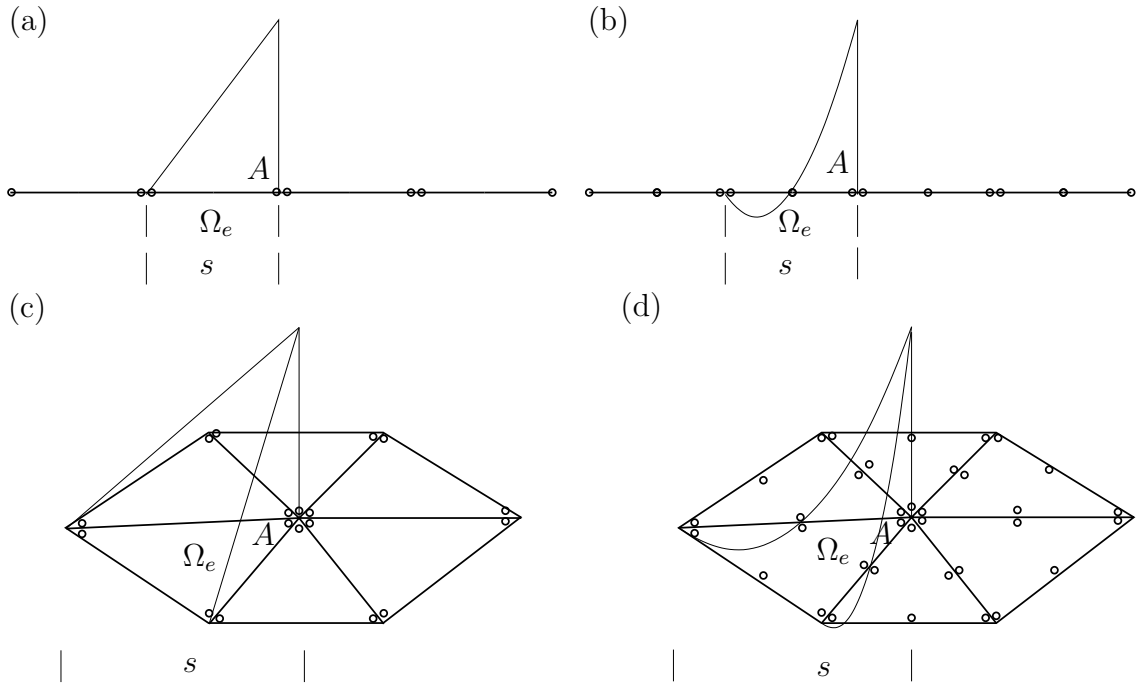


Figure 2.2: One-dimensional (a, b) and two-dimensional (c, d) schematics of piecewise linear (a, c) and piecewise quadratic (b, d) discontinuous shape functions. The shape function has value 1 at node A descending to 0 at all surrounding nodes. The number of nodes per element, Ω_e, depends on the polynomial order while the support, s, covers the same area as the element, Ω_e.

which can then be solved numerically for u_{iB} by inverting, or approximating the inverse to, $\mathbf{A}$. The assembly of $\mathbf{A}$ can be simplified by reducing the equation from a global system to a series of local elemental assembly steps.

As the support of N_A is limited to a finite set of elements that share node A, the product of N_A and N_B is only non-zero in those elements that contain both nodes A and B. Hence, Equation 2.17 can be reduced to a series of elemental integrals producing local $n_e \times n_e$ matrices. These can then be inserted into a global $n_s \times n_s$ matrix equation.

Furthermore, as A will only have a relationship with nodes that it shares an element with, the global matrices will be sparse, only having non-zero entries for those nodes that are connected to each other. By only storing non-zero entries this greatly reduces the size of the matrix and hence the computational cost of storing it.

The elemental matrix and right hand side contributions therefore take the form:

$$\mathbf{m}_{ab} = \int_{\Omega_e} N_a \rho N_b \mathrm{d}V \qquad \Rightarrow \mathbf{M}_{AB} \quad \text{mass matrix} \tag{2.19a}$$

$$\begin{aligned}\mathbf{n}_{ab} = &- \int_{\Omega_e} N_{a,j} \rho u_j N_b \mathrm{d}V - \int_{\Omega_e} N_a \rho u_{j,j} N_b \mathrm{d}V \\ &+ \int_{\substack{\partial\Omega_e \\ \in \partial\Omega_h}} N_a \rho u_j n_j N_b \mathrm{d}s \qquad \Rightarrow \mathbf{N}_{AB} \quad \text{advection matrix}\end{aligned} \tag{2.19b}$$

$$\mathbf{k}_{ab} = \int_{\Omega_e} N_{a,j} \mu N_{b,j} \mathrm{d}V \qquad \Rightarrow \mathbf{K}_{AB} \quad \text{stress matrix} \tag{2.19c}$$

$$g_{ai} = - \int_{\Omega_e} N_a p_{,i} \mathrm{d}V \qquad \Rightarrow \mathbf{g}_{Ai} \quad \text{pressure gradient} \tag{2.19d}$$

$$f_{ai} = \int_{\Omega_e} N_a f_i \mathrm{d}V \qquad \Rightarrow \mathbf{f}_{Ai} \quad \text{body forces} \tag{2.19e}$$

$$\begin{aligned}b_{ai} = &- \int_{\substack{\partial\Omega_e \\ \in \partial\Omega_{g_i}}} N_a \rho u_j n_j d_i \mathrm{d}s \\ &+ \int_{\substack{\partial\Omega_e \\ \in \partial\Omega_{g_i}}} N_a \mu d_{i,j} n_j \mathrm{d}s + \int_{\substack{\partial\Omega_e \\ \in \partial\Omega_h}} N_a h_i \mathrm{d}s \qquad \Rightarrow \mathbf{b}_{Ai} \quad \text{boundary conditions}\end{aligned} \tag{2.19f}$$

where lowercase indices, a and b, imply elemental node numberings translated by an indexing operation, $\Rightarrow$, to global, uppercase indices, A and B. Hence, Equation 2.17 becomes:

$$\mathbf{M}u_{i,t} + \mathbf{N}u_i + \mathbf{K}u_i = \mathbf{g}_i + \mathbf{f}_i + \mathbf{b}_i \tag{2.20}$$

where the global index notation has been dropped for simplicity.

The final step in the discretisation process is to approximate the time derivatives and select the time level at which the advection and stress terms will be evaluated. While the discretisation so far has made no assumptions about the structure of the mesh spatially it is now required that the time discretisation be uniform across the domain, taking the same time step globally.

Using a finite difference θ-method Equation 2.20 becomes:

$$\mathbf{M}\frac{u_i^{n+1} - u_i^n}{\Delta t} + \theta \mathbf{N} u_i^{n+1} + \theta \mathbf{K} u_i^{n+1} = -\left(1-\theta\right)\mathbf{N}u_i^n - \left(1-\theta\right)\mathbf{K}u_i^n + \mathbf{g}_i + \mathbf{f}_i + \mathbf{b}_i \tag{2.21}$$

where Δt is the timestep between time levels n and $n+1$ and $\theta = [0, 1]$ controls whether the equation is explicit, $\theta = 0$, or implicit, $\theta = 1$.

Fully explicit timestepping bases the new solution on information from the previous time level only. In this case the only contribution to the left hand side is the mass matrix, **M**. Fully implicit timestepping, on the other hand, incorporates information from the new solution by including the advection, **N**, and stress matrices, **K**, in the left hand side of the equation.

The advection matrix however is nonlinear as its assembly depends on the value of the velocity. It is therefore not possible to treat it fully implicitly. In order to recognise this it is rewritten such that:

$$\mathbf{n}_{ab} = -\int_{\Omega_e} N_{a,j}\rho\tilde{u}_j N_b \mathrm{d}V - \int_{\Omega_e} N_a\rho\tilde{u}_{j,j}N_b\mathrm{d}V + \int_{\substack{\partial\Omega_e \\ \in\partial\Omega_h}} N_a\rho\tilde{u}_j n_j N_b \mathrm{d}s$$

where $\tilde{u}_j$ implies that the velocity term inside the integral is nonlinear. This is discretised using a relaxation parameter, θ_I, such that:

$$\tilde{u}_j = \theta_I u_j^* + (1-\theta_I)\, u_j^n$$

where u_j^* is the best available approximation to u_j^{n+1}. It can now be seen that Equation 2.21 is, with some minor rearrangement of terms, in the form of Equation 2.18.

Discontinuous Galerkin

Discontinuous Galerkin methods follow a very similar discretisation strategy to continuous methods. The solution nodes however are not shared across elements so each element becomes an isolated problem.

Within an individual element the weak form therefore becomes:

$$\int_{\Omega_e} N_a\rho N_b \mathrm{d}V\, u_{ib,t} - \int_{\Omega_e} N_{a,j}\rho u_j N_b \mathrm{d}V\, u_{ib} + \int_{\partial\Omega_e} N_a\rho u_j n_j u_i|_{\partial\Omega_e}\mathrm{d}s - \int_{\Omega_e} N_a\rho u_{j,j}N_b\mathrm{d}V\, u_{ib}$$
$$+ \int_{\Omega_e} N_{a,j}\mu N_{b,j}\mathrm{d}V\, u_{ib} - \int_{\partial\Omega_e} N_a\mu n_j u_{i,j}|_{\partial\Omega_e}\mathrm{d}s = -\int_{\Omega} N_a p_{,i}\mathrm{d}V + \int_{\Omega} N_a f_i \mathrm{d}V, \quad (2.22)$$

However, a problem becomes immediately apparent: the test and trial spaces and their derivatives are not defined on the element boundaries where surface integrals need to be evaluated.

This problem can be overcome by treating the undefined elemental surface integrals as boundary conditions on that element. By using the values in neighbouring elements (except when on a domain boundary) the individual elements become coupled across the domain.

For instance, in the case of the surface integrals of the advection term, the values on face f of the elemental boundary, $\partial\Omega_e$, may be upwinded such that:

$$u_i|_{\partial\Omega_{e_f}} = \begin{cases} N_b u_{ib}|_{\partial\Omega_{e_f}} & \text{if } \bar{\tilde{u}}_j n_j|_{\partial\Omega_{e_f}} > 0 \Rightarrow I = 0, \\ N_c u_{ic}|_{\partial\Omega_{e_f}} & \text{if } \bar{\tilde{u}}_j n_j|_{\partial\Omega_{e_f}} < 0 \Rightarrow I = 1. \end{cases} \quad (2.23)$$

where $\bar{\tilde{u}}_j|_{\partial\Omega_{e_f}}$ $\left(= \left(N_b\tilde{u}_{jb}|_{\partial\Omega_{e_f}} + N_c\tilde{u}_{jc}|_{\partial\Omega_{e_f}}\right)/2\right)$ is the average nonlinear flux across face f, b is the local node numbering of the element, Ω_e, c is that of the neighbouring element

across face f, and n_j is the outward pointing normal to face f. I has also been defined in Equation 2.23 as a parameter that indicates whether the flow is incoming, $I = 1$, or not, $I = 0$. By this mechanism, when the nonlinear average flux is incoming to an element, the information is propagated from its neighbour and vice-versa. When the face has no neighbour (i.e. it is on the domain boundary) the values of u_{ic} are taken from the boundary condition.

Unlike the advection term the elemental surface integral for the stress term requires the evaluation of derivatives on the surfaces of the elements. Several schemes have been proposed for this (Arnold *et al.*, 2000) most of which introduce an auxiliary variable, $q_j = N_a q_{ja}$, such that:

$$q_j = u_{i,j} \tag{2.24}$$

and

$$\int_{\Omega_e} N_{a,j}\mu N_{b,j}\mathrm{d}V\, u_{ib} - \int_{\partial\Omega_e} N_a\mu n_j u_{i,j}|_{\partial\Omega_e}\mathrm{d}s = \int_{\Omega_e} N_{a,j}\mu N_b\mathrm{d}V\, q_{jb} - \int_{\partial\Omega_e} N_a\mu n_j q_j|_{\partial\Omega_e}\mathrm{d}s \tag{2.25}$$

The new auxiliary equation (2.24) can be discretised and q_j found using the Galerkin projection:

$$\int_{\Omega_e} N_a N_b\mathrm{d}V q_{jb} = -\int_{\Omega_e} N_{a,j}N_b\mathrm{d}V u_{ib} + \int_{\partial\Omega_e} N_a n_j u_i|_{\partial\Omega_e}\mathrm{d}s. \tag{2.26}$$

This reduces the problem to one where $q_i|_{\partial\Omega_e}$ and $u_i|_{\partial\Omega_e}$ are not defined. Bassi & Rebay (1997) proposed that the average value across a face, f:

$$q_i|_{\partial\Omega_{e_f}} = \frac{N_b q_{ib}|_{\partial\Omega_{e_f}} + N_c q_{ic}|_{\partial\Omega_{e_f}}}{2}, \tag{2.27}$$

$$u_i|_{\partial\Omega_{e_f}} = \frac{N_b u_{ib}|_{\partial\Omega_{e_f}} + N_c u_{ic}|_{\partial\Omega_{e_f}}}{2}, \tag{2.28}$$

making Equations 2.25 and 2.26 transpose operations (up to a factor of $-\mu$).

This method is known to be associated with some numerical issues. For instance, as the mesh is refined, reducing the distance between grid points, the solution may not converge to the exact solution as quickly as would be expected (Karniadakis & Sherwin, 1999).

Other schemes, however, generally require the evaluation of derivatives at the element surface, which restricts their usage to shape functions of at least linear polynomial order. Therefore, due to the relative simplicity of Equations 2.27 and 2.28 and their applicability to all polynomial orders they are selected for use here.

A disadvantage of all auxiliary variable schemes is that they introduce an additional equation (2.24). However, as the basis functions are discontinuous, the left hand side of Equation 2.26 only uses solution nodes within a single element. Therefore it can be solved locally and the resulting auxiliary solution inserted implicitly back into Equation 2.25.

Implicit insertion of the auxiliary equation has the disadvantage that it creates a large nodal stencil (Sherwin *et al.*, 2006) in the resulting momentum equation. An element requires information not only from its neighbours but from their neighbours as well. However, it has the advantage that it allows the stress terms to be treated fully implicitly.

Hence, again using the finite difference θ-method for the temporal discretisation, the overall form of Equation 2.21 remains unchanged:

$$\mathbf{M}\frac{u_i^{n+1}-u_i^n}{\Delta t}+\theta\mathbf{N}u_i^{n+1}+\theta\mathbf{K}u_i^{n+1}=-\left(1-\theta\right)\mathbf{N}u_i^n-\left(1-\theta\right)\mathbf{K}u_i^n+\mathbf{g}_i+\mathbf{f}_i+\mathbf{b}_i,$$

although the definitions of the advection and stress operators change significantly:

$$\mathbf{m}_{ab}=\int_{\Omega_e}N_a\rho N_b\mathrm{d}V \qquad \Rightarrow\mathbf{M}_{AB} \quad \text{mass matrix} \tag{2.29a}$$

$$\begin{aligned}\mathbf{n} =&-\int_{\Omega_e}N_{a,j}\rho\tilde{u}_jN_b\mathrm{d}V-\int_{\Omega_e}N_a\rho\tilde{u}_{j,j}N_b\mathrm{d}V\\ &+I\int_{\substack{\partial\Omega_e\\ \ni\partial\Omega_{g_i}}}N_a\rho\bar{\tilde{u}}_jn_jN_c\mathrm{d}s\\ &+(1-I)\int_{\partial\Omega_e}N_a\rho\bar{\tilde{u}}_jn_jN_b\mathrm{d}s \qquad \Rightarrow\mathbf{N} \quad \text{advection matrix}\end{aligned} \tag{2.29b}$$

$$\mathbf{m}_{\mathbf{q}ab}=\int_{\Omega_e}N_aN_b\mathrm{d}V \qquad \Rightarrow\mathbf{M}_{\mathbf{q}AB} \quad \text{auxiliary mass} \tag{2.29c}$$

$$\begin{aligned}\mathbf{c}_{\mathbf{q}j} =&-\int_{\Omega_e}N_{a,j}N_b\mathrm{d}V+\frac{1}{2}\int_{\substack{\partial\Omega_e\\ \ni\partial\Omega_{g_i}}}N_an_jN_b\mathrm{d}s\\ &+\frac{1}{2}\int_{\substack{\partial\Omega_e\\ \ni\partial\Omega_{g_i}}}N_an_jN_c\mathrm{d}s \qquad \Rightarrow\mathbf{C}_{\mathbf{q}j} \quad \text{auxiliary gradient}\end{aligned} \tag{2.29d}$$

$$\mathbf{k}=\mathbf{c}_{\mathbf{q}j}^{\,T}\mu\mathbf{m}_{\mathbf{q}ab}^{-1}\mathbf{c}_{\mathbf{q}j} \qquad \Rightarrow\mathbf{K} \quad \text{stress matrix} \tag{2.29e}$$

$$g_{ai}=-\int_{\Omega_e}N_ap_{,i}\mathrm{d}V \qquad \Rightarrow\mathbf{g}_{Ai} \quad \text{pressure gradient} \tag{2.29f}$$

$$f_{ai}=\int_{\Omega_e}N_af_i\mathrm{d}V \qquad \Rightarrow\mathbf{f}_{Ai} \quad \text{body forces} \tag{2.29g}$$

$$\begin{aligned}b_{ai} =&-I\int_{\substack{\partial\Omega_e\\ \in\partial\Omega_{g_i}}}N_a\rho\tilde{u}_jn_jd_i\mathrm{d}s\\ &-\mathbf{c}_{\mathbf{q}j}^{\,T}\mu\mathbf{m}_{\mathbf{q}bc}^{-1}\int_{\substack{\partial\Omega_e\\ \in\partial\Omega_{g_i}}}N_cd_in_j\mathrm{d}s\\ &+\int_{\substack{\partial\Omega_e\\ \in\partial\Omega_h}}N_ah_i\mathrm{d}s \qquad \Rightarrow\mathbf{b}_{Ai} \quad \text{boundary conditions}\end{aligned} \tag{2.29h}$$

Here matrix indices have been dropped on matrices that are no longer assembled local to a single element.

The above derivation has focussed on discontinuous shape functions with a polynomial order of at least 1 (see Figure 2.2). However, it is valid for piecewise constant (polynomial order 0, see Figure 2.3) with only minor modifications. In the advection matrix, the integral of the velocity divergence must be integrated by parts as it cannot be evaluated

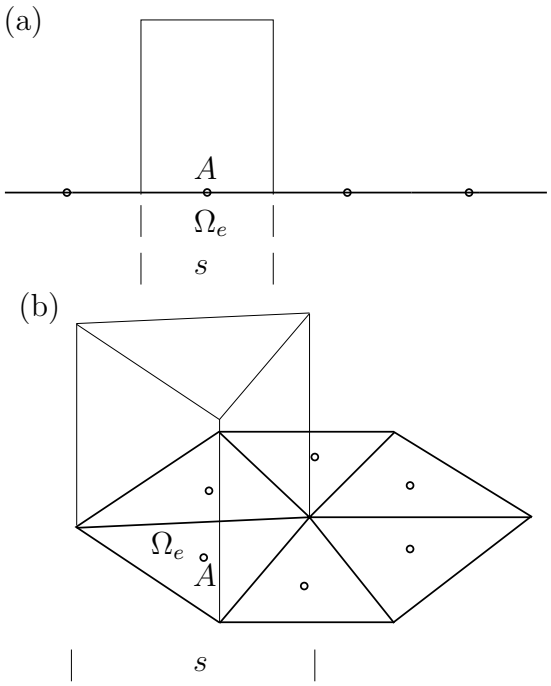


Figure 2.3: One-dimensional (a) and two-dimensional (b) schematics of piecewise constant shape functions. The shape function has value 1 at node A and across the element, Ω_e, descending to 0 at the element boundaries. As with other discontinuous shape functions, the support, s, coincides with the element, Ω_e.

when the velocity is piecewise constant. Additionally, both the advection matrix and the auxiliary gradient matrix can be simplified by removing all terms involving the derivative of a shape function as $N_{a,j} = 0$:

$$\mathbf{n} = I\left(\int_{\substack{\partial\Omega_e \\ \ni\partial\Omega_{g_i}}} N_a\rho\bar{\tilde{u}}_j n_j N_c \mathrm{d}s - \int_{\partial\Omega_e} N_a\rho\bar{\tilde{u}}_j n_j N_b \mathrm{d}s\right) \tag{2.30a}$$

$$\mathbf{c}_{\mathbf{q}j} = \frac{1}{2}\int_{\substack{\partial\Omega_e \\ \ni\partial\Omega_{g_i}}} N_a n_j N_b \mathrm{d}s + \frac{1}{2}\int_{\substack{\partial\Omega_e \\ \ni\partial\Omega_{g_i}}} N_a n_j N_c \mathrm{d}s \tag{2.30b}$$

These modifications effectively reduce the finite element method to an upwinded finite volume discretisation. This derivation can be reached in a simpler manner without considering test and trial spaces as will be shown below.

2.2.2 Finite Volume Method

The finite volume method is also a weighted residual method. As has just been shown it is effectively the lowest order discontinuous Galerkin discretisation. As with the discontinuous Galerkin method the domain is initially discretised into a finite number of volumes, $\partial\Omega_v$, which may (as above) coincide with the elements used in the finite element discretisation.

For example, the linear advection equation (2.11) may be rewritten using the product

rule:

$$\alpha_{,t} + \underbrace{(u_i\alpha)_{,i}}_{\text{advective flux term}} - \underbrace{\alpha u_{i,i}}_{\text{velocity divergence term}} = 0, \tag{2.31}$$

and then, in a single volume, A, of the domain, integrated by parts:

$$\int_{\Omega_v} \mathrm{d}V\, \alpha_{A,t} + \underbrace{\int_{\partial\Omega_v} \tilde{u}_i n_i \alpha|_{\partial\Omega_v} \mathrm{d}s}_{\text{advective flux term}} - \underbrace{\int_{\partial\Omega_v} \tilde{u}_i n_i \mathrm{d}s\, \alpha_A}_{\text{velocity divergence term}} = 0. \tag{2.32}$$

As with the advection term in the discontinuous Galerkin method the ambiguity over the definition of $\alpha|_{\partial\Omega_v}$ allows neighbouring volumes to be coupled together. For the discontinuous Galerkin discretisation this was selected as the upwind flux as it is simple to implement and easily generalises to different orders of test and trial spaces. It also helps to smooth, or dissipate, the local oscillations that occur within the elements.

However, this numerical diffusion means that first-order upwinding is generally unsuitable for finite volume advection, smearing out features rapidly. Higher-order methods that suffer from less dissipation are available but are not guaranteed to be bounded. For example, the average value at a face f:

$$\alpha|_{\partial\Omega_{v_f}} = \frac{\alpha_A + \alpha_B|_{\partial\Omega_{v_f}}}{2}, \tag{2.33}$$

where $\alpha_B|_{\partial\Omega_{v_f}}$ is the value in the neighbouring volume across face f, is guaranteed to be unbounded when used as a flux (Rudman, 1997). In fact, first-order upwinding is the only method that guarantees monotonic, bounded advection (LeVeque, 1992). Therefore, to use higher-order methods a 'limiting' step must be introduced. This checks the face value produced by the higher-order method and if it fails some test of boundedness it is adjusted such that the solution remains monotonic.

Such a limiting step prevents the advective flux from being implemented fully implicitly, which constrains the size of the permitted timestep. Low-order methods however can be used to gain some implicit information if the face value is split into three parts (LeVeque, 1992):

- a 'pivot' value evaluated at the face by a low-order method which can be implemented implicitly, $\alpha|^{LO}_{\partial\Omega_v}$
- the best guess at the high-order solution, $\tilde{\alpha}|^{HO}_{\partial\Omega_v}$, which is then limited, $\hat{\tilde{\alpha}}|^{HO}_{\partial\Omega_v}$
- the best guess at the low-order solution, $\tilde{\alpha}|^{LO}_{\partial\Omega_v}$

Hence:

$$\alpha|_{\partial\Omega_{v_f}} = \alpha|^{LO}_{\partial\Omega_{v_f}} + \hat{\tilde{\alpha}}|^{HO}_{\partial\Omega_{v_f}} - \tilde{\alpha}|^{LO}_{\partial\Omega_{v_f}}. \tag{2.34}$$

This system can be iterated upon, updating the best guess high and low-order solutions after each iteration. If the system converges then the low-order solutions cancel each other out and the limited high-order face value is used.

Selecting first-order upwinding as the low-order solution method, and again introducing a finite difference θ timestepping algorithm, the fully discretised finite volume advection equation may then be written:

$$\begin{aligned}
&\int_{\Omega_v} \mathrm{d}V \, \frac{\alpha_A^{n+1} - \alpha_A^n}{\Delta t} \\
&+ \underbrace{\theta_p \left(1 - I\right) \int_{\partial\Omega_v} \tilde{u}_i n_i \mathrm{d}s \, \alpha_A^{n+1} + \theta_p I \int_{\partial\Omega_v} \tilde{u}_i n_i \mathrm{d}s \, \alpha_B^{n+1}}_{\text{implicit low-order advective flux term}} - \underbrace{\theta \int_{\partial\Omega_v} \tilde{u}_i n_i \mathrm{d}s \, \alpha_A^{n+1}}_{\text{implicit velocity divergence term}} \\
&= \underbrace{-\theta \int_{\partial\Omega_v} \tilde{u}_i n_i \hat{\tilde{\alpha}}|_{\partial\Omega_v}^{HO} \mathrm{d}s - (1-\theta) \int_{\partial\Omega_v} \tilde{u}_i n_i \hat{\alpha}^n|_{\partial\Omega_v}^{HO} \mathrm{d}s}_{\text{high-order advective flux term}} \\
&+ \underbrace{\theta_p \left(1 - I\right) \int_{\partial\Omega_v} \tilde{u}_i n_i \mathrm{d}s \, \tilde{\alpha}_A + \theta_p I \int_{\partial\Omega_v} \tilde{u}_i n_i \mathrm{d}s \, \tilde{\alpha}_B}_{\text{nonlinear low-order advective flux term}} - \underbrace{(1-\theta) \int_{\partial\Omega_v} \tilde{u}_i n_i \mathrm{d}s \, \alpha_A^n}_{\text{explicit velocity divergence term}}
\end{aligned} \tag{2.35}$$

where θ_p controls the time discretisation of the low-order pivot solution and node B belongs to the set of nodes neighbouring node A across the volume boundary, $\partial\Omega_v$.

Selecting different methods for the calculation of the high-order face value leads to a vast range of behaviours of the discretised advection equation. Some of these will be discussed in Chapter 3.

2.2.3 Numerical Solutions

The assembly of Equations 2.21 and 2.35 requires the evaluation of integrals, which is not generally possible analytically. Instead these integrals must be approximated using numerical integration. Using such quadrature rules the definite integral of a polynomial function, p, in a canonical domain is approximated by a summation at a discrete number of points:

$$\int_{\Omega_\xi} p(\xi_i) \mathrm{d}V \approx \sum_{q=1}^{n_q} w_q p(\xi_i(q)), \tag{2.36}$$

where w_q are weights associated with the n_q quadrature points $\xi_i(q)$ in the canonical coordinate system ξ_i, $i = 1, \ldots, d$. Rules giving the positions of these quadrature points and their associated weight values are taken from the Encyclopaedia of Cubature (Cools, 2003; Cools & Rabinowitz, 1993; Cools, 1999).

The number of quadrature points, n_q, is determined by the degree of accuracy required which in turn depends on the order of the polynomial function. In the case of the integrals required in the above discretisations the order of p is determined by the sum of the orders of the shape functions and any other coefficients (e.g. ρ) within the integral. Generally the degree of accuracy of the quadrature rule is taken to be at least 1 higher than this sum to ensure the solution is adequately represented.

In general the elements on an unstructured mesh will not conform to the canonical domain of integration. Hence the determinant of the Jacobian matrix, $|J_{ij}|_q$, evaluated at a quadrature point q, is used to transform between the physical and canonical coordinate

systems:

$$\int_{\Omega} p(x_i)\mathrm{d}V \approx \sum_{q=1}^{n_q} w_q p(\xi_i(q))|J_{ij}|_q, \tag{2.37}$$

where:

$$J_{ij} = \frac{\partial x_i}{\partial \xi_j}. \tag{2.38}$$

The Jacobian matrix is also used to transform the derivatives of the shape functions:

$$N_{a,i} = J_{ij}^{-1}\frac{\partial N_a}{\partial \xi_j}, \tag{2.39}$$

so it is only necessary to store the values of the shape functions and their derivatives at the quadrature points in canonical space. It is now possible to assemble Equations 2.21 and 2.35 into matrix form. The resulting global matrices are sparse due to the limited overlapping support of the shape functions and the choice of neighbouring values in the implicit surface integrals.

Despite this sparsity the matrices can still become very large if high resolution, three-dimensional simulations are performed. Hence direct solution methods are not feasible and instead iterative solvers are used. The PETSc library of solvers (Satish *et al.*, 2001, 2004; Balay *et al.*, 1997), which is used throughout this project, provides a wide range of parallel-ready linear solvers and preconditioners.

Iterative methods find successive approximations to the solution of systems of the form:

$$\mathbf{A}y = b, \tag{2.40}$$

such that, if the approximations converge, after k iterations $y_k \approx \mathbf{A}^{-1}b$, where $\mathbf{A}$ is a square real matrix. Iterative methods approximate the action of $\mathbf{A}^{-1}$ without explicitly forming it. Instead the most important operation in most solvers is the matrix vector multiplication $\mathbf{A}y$. This allows the solver to exploit the sparsity structure of the matrix to minimise the number of operations. If the inverse were formed, it would not necessarily have a sparse structure thereby increasing both the number of operations and the memory requirements.

The number of iterations an iterative method requires depends on the eigenspectrum of the matrix $\mathbf{A}$. This effectively means that the closer $\mathbf{A}$ is to the identity matrix the faster the convergence rate will be and the lower k will be. Preconditioning aims to improve the spectrum of $\mathbf{A}$ by multiplying the system (Equation 2.40) by a preconditioning matrix $\hat{\mathbf{A}}$. Focusing on left preconditioning only, Equation 2.40 therefore becomes:

$$\hat{\mathbf{A}}^{-1}\mathbf{A}y = \hat{\mathbf{A}}^{-1}b \tag{2.41}$$

Clearly the most effective preconditioning matrix would be $\hat{\mathbf{A}} = \mathbf{A}$ however this would be impractical (shifting all the solver's work to the preconditioning step). Instead $\hat{\mathbf{A}}^{-1}$ should approximate the effect of $\mathbf{A}^{-1}$ while also being inexpensive to compute.

Throughout this project a single iteration of successive over-relaxation (Saad, 2003) or a related algorithm (Eisenstat, 1981) is used to precondition Equations 2.21 and 2.35. The preconditioned systems are then solved using the generalised minimal residual method

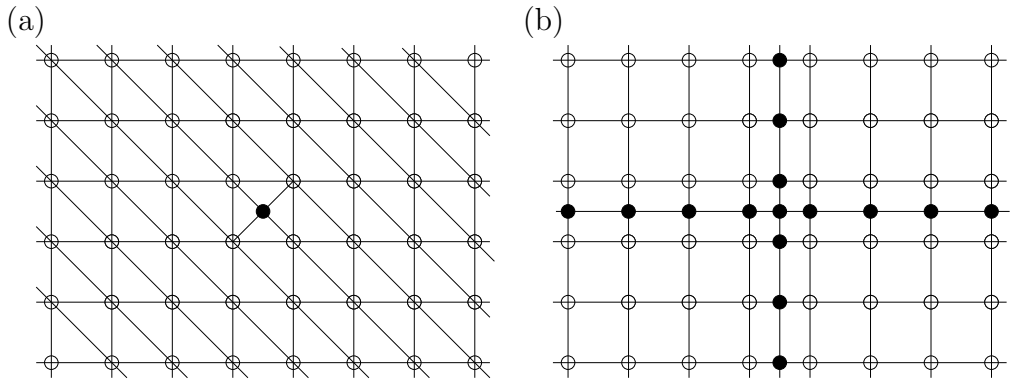


Figure 2.4: The addition of a vertex to a simplex mesh (a) and a cube mesh (b). In the simplex case only a single vertex and two elements are added to the mesh, whereas the same operation in the cube mesh requires the addition of many new vertices (highlighted in black) and elements.

(GMRES, Saad & Schultz, 1986; Silvester *et al.*, 2001) or the method of conjugate gradients (CG, Hestenes & Stiefel, 1952).

GMRES and CG are Krylov subspace methods (Saad, 2003) suitable for finding the solutions to nonsymmetric and symmetric matrix equations respectively. The restarted variant of GMRES, storing a maximum of 30 Krylov directions (Saad & Schultz, 1986), is used on Equations 2.21 and 2.35. CG is used on symmetric matrix equations (for instance those lacking advection terms).

2.3 Mesh Optimisation

The discretisation methods presented thus far have imposed very few restrictions on the spatial tessellation of the domain. The elements or volumes must have non-negative and non-zero length, area or volume and possess a one-to-one mapping to and from the canonical space. Additionally, the vertices must have a known connectivity to each other. The total number of solution nodes will be related to the number of elements and the polynomial degree used.

Elements constructed out of simplices – triangles (as demonstrated in Figures 2.1–2.3) in two dimensions and tetrahedra in three dimensions – are most suited to unstructured meshes. For instance, the introduction of extra nodes requires only local modifications to the mesh whereas with a cube mesh – quadrilaterals in two dimensions and hexahedra in three – this would require either the introduction of hanging vertices (belonging to one element but not its neighbour) or global variations in resolution (see Figure 2.4).

Given an initial piecewise linear continuous simplex mesh, the mesh optimisation libraries libadaptivity (on tetrahedra in three dimensions, Pain *et al.*, 2001) and libmba2d (on triangles in two dimensions, Vasilevski & Lipnikov, 1999) exploit this locality to improve the mesh quality on an element-by-element basis. Several local topological operations are permitted (see Figure 2.5):

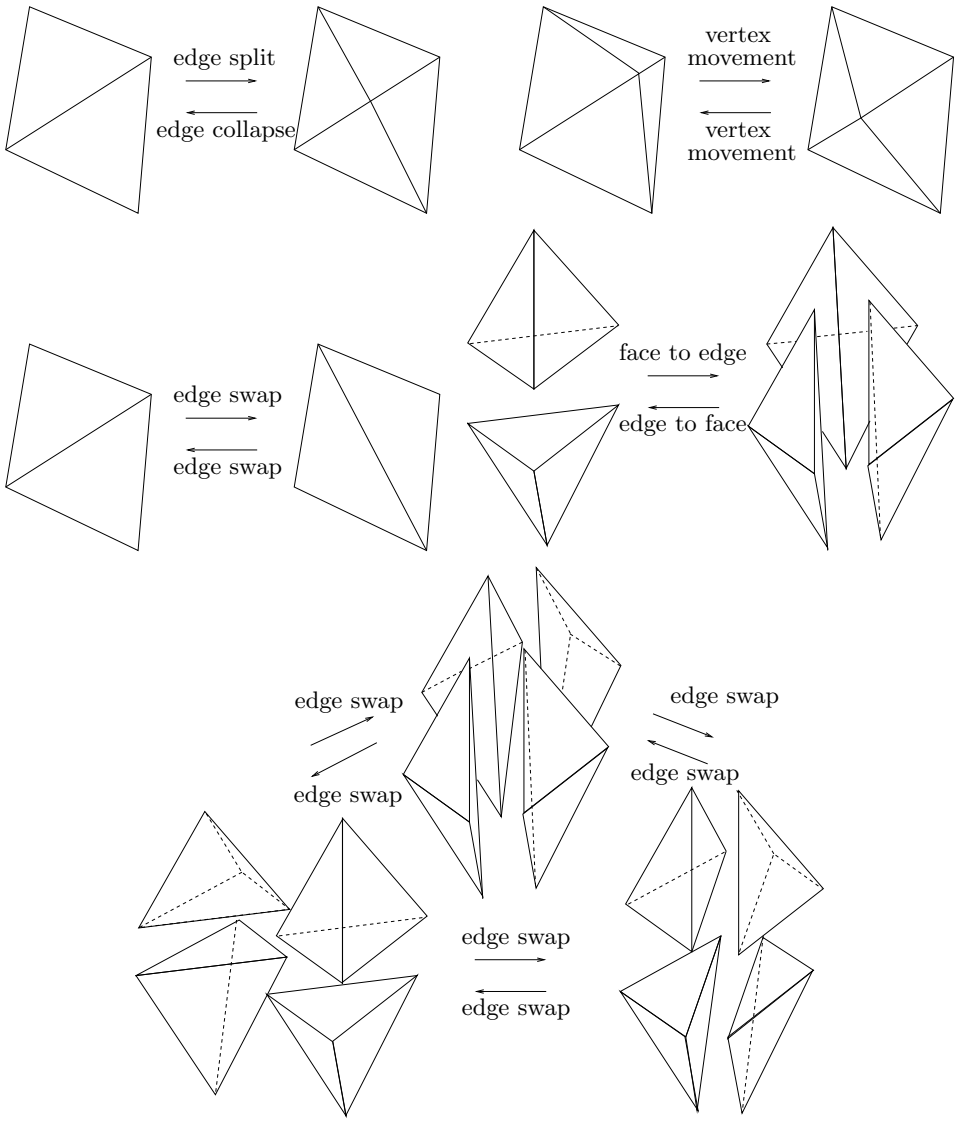


Figure 2.5: Topological operations performed on a local cluster of elements to optimise a mesh.

- vertex movement – this preserves the number of nodes but adjusts the position of an element vertex such that the surrounding elements are in some way improved
- edge splitting – effectively the introduction of a new vertex at the midpoint of an edge, this procedure creates new elements (for example two elements become four in two dimensions)
- edge collapsing – the inverse of edge splitting, this involves the removal of a vertex and the merging of elements
- edge swapping – preserves the number of nodes but manipulates the edge lengths and element shapes by changing the configuration of an edge between two elements in two dimensions and four elements in three

- face and edge swapping – in three dimensions only if two elements share a face and their combined interior is convex then the face can be converted into an edge
- edge and face swapping – again in three dimensions only, the reverse operation is possible when three elements surround an edge

Given this suite of operations it is therefore possible to adjust each cluster of elements to improve their quality. However, it is still necessary to define what is to be regarded as an improvement. To do this a mesh quality functional is defined which considers the size and shape of an individual element.

In two dimensions (in the libmba2d library, Vasilevski & Lipnikov, 1999) this takes the form:

$$Q_{\Omega_e}|_M = \underbrace{12\sqrt{3}\frac{A_{\Omega_e}|_M}{\left(\sum l_{\partial\Omega_{e_k}}|_M\right)^2}}_{\mathrm{I}} \underbrace{F\left(\frac{\sum l_{\partial\Omega_{e_k}}|_M}{3}\right)}_{\mathrm{II}}, \tag{2.42}$$

where A_{Ω_e} is the area of an element and $l_{\partial\Omega_{e_k}}$ is the length of edge k of element Ω_e. Additionally, $F(x) = [\min(x, 1/x)(2 - \min(x, 1/x)]^3$.

The first term (I) in Equation 2.42 considers the shape of element Ω_e. In the case of an equilateral triangle $l_{\partial\Omega_{e_k}} = l$ for all k and $A_{\Omega_e} = l^2\sqrt{3}/4$ so $\mathrm{I} = 1$. For non-equilateral triangles $\mathrm{I} < 1$. The second term (II) controls the size of the element. F is a smooth function with a maximum of 1 when the element is equilateral with unit edge length, $l_{\partial\Omega_{e_k}} = 1$ for all k. Thus in two dimensions $0 < Q_{\Omega_e} \leq 1$ and the highest quality element is an equilateral triangle with unit edge length.

In three dimensions (in the libadaptivity library, Pain *et al.*, 2001) the same considerations take the form:

$$Q_{\Omega_e}|_M = \underbrace{\left(\frac{1}{2\sqrt{6}r_{\Omega_e}|_M} - 1\right)^2}_{\mathrm{I}} + \underbrace{\frac{1}{2}\sum\left(l_{\partial\Omega_{e_k}}|_M - 1\right)}_{\mathrm{II}}, \tag{2.43}$$

where r_{Ω_e} is the radius of the inscribed sphere of element Ω_e.

Again the first term (I) in Equation 2.43 gauges the shape of element Ω_e. The leading factor has been chosen such that when the radius describes an equilateral tetrahedra $\mathrm{I} = 0$. Similarly the second term (II) considers the size of the element such that when all the edge lengths are unity $\mathrm{II} = 0$.

The objective of Equations 2.42 and 2.43 is the same: to define an optimal element as one with equal unit edge lengths. However, in three dimensions this is achieved by driving the Q_{Ω_e} towards 0 rather than 1 (as in the two-dimensional case). Given a sub-optimal element, mesh optimisation therefore attempts to drive its mesh quality functional towards one by performing the local operations listed above. However, using the mesh quality functionals described above will result in a mesh with a uniform edge length of 1.

In order to achieve anisotropic meshes with spatially varying resolution it is necessary to evaluate Q_{Ω_e} in a non-Euclidean spatially varying metric M, $Q_{\Omega_e}|_M$. Assuming that

M is constant across an element the various contributions to $Q_{\Omega_e}|_M$ can be evaluated as:

$$A_{\Omega_e}|_M = \sqrt{\det(M)} A_{\Omega_e}, \tag{2.44}$$

$$l_{\partial\Omega_{ek}}|_M = \sqrt{\mathbf{l}^T_{\partial\Omega_{ek}} M \mathbf{l}_{\partial\Omega_{ek}}}, \tag{2.45}$$

$$r_{\Omega_e}|_M = \frac{3V_{\Omega_e}|_M}{\sum_{k\in\Omega_e} A_{\partial\Omega_{ek}}|_M}, \tag{2.46}$$

where

$$V_{\Omega_e}|_M = \sqrt{\det(M)} V_{\Omega_e}, \tag{2.47}$$

is the volume of the element and in three dimensions $A_{\partial\Omega_{ek}}$ represents the areas of the face k surrounding element e. Additionally, $\mathbf{l}_{\partial\Omega_{ek}}$ is the vector pointing in the direction of edge k with magnitude equal to its length $l_{\partial\Omega_{ek}}$. All variables without a $|_M$ are evaluated in physical space.

The choice of the metric in which to evaluate the mesh quality functional should be based on some measure of the error in the solution. Then if the error along an edge exceeds some user defined error bound it would be translated into metric space as an edge length greater than one which optimisation should drive down. Similarly if the error was lower than an error bound it would result in a metric edge length less than unity which could be increased.

For a piecewise linear continuous approximation to a continuous field the bound on the interpolation error, ϵ, of field ψ in the direction and over the length of $\mathbf{v}$, is given by:

$$\epsilon \leq \mathbf{v}^T |H| \mathbf{v}, \tag{2.48}$$

where $|H|$ is the positive definite Hessian matrix:

$$H_{ij} = \psi_{,ij}. \tag{2.49}$$

Once evaluated the signs of the curvatures are removed by taking the absolute values of the eigenvalues of H_{ij} to form the final Hessian. This is done because only the magnitude and not the sign of the curvature is of interest.

Comparing Equations 2.45 and 2.48 a sensible choice of metric is given by:

$$M_\psi = \frac{1}{\hat{\epsilon}} |H|, \tag{2.50}$$

for some user defined interpolation error, $\hat{\epsilon}$. The edge lengths evaluated in this metric, $l_{\partial\Omega_{ek}}|_M$, will therefore tend towards unity as the interpolation error in that direction tends towards $\hat{\epsilon}$.

In order to recover the Hessian of a piecewise linear field an auxiliary variable, similar to that used in Equation 2.24, is introduced such that:

$$q_i = \psi_{,i}. \tag{2.51}$$

Discretising both q and ψ using piecewise linear continuous shape functions, N_a, the Hessian can then be recovered using a double Galerkin projection:

$$\mathbf{M_q} H_{ij} = \mathbf{C_{q}}_j q_i, \tag{2.52}$$

$$\mathbf{M_q} q_j = \mathbf{C_{q}}_j \psi, \tag{2.53}$$

where

$$\mathbf{m}_{\mathbf{q}ab} = \int_{\Omega_e} N_a N_b \mathrm{d}V \qquad \Rightarrow \mathbf{M}_{\mathbf{q}AB} \qquad \text{mass matrix} \tag{2.54a}$$

$$\mathbf{c}_{\mathbf{q}jab} = \int_{\Omega_e} N_a N_{b,j} \mathrm{d}V \qquad \Rightarrow \mathbf{C}_{\mathbf{q}jAB} \qquad \text{gradient matrix} \tag{2.54b}$$

The off-diagonal entries of the Hessian recovered in this manner are averaged to enforce the symmetry of the final solution.

To accelerate this calculation the mass matrix is normally lumped, $\mathbf{M}^L_{\mathbf{q}}$, such that:

$$\mathbf{M}^L_{\mathbf{q}ii} = \sum_j \mathbf{M}_{\mathbf{q}ij}, \tag{2.55}$$

$$\mathbf{M}^L_{\mathbf{q}ij} = 0, \, \forall i \neq j. \tag{2.56}$$

Mass lumping produces a diagonal matrix which, for piecewise linear fields, shares the conservation properties of the full mass matrix and can be directly inverted without the use of a linear solver. Mass lumping will be discussed further in Chapter 4.

Using this Hessian to define the metric is particularly appropriate when optimising the mesh to the interfaces between materials. In these regions the curvatures of the fields describing the materials will be particularly high. Additionally the curvatures will be directionally dependent with high curvatures across the interface and low curvatures along it. Such directional dependence is automatically incorporated into the metric through the Hessian. This can be seen in the eigen-decomposition of M_ψ where the orthonormal eigenvectors represent a rotated coordinate system in the directions of the principal curvatures. The magnitudes of these curvatures are given by the eigenvalues, λ_i, which in turn are related to the desired edge length, $\hat{l}_i$, in direction i by:

$$\hat{l}_i = \frac{1}{\sqrt{\lambda_i}}. \tag{2.57}$$

Multiple fields may vary across an interface and different interfaces may be described by different fields. Each field may be assigned an independent desired error, $\hat{\epsilon}_\psi$, which, in combination with the Hessian for that field, H_ψ, leads to multiple metrics. In these situations the minimum edge length requested is taken to form a single combined metric in which to evaluate the mesh quality.

The relationship between the edge lengths and the eigenvalues allows this combination to be performed through a metric superposition procedure. This can be thought of graphically (see Figure 2.6 in two dimensions) as overlaying the ellipsoids describing each field's metric. The principal axes of the ellipsoids correspond to the eigenvectors of M_ψ while their lengths are equal to the desired edge lengths. They can thus be combined by taking the largest ellipsoid that fits within all of the individual ellipsoids.

The superposition technique can also be used to impose maximum edge lengths, through the construction of, and superposition with, a maximum metric:

$$M_{\max} = \mathrm{diag}\left(\frac{1}{\sqrt{\hat{l}_{1\max}}}, \ldots, \frac{1}{\sqrt{\hat{l}_{d\max}}}\right). \tag{2.58}$$

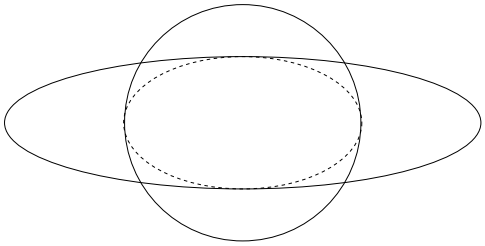


Figure 2.6: Graphical representation of the superposition of two metrics in two dimensions. The principal radii of the ellipsoids give the edge lengths in the directions of maximum curvature. The edge lengths of the resulting metric (dashed line) are limited to satisfy the minimum edge lengths of both the original metrics.

Similarly minimum element sizes can be imposed by finding the smallest ellipsoid that envelopes the combined field metric and a minimum metric:

$$M_{\min} = \operatorname{diag}\left(\frac{1}{\sqrt{\hat{l}_{1\min}}}, \ldots, \frac{1}{\sqrt{\hat{l}_{d\min}}}\right). \tag{2.59}$$

Furthermore the anisotropy can be constrained by limiting the aspect ratio, a, of the eigenvalues:

$$\lambda_i = \max\left(\lambda_i, \frac{1}{a^2}\max_j \lambda_j\right). \tag{2.60}$$

For example, in Figure 1.1(b) $a = 1$ ensuring that the mesh optimisation drives the discretisation of the domain towards equilateral triangles with an edge length prescribed by the smallest interpolation error.

Using the combined metric, averaged across each element, the suitability of a given mesh for any combination of fields can be determined. Generally the lowest quality element is used to determine the overall mesh quality and to decide if it requires optimisation.

This process can be repeated at specified intervals to ensure that the mesh is maintained as close to optimal as possible throughout the simulation. Such dynamic adaptivity allows the full benefits of unstructured meshes to be exploited in multi-material simulations, focusing resolution around and along interfaces and other regions of dynamic significance.

2.4 Interpolation

Repeated mesh optimisation ensures that an interface can move through a domain while being represented on as few nodes as possible. In other words high resolution is unnecessary except in a limited area surrounding the interface (or other features of interest). It is therefore unnecessary to place high resolution everywhere in anticipation of the interface's potential movements.

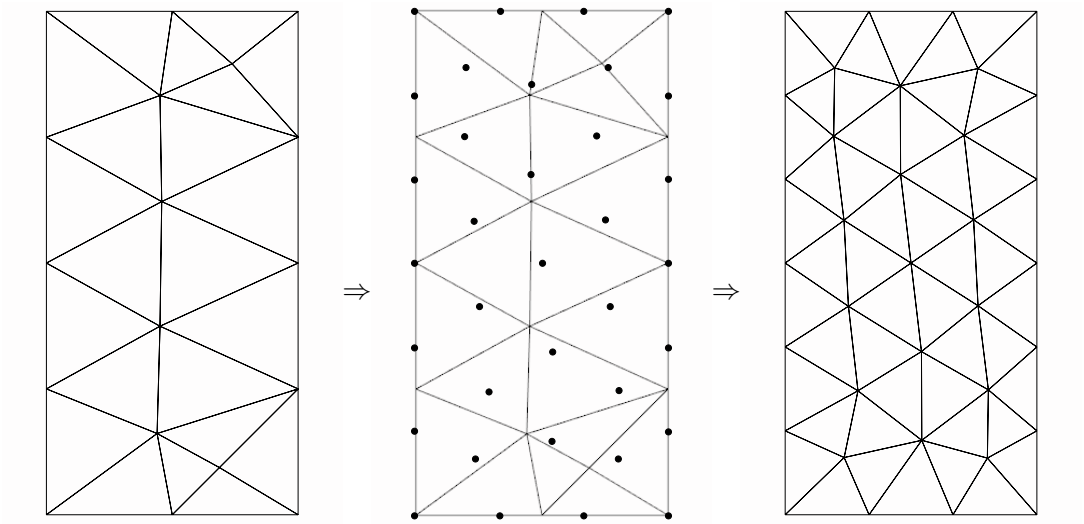


Figure 2.7: Consistent interpolation for a piecewise linear continuous field. Field data are interpolated from mesh I onto the nodal points of mesh II.

After a step of mesh optimisation however two meshes exist: the old mesh (referred to here as mesh I) and a new mesh (mesh II). Mesh I was designed to be optimal for the conditions that existed a certain number of timesteps ago, when the last mesh optimisation was performed, and contains up to date information about all the fields at the current time. Mesh II has just been created to be optimal for the current condition of the fields. Other than describing the same domain, Ω, the meshes will in general be completely unrelated to each other. However, it is necessary to transfer the field information from mesh I to mesh II. This allows solutions to be found on mesh II for a further interval of timesteps until the next step of mesh optimisation takes place. Two options exist for transferring this information: consistent interpolation and projection.

Consistent interpolation transfers information between mesh I and mesh II using the underlying basis functions of each field (see Figure 2.7). The locations of the solution nodes of mesh II are found in mesh I using the R-tree search spatialindex library (Manolopoulos *et al.*, 2006). Field values can then be interpolated from the solution nodes of mesh I to these locations using the field basis functions of mesh I as interpolants.

Consistent interpolation is simple, fast and results in a bounded field. However, it is not conservative. The integral of a field over the whole domain on mesh I will not be preserved on mesh II. Furthermore, due to the co-location of solution nodes on a discontinuous mesh, a discontinuous field on mesh I will be smoothed during the transfer to mesh II. In other words all co-located nodes will share the same value.

Projecting rather than interpolating between the meshes offers some important improvements. For instance it is possible to conserve the integral of a field during the transfer. Considering a field α represented on mesh I, α^{I}, and on mesh II, α^{II}, then, in a

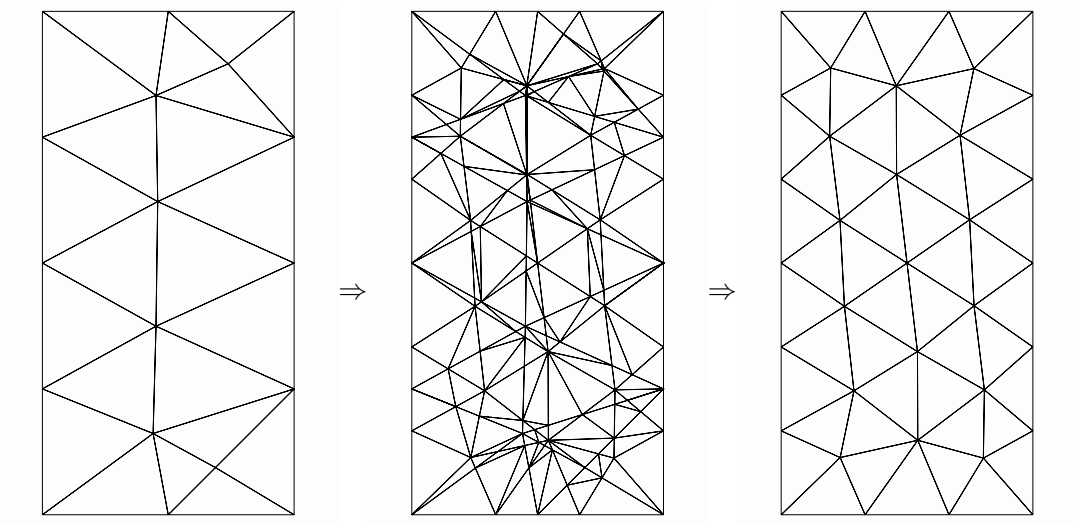


Figure 2.8: Projection using the supermesh for a piecewise linear continuous field. Field data are projected from mesh I onto mesh II via the supermesh, which contains all the vertices and edges of both mesh I and II.

weak sense, the integral:

$$\int_\Omega \omega^{\mathrm{II}} \alpha^{\mathrm{II}} dV = \int_\Omega \omega^{\mathrm{II}} \alpha^{\mathrm{I}} dV, \tag{2.61}$$

where ω^{II} is a test function on mesh II, should be preserved. Replacing α^{I}, α^{II} and ω^{II} with their discrete counterparts $N_A^{\mathrm{I}}\alpha_A^{\mathrm{I}}$, $N_A^{\mathrm{II}}\alpha_A^{\mathrm{II}}$ and $N_A^{\mathrm{II}}\omega_A^{\mathrm{II}}$ respectively, Equation 2.61 can be written:

$$\mathbf{M}^{\mathrm{II}}\alpha^{\mathrm{II}} = \mathbf{M}^{\mathrm{III}}\alpha^{\mathrm{I}} \tag{2.62}$$

where

$$\mathbf{m}^{\mathrm{II}}{}_{ab} = \int_{\Omega_e} N_a^{\mathrm{II}} N_b^{\mathrm{II}} \mathrm{d}V \qquad \Rightarrow \mathbf{M}^{\mathrm{II}} \qquad \text{mesh II mass matrix} \tag{2.63}$$

$$\mathbf{m}^{\mathrm{III}}{}_{ab} = \int_{\Omega_e} N_a^{\mathrm{II}} N_b^{\mathrm{I}} \mathrm{d}V \qquad \Rightarrow \mathbf{M}^{\mathrm{III}} \qquad \text{mixed mass matrix.} \tag{2.64}$$

$$\tag{2.65}$$

Therefore a conservative representation of α can be found on mesh II by solving Equation 2.62 for α^{II} (Farrell *et al.*, 2009).

Solving Equation 2.62 however requires the construction of a mixed mass matrix, $\mathbf{M}^{\mathrm{III}}$ which contains both the basis functions of mesh I, N_A^{I}, and the basis functions of mesh II, N_A^{II}. Assembly of this matrix requires the construction of a so-called supermesh, containing all the nodes and edges of both mesh I and II (this is illustrated in Figure 2.8). A supermesh is capable of exactly representing both sets of basis functions.

Conservative projection of piecewise continuous fields through the supermesh can be thought of as a consistent interpolation followed by a projection. The supermesh contains

both the nodes from mesh I, which have known values of the fields located at them, and the nodes from mesh II, which have no data associated with them. However, the values at the nodal points belonging to mesh II can be interpolated on the supermesh using the basis functions from mesh I. As the supermesh also contains the nodes of mesh I no data are deleted and this process is conservative. The unwanted nodes are then removed when the projection takes place from the supermesh to mesh II.

In reality the intermediary interpolation stage on the supermesh can be skipped. Given a mapping from an element of mesh II to its child elements within an element of mesh I on the supermesh, the elemental mixed mass matrix, $\mathbf{m}^{\mathrm{III}}{}_{ab}$ can be assembled without the need to construct the entire supermesh. Instead just the intersections between these elements are found using Wild Magic Geometric Tools library (Eberly, 2001) and the Computational Geometry Algorithms library (CGAL, Overmars, 1996; Fabri *et al.*, 2000). Furthermore since the mixed mass matrix is only ever multiplied by the field on mesh I, α^{I}, it never needs to be globally assembled. Instead the action of its elemental components on α^{I} may be computed during assembly.

These optimisations reduce the cost of projection between the meshes. However, the creation of the map from mesh II to the supermesh and finding the solution to Equation 2.62 mean it is still more expensive than consistent interpolation. Additionally, whilst projection is conservative, preserving the integral of a field, α^{II} is not necessarily bounded as with consistent interpolation.

For piecewise linear continuous fields bounding of the solution can be enforced by replacing $\mathbf{M}^{\mathrm{II}}$ in Equation 2.62 with its lumped mass equivalent, $\mathbf{M}^{\mathrm{II}^L}$:

$$\mathbf{M}^{\mathrm{II}}{}^L_{ii} = \sum_j \mathbf{M}_{\mathrm{II}ij}, \tag{2.66}$$

$$\mathbf{M}^{\mathrm{II}}{}^L_{ij} = 0, \forall i \neq j. \tag{2.67}$$

Lumping the mass matrix in this manner has no effect on the conservation properties of the projection, however it does guarantee that the solution is bounded (Farrell *et al.*, 2009). The disadvantage of lumping the mass matrix is that it adds an artificial numerical diffusion to the solution (Zienkiewicz & Taylor, 2000b).

Farrell *et al.* (2009) proposed a minimally dissipative correction for the projection method to ensure boundedness for piecewise linear continuous fields. First the unbounded solution, α^{II}, is found by solving Equation 2.62. Assuming an envelope of bounded values between $\alpha^{\mathrm{II}}_{\min}$ and $\alpha^{\mathrm{II}}_{\max}$ is known then the deviation of α^{II} from boundedness, $\alpha^{\mathrm{II}}_{\mathrm{dev}}$, is defined at each node using:

$$\alpha^{\mathrm{II}}_{\mathrm{dev}} = \begin{cases} \alpha^{\mathrm{II}} - \alpha^{\mathrm{II}}_{\max} & \text{if } \alpha^{\mathrm{II}} > \alpha^{\mathrm{II}}_{\max}, \\ \alpha^{\mathrm{II}} - \alpha^{\mathrm{II}}_{\min} & \text{if } \alpha^{\mathrm{II}} < \alpha^{\mathrm{II}}_{\min}, \\ 0 & \text{if } \alpha^{\mathrm{II}}_{\min} < \alpha^{\mathrm{II}} < \alpha^{\mathrm{II}}_{\max}. \end{cases} \tag{2.68}$$

A bounded field would therefore have $\alpha^{\mathrm{II}}_{\mathrm{dev}} = 0$ everywhere.

In this project the bounds, $\alpha^{\mathrm{II}}_{\min}$ and $\alpha^{\mathrm{II}}_{\max}$, were found using the bounded lumped mass solution to Equation 2.62, $\alpha^{\mathrm{II}}_{\mathrm{lump}}$. Given a node A in mesh II, $\alpha^{\mathrm{II}}_{\min A}$ was defined as

being the minimum value of $\alpha^{\mathrm{II}}_{\mathrm{lump}}$ at the nodes directly connected to A. Similarly $\alpha^{\mathrm{II}}_{\mathrm{max}A}$ was defined as the maximum value of $\alpha^{\mathrm{II}}_{\mathrm{lump}}$ at the nodes surrounding A. Using Equation 2.68 these bounds provide an envelope of values around the lumped mass solution, $\alpha^{\mathrm{II}}_{\mathrm{lump}}$, within which the actual solution, α^{II}, is considered bounded.

Using the deviation, $\alpha^{\mathrm{II}}_{\mathrm{dev}}$, a new field, $\alpha^{\mathrm{II}}_{\mathrm{alt}}$, is then found by solving:

$$\mathbf{M}^{\mathrm{II}^L}\alpha^{\mathrm{II}}_{\mathrm{alt}} = \mathbf{M}^{\mathrm{II}}\alpha^{\mathrm{II}}_{\mathrm{dev}}. \tag{2.69}$$

This equation exploits the diffusive properties of the lumped mass matrix. In effect $\alpha^{\mathrm{II}}_{\mathrm{alt}}$ is a smeared version of the deviation, $\alpha^{\mathrm{II}}_{\mathrm{dev}}$. However as the lumped mass matrix shares the conservation properties of the full mass matrix the integral of the deviation is unchanged. Hence it is possible to update the solution conservatively such that:

$$\alpha^{\mathrm{II}} \leftarrow \alpha^{\mathrm{II}} - \alpha^{\mathrm{II}}_{\mathrm{dev}} + \alpha^{\mathrm{II}}_{\mathrm{alt}}. \tag{2.70}$$

Solving Equation 2.69 repeatedly, each time updating the solution field, α^{II}, and recalculating $\alpha^{\mathrm{II}}_{\mathrm{dev}}$, moves the unboundedness around the domain. Locality of this movement is maintained through the sparsity structure of the mass matrix, which ensures that at each iteration information only propagates to the nearest neighbour of a node. When a deviation is smeared into nodes with α^{II} values that lie between the bounds and that have sufficient capacity to absorb the deviation then it is set to zero and is considered bounded. Once this occurs at all nodes the iterations terminate.

As the lumped mass matrix is diagonal it can be inverted directly so each iteration involves only a matrix-vector multiplication, an array division and two vector additions. Farrell *et al.* (2009) found that approximately 1,000 iterations were required to bound a solution satisfactorily.

In summary therefore, it is possible to transfer field data between an old mesh and a newly optimised mesh using a variety of methods with a range of properties:

- consistent interpolation, which is bounded but non-conservative;
- Galerkin projection, which is conservative but not generally bounded;
- lumped projection, which is conservative and bounded but diffusive and only applies to piecewise linear continuous fields;
- bounded projection, which is conservative, bounded and minimally dissipative but only applies to piecewise linear continuous fields.

Selection between the different methods will therefore depend on the discretisation and physical meaning of the field in question.

2.5 Summary

In this chapter the fundamental equations describing geophysical flows have been presented along side suitable methods for discretising them on arbitrarily unstructured

meshes. It has also been shown how these discrete equations are assembled and solved numerically and how a given domain can be tessellated optimally for their solutions.

The suitability of these methods for modelling multiple material flows will be the main focus of the following chapters. In the next chapter advection methods that can be used to maintain sharp discontinuities in material properties will be developed. In Chapter 4 this work will be linked to the Navier-Stokes equations to form a fully coupled system of equations. Finally this will be expanded into a system for arbitrary numbers of materials in Chapter 5.

Chapter 3

Material Tracking

The ability to differentiate between regions with differing material properties is of fundamental importance in the modelling of many physical systems. In the most general numerical scheme each set of properties is referred to as a phase and each phase is described using a full system of momentum and field equations (Equations 2.1 and 2.11). Interaction between phases is introduced through a common pressure field or extra linking terms in the momentum equation, such as inter-phase drag (Neri *et al.*, 2003; Brennen, 2005). In such a *multi-phase* model, phases are described over the entire domain and are allowed to inter-penetrate (subject to local volume or mass constraints). This allows the approximation of dilute suspensions not resolved by the mesh resolution. However, if individual phases are concentrated in specific regions then describing their behaviour over the entire domain is computationally inefficient. Instead, in a *multi-material* model, the phases are considered to be immiscible materials separated by a sharp interface.

In a multi-material approach the properties for any subsection of the domain will be derived from a single material. Hence, it makes sense that only a single velocity field is used over the entire domain. This velocity is found by solving the momentum equation once using a bulk constitutive law and equation of state (Equations 2.2 and 2.3). In the case of an isotropic and isothermal Newtonian fluid this means that at any point the density and viscosity incorporated into Equation 2.7 are taken to be an average of the properties of the materials in the vicinity of that point. In regions containing only one material the local dynamics are therefore controlled by its density and viscosity. In the vicinity of the interface however the local dynamics may be dependent on an average of the properties of more than one material.

The grid point spacing of a numerical simulation defines the spatial scale on which the interface is defined and on which the averaging of different material properties occurs. In most geophysical flow models this is generally larger than the length-scale on which physical (molecular) mixing occurs around an interface. Therefore the behaviour of the bulk fluid in these regions is questionable. Standard advection algorithms on a fixed mesh tend to add numerical diffusion, smearing out the interface further. Material tracking methods encompass a broad spectrum of numerical tools which aim to minimise these mixed cells, and hence more accurately represent multi-material flows.

The simplest mechanism to represent regions with distinct material properties dis-

cretely is to define a mesh that incorporates the interfaces. The mesh must then be deformed as the simulation progresses to ensure that the material boundaries are always represented in the tessellation. Such a surface fitting, or Lagrangian, model is however unsuitable for simulating processes involving large degrees of deformation. Modelling less viscous materials leads to significant relative motion between the grid points, which may tangle the mesh. This violates the requirement that the elements have non-zero length, area or volume and so the mesh becomes unsuitable (Zhao *et al.*, 2002; Belytschko *et al.*, 2000). Mesh tangling may be avoided if the mesh is left undeformed and some sort of marker is used on to represent the position of the materials on the mesh. Such Eulerian approaches may be broadly characterised by whether the marker used is a field or not.

Non-field based material tracking systems include particle based methods, where a set of unconnected particles are advected through the fixed mesh in a Lagrangian manner (Daly *et al.*, 1966; Sulsky *et al.*, 1995). These particles contain material information including a temporally invariant colour function indicating to which material each particle belongs. The location and local relative densities of these particles can then be used to calculate bulk material properties for each cell in the mesh. However, in complex flows there is no guarantee that such particles will remain closely packed enough to clearly define the boundary location, thus leading to the formation of mixed material cells. This can be avoided by dynamically changing the number of particles but this may become computationally expensive, especially in three dimensions (Zhao *et al.*, 2002), and potentially introduces conservation errors.

Rather than advecting particles representing the location of the materials themselves, it is also possible to advect lines or surfaces representing the location of the interface between different materials. Such contour advection algorithms advect lower dimensional objects in a Lagrangian manner around a fixed mesh (Dritschel *et al.*, 1999). The location of these contours can then be used to construct the bulk properties while preventing the region of mixed cells extending beyond one unit cell width around the interface. While this minimises the problem of mixed cells it also introduces new difficulties. For instance, with overturning flows the contour advection algorithm must be able to merge the boundaries while minimising conservation errors.

Field-based methods overcome these difficulties by representing the colour functions that distinguish different materials as continuous fields over the entire domain. Such methods typically assign a physical meaning to the colour function, such as the fraction (by mass or volume) of material in a particular cell. The volume or mass fractions can then be used to compute the bulk material properties. These fields are advected around the domain according to Equation 2.11 and their values used to delineate regions containing different materials. Individual movements of the field may be on a scale smaller than the grid spacing. Hence, all field advection algorithms introduce some numerical diffusion which smears sharp interfaces thereby increasing the number of mixed material cells.

Level-set methods overcome this difficulty by defining a smooth colour function, which typically defines the distance from the interface (i.e. not a sharply varying volume fraction field). The interface between the materials is pinned to a certain level in the smooth field

(Sussman *et al.*, 1994; Hu & Adams, 2006). This interface position is used to calculate the bulk material properties as in a contour tracking scheme. Excessive diffusion however can still cause non-physical interface movements. Therefore the colour function must be reset or adjusted to control the interface position (Quecedo *et al.*, 2004). This process however is not conservative and requires an extra correction step (Zahedi *et al.*, 2009; Zhang *et al.*, 2009).

The volume of fluids (VOF) family of methods (Hirt & Nichols, 1981; Rudman, 1997) use a volume fraction to reconstruct the interface between different materials. Methods for this are varied and may be complex. Typically, the volume fraction gradient is used to determine the interface orientation while the volume fraction in the cell is used to find its position (Youngs, 1982; Rudman, 1997). The intersection of the interface and the surrounding mesh can then be used to define a flux with which to advect the volume fraction whilst minimising numerical diffusion in a finite volume discretision (see Sections 2.2.2 and 3.3).

As with particle tracking schemes such field-based interface reconstruction becomes more complicated on unstructured, adaptive meshes. Rather than considering just one standard shaped cell with identical, structured neighbours it is necessary to construct the interface or locate the particle in arbitrarily shaped and connected cells, which change after every mesh optimisation. To overcome this, the development of a flux limited field-based material advection method for use on unstructured meshes is described in this chapter. The use of flux limiting means that interface reconstruction and particle tracking are not required, considerably simplifying the scheme.

3.1 Field Advection

Field-based methods of tracking different materials that do not entail interface construction have several advantages over non field-based methods. Firstly, the field may be assigned a useful physical meaning. Here the field is used to describe the fractional volume occupied by a material in a particular cell. Hence all materials are represented by volume fractions which individually may vary between 0 and 1, but collectively must sum to 1 across the domain. Thus the location of K materials may be uniquely described by $K-1$ volume fractions as, at any particular location:

$$\alpha^K = 1 - \sum_{k=1}^{K-1} \alpha^k, \tag{3.1}$$

where α^k is the volume fraction field for material k. This definition is particularly useful as it allows for the easy calculation of bulk material properties for use in the global momentum equations (2.7). For example, considering density at any location in the domain:

$$\rho = \alpha^k \rho^k \tag{3.2}$$

where ρ^k is the density of material k.

Interface reconstruction or particle tracking is generally the most demanding part of material advection, particularly on unstructured grids or in three dimensions (Zhao

et al., 2002). Avoiding this expense, using field-based methods, potentially reduces the computational cost both of running calculations and of implementation. The latter may also be lowered by building on existing advection methods in models. Unlike interface reconstruction and particle tracking, field-based methods offer additional flexibility by allowing diffusional processes to be incorporated should the requirement of maintaining completely immiscible materials be relaxed. Finally, if consistent discretisations are used for the momentum and material advection steps, field-based methods ensure conservation with no additional intervention (see Chapter 4).

The suitability of a field-based method is highly dependent on the advection method used. The advection must be strictly bounded as a volume fraction greater than 1 or less than 0 makes no physical sense and, through Equations 3.1 and 3.2, may produce negative bulk densities causing Equations 2.7 to become ill-posed. Additionally, the change in volume fraction from 1 to 0 should occur over as small a number of cells as possible. Minimising this zone minimises the number of mixed nodes and the uncertainty in the position of the interface, which though not reconstructed is represented by this change in the volume fraction.

Of the available discretisation methods – finite volume, finite element and finite difference – finite volume and finite element are most suited to unstructured meshes and complex domains. Continuous Galerkin finite element methods are unsuited to advection dominated problems, producing oscillatory results (Zienkiewicz & Taylor, 2000a). Hence they require stabilisation, generally taking the form of an extra diffusion operator (Donéa & Huerta, 2003) to smooth out these oscillations. As this leads to an increased number of mixed cells this discretisation method is ruled out. Discontinuous Galerkin finite element methods are better suited to advection problems as the upwinded surface fluxes used to couple the elements together tend to prevent the growth of non-physical oscillations. They are however still locally unbounded and require limiting or smoothing (Cockburn & Shu, 1998; Qiu & Shu, 2004), which again tends to increase the number of mixed cells. Only finite volume methods can guarantee a physical, bounded solution. However, as discussed in Section 2.2.2 the only implicit method of selecting the surface integral face value that satisfies these requirements is first-order upwinding, which also introduces significant amounts of dissipation (LeVeque, 1992). It is therefore necessary to consider higher-order limited schemes for selecting the face value.

3.2 One-Dimensional Limiting

Flux limiting is a process through which finite volume schemes attempt to predict whether surface fluxes will produce non-physical oscillations and, if so, correct them such that the solution remains bounded. Three things are therefore necessary – a definition of what is meant by boundedness, a method of detecting non-physical oscillations in the solution and a method of correcting the surface fluxes to prevent and to repair any unwanted behaviour.

For simplicity, this problem is initially considered as advection in a single dimension,

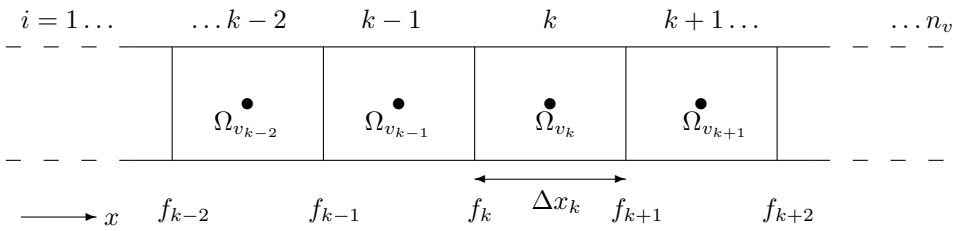


Figure 3.1: Section of the discretised one-dimensional line showing the indexing of the volumes and faces as well as the cell spacing.

along a line in the x direction:

$$\alpha_{,t} + (u\alpha)_{,x} - \alpha u_{,x} = 0. \tag{3.3}$$

This line is discretised into a series of n_v volumes with width Δx_i, $i = 1, \ldots, n_v$. Each volume, Ω_{v_i}, is contained by its boundary $\partial\Omega_{v_i}$, which is split into faces f_i and f_{i+1} separating it from its neighbours, $\Omega_{v_{i-1}}$ and $\Omega_{v_{i+1}}$ respectively (see Figure 3.1).

With this simplified tessellation the discretised Equation 2.35 for cell i becomes:

$$\begin{aligned}
&\alpha_i^{n+1} \underbrace{- \theta_p \sum_{l=0}^{1} c_{i_{f_{i+l}}} I_{i_{f_{i+l}}} \alpha_{i-1+2l}^{n+1} + \theta_p \sum_{l=0}^{1} c_{i_{f_{i+l}}} \left(1 - I_{i_{f_{i+l}}}\right) \alpha_{i-1+2l}^{n+1}}_{\text{implicit low-order advective flux term}} \\
&\underbrace{- \theta \left(\sum_{l=0}^{1} c_{i_{f_{i+l}}} \left(1 - I_{i_{f_{i+l}}}\right) - \sum_{l=0}^{1} c_{i_{f_{i+l}}} I_{i_{f_{i+l}}} \right) \alpha_i^{n+1}}_{\text{implicit velocity divergence term}} = \alpha_i^n + \\
&\underbrace{\sum_{l=0}^{1} c_{i_{f_{i+l}}} I_{i_{f_{i+l}}} \left(\theta \hat{\tilde{\alpha}}|_{f_{i+l}}^{HO} + (1-\theta)\hat{\alpha}^n|_{f_{i+l}}^{HO}\right) - \sum_{l=0}^{1} c_{i_{f_{i+l}}} \left(1 - I_{i_{f_{i+l}}}\right) \left(\theta \hat{\tilde{\alpha}}|_{f_{i+l}}^{HO} + (1-\theta)\hat{\alpha}^n|_{f_{i+l}}^{HO}\right)}_{\text{high-order advective flux term}} \\
&\underbrace{- \theta_p \sum_{l=0}^{1} c_{i_{f_{i+l}}} I_{i_{f_{i+l}}} \tilde{\alpha}_{i-1+2l} + \theta_p \sum_{l=0}^{1} c_{i_{f_{i+l}}} \left(1 - I_{i_{f_{i+l}}}\right) \tilde{\alpha}_{i-1+2l}}_{\text{nonlinear low-order advective flux term}} \\
&\underbrace{+ (1-\theta) \left(\sum_{l=0}^{1} c_{i_{f_{i+l}}} \left(1 - I_{i_{f_{i+l}}}\right) - \sum_{l=0}^{1} c_{i_{f_{i+l}}} I_{i_{f_{i+l}}} \right) \alpha_i^n}_{\text{explicit velocity divergence term}}
\end{aligned} \tag{3.4}$$

where $c_{i_{f_l}}$ is the Courant number of node i associated with face l:

$$c_{i_{f_l}} = \frac{|\tilde{u}|_{f_l}|\Delta t}{\Delta x_i}, \tag{3.5}$$

and $I_{i_{f_l}}$ indicates whether the velocity flux is incoming to node i across face l:

$$I_{i_{f_l}} = \begin{cases} 0 & \text{if } \tilde{u}|_{f_l} n_{i_{f_l}} > 0, \\ 1 & \text{if } \tilde{u}|_{f_l} n_{i_{f_l}} < 0, \end{cases} \tag{3.6}$$

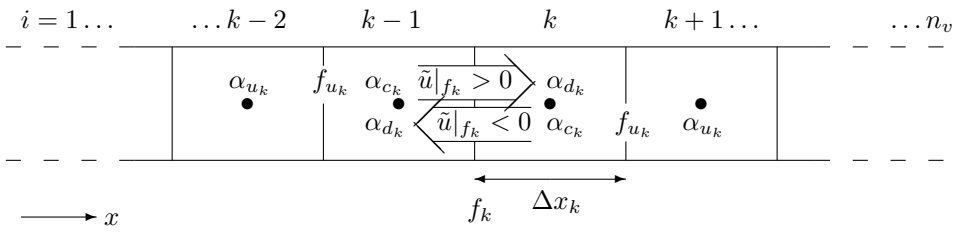


Figure 3.2: Section of the discretised line illustrating how the upwind, α_{u_k}, donor, α_{c_k}, and downwind, α_{d_k}, nodal values and the upwind face, f_{u_k}, are positioned around the face f_k depending on the direction of the velocity at the face, $\tilde{u}|_{f_k}$.

where $n_{i_{f_l}}$ is the unit normal to face l pointing away from node i, such that $n_{i_{f_i}} = -1$ and $n_{i_{f_{i+1}}} = 1$.

The direction of the flux can also be used to define generic relationships between a face, say f_k, and the nodes surrounding it. The nodal value to which the flux across f_k is incoming is referred to as the downwind value, α_{d_k}, while its neighbour across f_k is the donor or central value, α_{c_k}. A final node neighbouring α_{c_k} across the upwind face, f_{u_k}, further upstream to f_k is called the upwind node α_{u_k}. These relationships are illustrated in Figure 3.2.

3.2.1 Sensor Functions: Detecting Non-Physical Oscillations

The relative values between the downwind, donor and upwind nodes can be used to determine if a solution around a particular face is locally smooth:

$$\alpha_{u_k} > \alpha_{c_k} > \alpha_{d_k}, \tag{3.7a}$$

$$\alpha_{d_k} > \alpha_{c_k} > \alpha_{u_k}, \tag{3.7b}$$

or contains a local extremum:

$$\alpha_{c_k} \geq \alpha_{u_k} \geq \alpha_{d_k}, \tag{3.8a}$$

$$\alpha_{c_k} \leq \alpha_{u_k} \leq \alpha_{d_k}, \tag{3.8b}$$

$$\alpha_{c_k} \geq \alpha_{d_k} \geq \alpha_{u_k}, \tag{3.9a}$$

$$\alpha_{c_k} \leq \alpha_{d_k} \leq \alpha_{u_k}. \tag{3.9b}$$

Defining the normalised variable (Gaskell & Lau, 1988) of the donor value as:

$$\bar{\alpha}_{c_k} = \frac{\alpha_{c_k} - \alpha_{u_k}}{\alpha_{d_k} - \alpha_{u_k}}, \tag{3.10}$$

Equation 3.7 can be rewritten as:

$$0 < \bar{\alpha}_{c_k} < 1, \tag{3.11}$$

while Equations 3.8 and 3.9 become:

$$\bar{\alpha}_{c_k} \geq 1 \text{ and} \tag{3.12a}$$

$$\bar{\alpha}_{c_k} \leq 0. \tag{3.12b}$$

Thus the discrete solution can be considered smooth when the normalised donor value lies between 0 and 1 (see Figure 3.3(a)).

A number of other extremum detectors can also be used. For instance Sweby (1984) suggested the ratio of the gradients evaluated at the face f_k and at the upwind face, f_{u_k}:

$$r = \frac{\frac{\partial \alpha}{\partial x}|_{f_k}}{\frac{\partial \alpha}{\partial x}|_{f_{u_k}}} = \frac{a}{b}. \tag{3.13}$$

Discretely a and b can be represented as:

$$a = \frac{\alpha_d - \alpha_c}{\frac{1}{2}\left(\Delta x_{d_k} + \Delta x_{c_k}\right)},$$
$$b = \frac{\alpha_c - \alpha_u}{\frac{1}{2}\left(\Delta x_{c_k} + \Delta x_{u_k}\right)}.$$

If the cell spacing, Δx_i, is constant a simple expression converting normalised variables to the Sweby gradient operator can be derived as:

$$r = \frac{1 - \bar{\alpha}_{c_k}}{\bar{\alpha}_{c_k}}. \tag{3.14}$$

The solution is considered to be smooth for $r > 0$ while a local extremum is detected when $r \leq 0$ (see Figure 3.3(b)).

Waterson (2008) suggested modifying the Sweby gradient function by making it symmetric around 0. This can be simply achieved by introducing a new extremum function, s, such that:

$$s = \frac{r - 1}{r + 1} = 1 - 2\bar{\alpha}_{c_k}. \tag{3.15}$$

The solution is unbounded when $|s| \geq 1$ and smooth when $|s| < 1$ (see Figure 3.3(c)). Therefore, on a structured one-dimensional mesh, all three of the sensor functions are equivalent (see Figure 3.3).

3.2.2 Face Value Functions: Defining Boundedness

The sensor functions provide a simplified method (reducing six possibilities to three) for determining if the solution is locally smooth around a face. While this is clearly an *a posteriori* estimate of the field behaviour it can still be used to determine how potential face values, $\alpha|_{f_k}^{HO}$, should be treated.

For instance, if a definition of boundedness is selected such that the solution should always tend toward monotonicity, then clearly the face values around an extremum should be treated in a manner that will reduce the size of the local oscillation. Such boundedness criteria are referred to as total variation diminishing (TVD). As first-order upwinding is the only guaranteed monotonic face value scheme it is generally selected for faces in the vicinity of extrema such that:

$$\hat{\alpha}|_{f_k}^{HO} = \alpha_{c_k}. \tag{3.16}$$

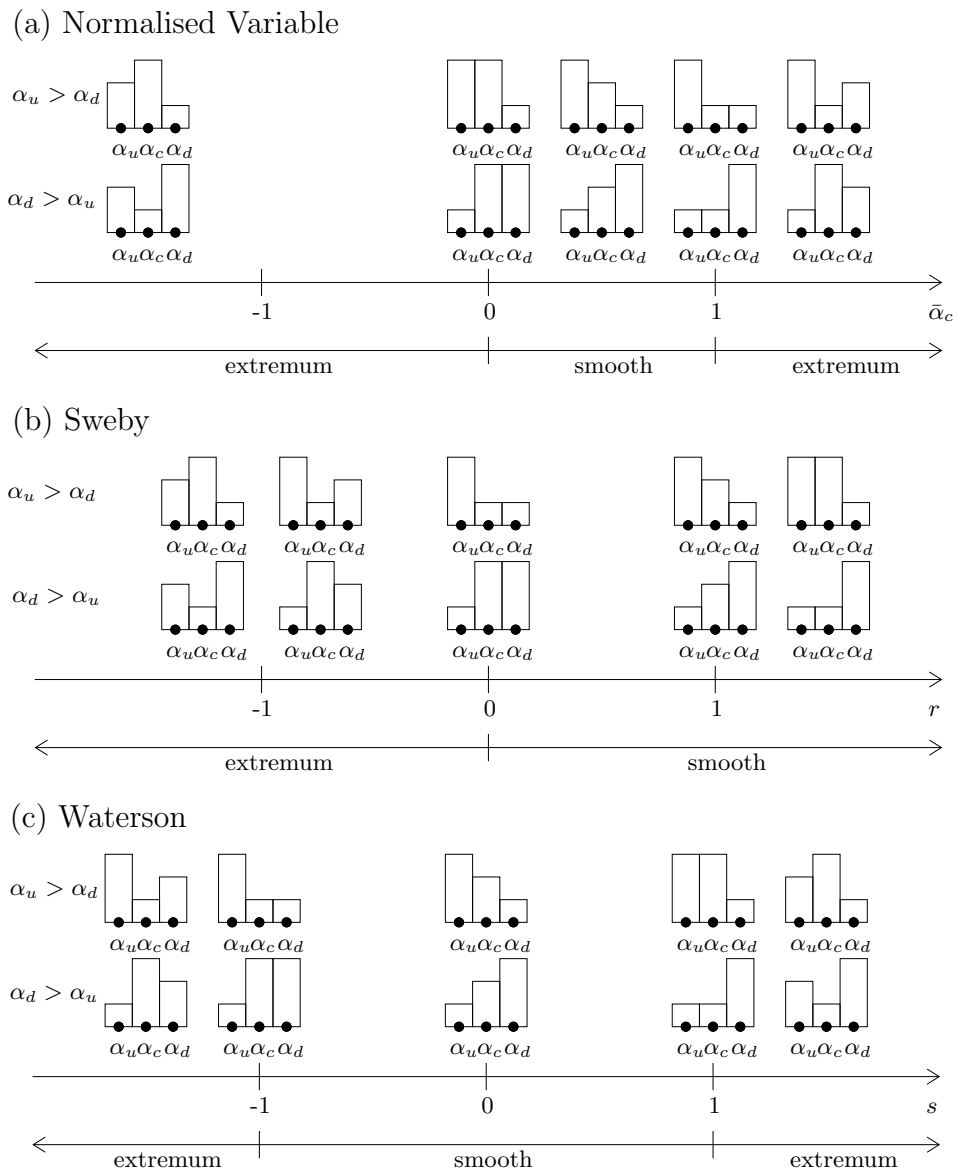


Figure 3.3: Extremum detection using (a) normalised variables, (b) the Sweby gradient function and (c) the Waterson gradient function.

There is more scope for selecting less diffusive schemes in smooth regions. However, ensuring that these prevent the introduction of new extrema is challenging. Physically realistic face values lie between the two values of the face's neighbours:

$$\alpha_{c_k} \leq \hat{\alpha}|_{f_k}^{HO} \leq \alpha_{d_k}, \tag{3.17a}$$

or

$$\alpha_{d_k} \leq \hat{\alpha}|_{f_k}^{HO} \leq \alpha_{c_k}. \tag{3.17b}$$

This can be represented graphically by introducing a new normalised variable for the field evaluated at the face f_k:

$$\bar{\alpha}|_{f_k}^{HO} = \frac{\alpha|_{f_k}^{HO} - \alpha_{u_k}}{\alpha_{d_k} - \alpha_{u_k}}, \tag{3.18}$$

reducing Equation 3.17 to:

$$\bar{\alpha}_{c_k} \leq \bar{\alpha}|_{f_k}^{HO} \leq 1, \tag{3.19}$$

within the smooth regions, $0 < \bar{\alpha}_{c_k} < 1$. This region can then be plotted on a normalised variable diagram (Gaskell & Lau, 1988) of normalised donor value, $\bar{\alpha}_{c_k}$, against normalised face value, $\bar{\alpha}|_{f_k}^{HO}$. In the smooth region, $0 < \bar{\alpha}_{c_k} < 1$, the range of potentially permissible face values can be seen shaded in Figure 3.4(a) while outside this area only first-order upwinding, $\bar{\alpha}|_{f_k}^{HO} = \bar{\alpha}_{c_k}$, is allowed.

Sweby (1984) represented this range of values on what has now become known as a Sweby Diagram, plotting the gradient ratio r, against the function:

$$\Psi = \frac{2\left(\alpha|_{f_k}^{HO} - \alpha_{c_k}\right)}{b\Delta x} = \frac{2\left(\bar{\alpha}|_{f_k}^{HO} - \bar{\alpha}_{c_k}\right)}{\bar{\alpha}_{c_k}}. \tag{3.20}$$

Using this face value function, the range of possible values falls within:

$$0 \leq \Psi \leq 2r, \tag{3.21}$$

in the smooth region, $r > 0$ (see Figure 3.4(b)).

Similarly, Waterson (2008) defined:

$$\Theta = (1 - s)\,\Psi\left(r\left(s\right)\right), \tag{3.22}$$

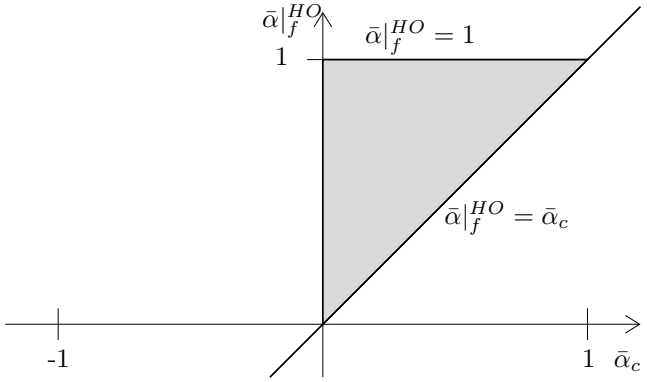
so that the possible range of face values falls within:

$$0 \leq \Theta \leq 2\left(1 + s\right), \tag{3.23}$$

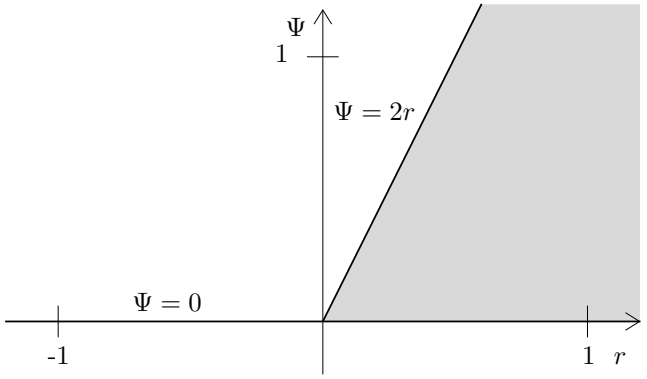
for $|s| < 1$ (see Figure 3.4(c)).

The region represented on these diagrams (see Figure 3.4) is referred to as the implicit convective boundedness criterion (CBC, Gaskell & Lau, 1988; Leonard, 1988; Jasak *et al.*, 1999). Unfortunately when used *a priori* it only gives the range of potentially valid face values but fails to predict which may lead to an unbounded solution.

(a) Normalised Variable Diagram (NVD)



(b) Sweby Diagram



(c) Waterson Diagram

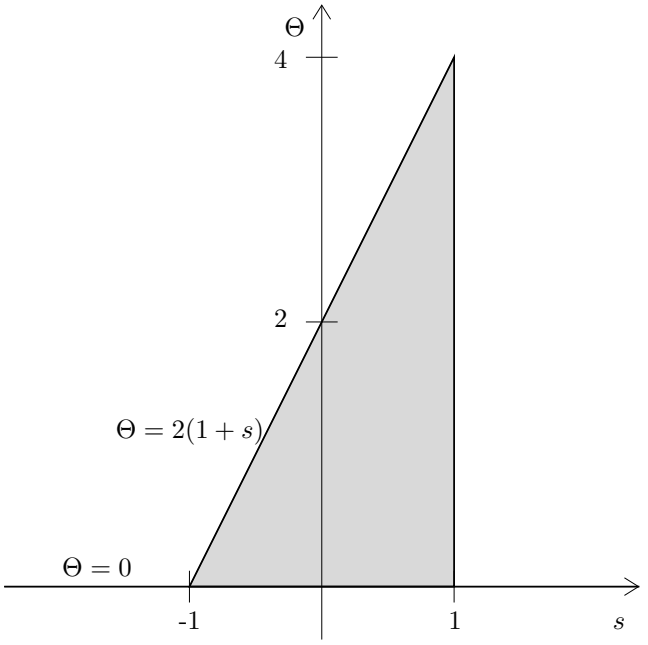


Figure 3.4: Potential range of bounded values represented on (a) a normalised variable diagram, (b) the Sweby diagram and (c) the Waterson diagram.

This can be demonstrated simply using a fully explicit, $\theta = 0$ and $\theta_p = 0$, constant velocity version of Equation 3.4:

$$\alpha_i^{n+1} = \alpha_i^n + \underbrace{\sum_{l=0}^{1} c_{i_{f_{i+l}}} I_{i_{f_{i+l}}} \hat{\alpha}^n|_{f_{i+l}}^{HO} - \sum_{l=0}^{1} c_{i_{f_{i+l}}} \left(1 - I_{i_{f_{i+l}}}\right) \hat{\alpha}^n|_{f_{i+l}}^{HO}}_{\text{high-order advective flux term}} \tag{3.24}$$

using the upper boundary of the CBC, first-order downwinding, in smooth regions and first-order upwinding otherwise:

$$\hat{\alpha}^n|_{f_k}^{HO} = \begin{cases} \alpha_{d_k}^n & \text{if } 0 < \bar{\alpha}_{c_k}^n < 1 \\ \alpha_{c_k}^n & \text{if } \bar{\alpha}_{c_k}^n \leq 0 \text{ or } \bar{\alpha}_{c_k}^n \geq 1. \end{cases} \tag{3.25}$$

Starting with a step function and considering the face, f_k, just ahead of the step (see Figure 3.5(a)), initially $\bar{\alpha}_{c_k}^n = 1$ so the face value $\hat{\alpha}^n|_{f_k}^{HO} = \alpha_{c_k}^n = 0$. Assuming a uniform Courant number of 2/5 across the domain, α_{c_k} gradually increases, first to 2/5 then to 4/5, due to fluxes across f_{u_k}. After each of these steps (see Figures 3.5(b, c)) the normalised donor value for face f_k, $\bar{\alpha}_{c_k}^n$, is 3/5 and 1/5 respectively. Therefore $\hat{\alpha}^n|_{f_k}^{HO} = \alpha_{d_k}^n = 0$ every time. However, as can be seen in Figure 3.5(d), this leads to α_{c_k} becoming unbounded. Therefore while the CBC gives a range of physically plausible face values it does not guarantee boundedness of the final solution.

3.2.3 Limiters: Correcting the Surface Fluxes

The fundamental problem with the implicit convective bounded criterion is that it is bounded by a discontinuity as $\bar{\alpha}_{c_k} \to 0$ ($r \to \infty$, $s \to 1$). In terms of the advection of a step function, illustrated in Figure 3.5, it does not return to first-order upwinding smoothly enough to prevent an overshoot. This has led to the concept of limiting the CBC to a smaller range of face values so that the discontinuity is removed from the boundary.

Sweby Limiter

Numerous variants of these limiting functions exist (Leonard, 1991; Darwish & Moukalled, 2003). For example, Sweby (1984) introduced the limited region shaded in Figure 3.6 for constant Courant number flows. The average or trapezoidal face value method (from Equation 2.33):

$$\alpha|_f^{HO} = \frac{\alpha_c + \alpha_d}{2} \tag{3.26}$$

is also shown in Figure 3.6 as a dashed line. Clearly, used by itself, this face value method is unbounded and would still have a discontinuity in its behaviour even if it were limited to the CBC. Instead, what the Sweby limiter suggests is that it should only be used within the shaded regions of Figure 3.6 and 'limited' back to first-order upwinding outside this zone:

$$\hat{\alpha}|_f^{HO} = \begin{cases} \frac{\alpha_c + \alpha_d}{2} & \text{if } \frac{1}{3} < \bar{\alpha}_{c_k}^n < 1 \\ 2\alpha_c - \alpha_u & \text{if } 0 < \bar{\alpha}_{c_k}^n < \frac{1}{3} \\ \alpha_c & \text{if } \bar{\alpha}_{c_k}^n \leq 0,\ \bar{\alpha}_{c_k}^n \geq 1. \end{cases} \tag{3.27}$$

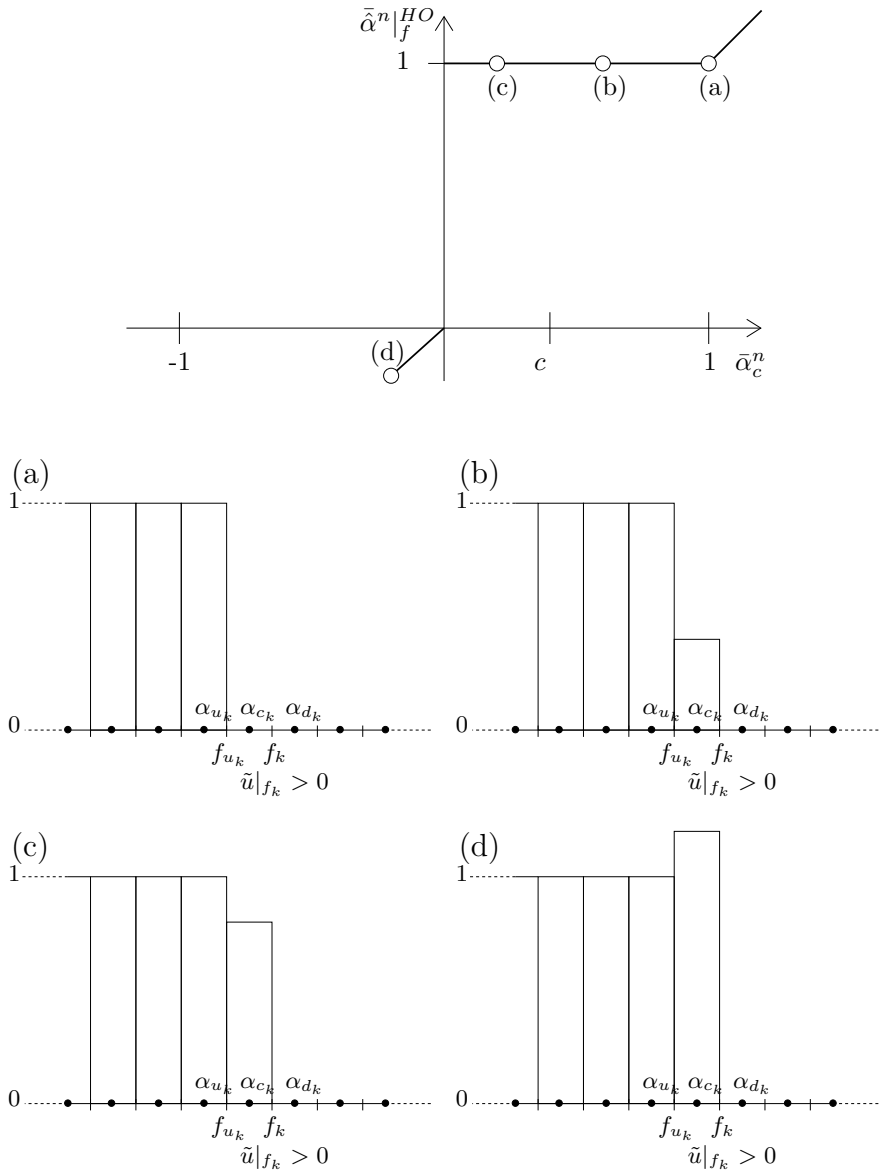
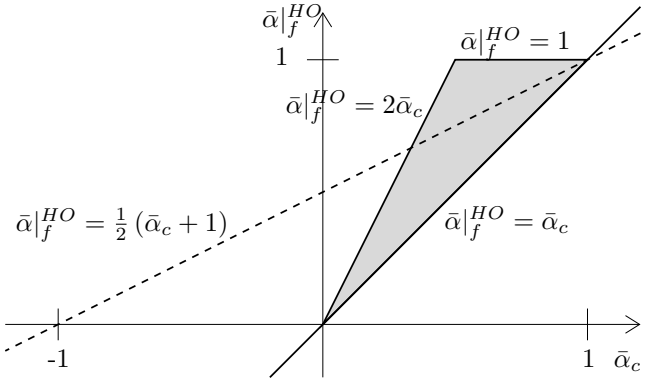
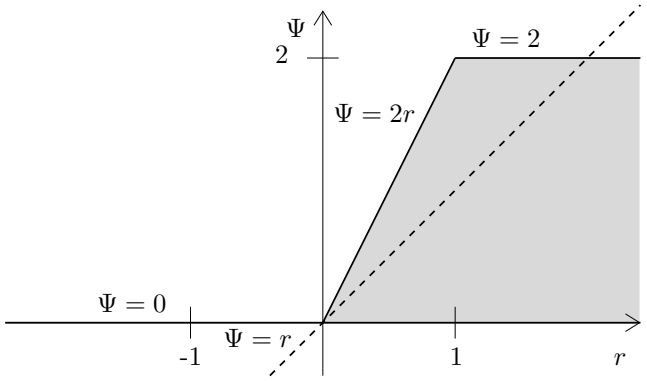


Figure 3.5: The advection of a step function using Equation 3.24 at a constant Courant number of 2/5. The face value is selected using the upper boundary of the convective boundedness criterion (i.e. first-order downwinding within smooth regions, first-order upwinding around extrema). Each timestep illustrated corresponds to a marked position on the normalised variable diagram above.

(a) Sweby Limiter - NVD



(b) Sweby Limiter - Sweby Diagram



(c) Sweby Limiter - Waterson Diagram

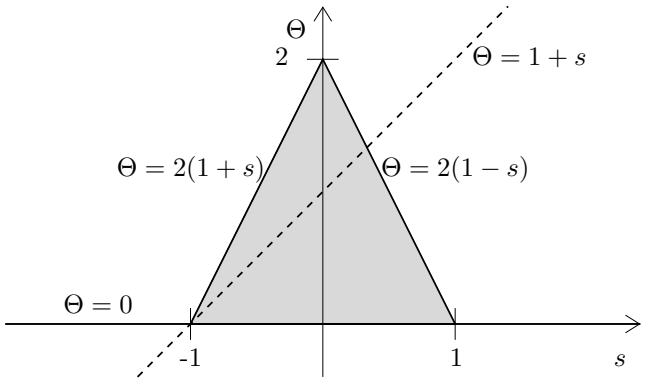


Figure 3.6: The Sweby limiter (shaded region) and the trapezoidal face value (dashed line) plotted on (a) a normalised variable diagram, (b) the Sweby diagram and (c) the Waterson diagram.

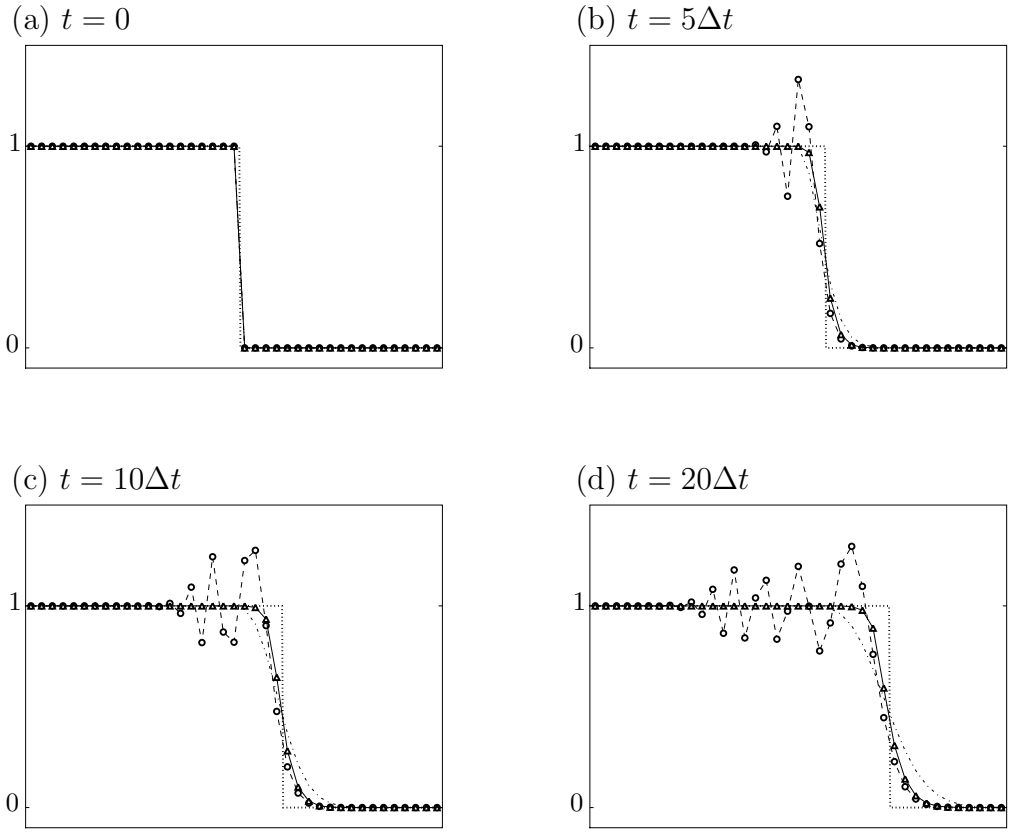


Figure 3.7: The advection of a step function using Equation 3.4 with $\theta = 1/2$, $\theta_p = 1$ at a constant Courant number of 2/5 with $\tilde{u} > 0$. A maximum of 15 iterations were used per timestep to achieve a maximum error (in the infinity norm) of 10^{-6} between the implicit and nonlinear pivot solutions. The high-order face value is determined using the trapezoidal method (circles, dashed line) and the 'Sweby limited' trapezoidal method (triangles, solid line). Four timesteps are shown: (a) the initial condition, (b) after 5 timesteps, (c) after 10 timesteps and (d) after 20 timesteps. For comparison the limited trapezoidal solution is bounded by the exact solution (dotted line) and the first-order upwinding solution (dot-dash line).

The solutions to Equation 3.4 using both the unlimited (Equation 3.26, circles, dashed line) and 'Sweby limited' (Equation 3.27, triangles, sold line) trapezoidal methods are shown in Figure 3.7 after 0, 5, 10 and 20 timesteps at a constant Courant number of 2/5. These solutions use $\theta = 1/2$ and $\theta_p = 1$ and use a maximum of 15 iterations per timestep to converge the implicit and nonlinear pivot solutions to an error of 10^{-6}.

Figure 3.7 clearly demonstrates the unbounded and oscillatory behaviour of the trapezoidal method and the improvement in the solution using limiting, which bounds the solution. Furthermore, the limited solution is less diffusive than the first-order upwind solution (dot-dash line). However, if the step function represented the change in volume fraction of a material around an interface then neither solution would be ideal. The unbounded step function clearly introduces non-physical negative volume fractions in the other material. The limited solution however is significantly more diffusive than the exact solution introducing several mixed cells and ambiguity about the position of the interface.

Courant Limited Downwinding

The key difference between the trapezoidal equation (3.26) and its limited counterpart (Equation 3.27) is that Equation 3.27 in some sense anticipates the filling up (or emptying) of the donor cell and increases (or decreases) the flux through a face appropriately. Furthermore, as part of this anticipation, the flux in the limited smooth region ($0 < \bar{\alpha}^n_{c_k} < 1/3$) extends its stencil to include information from the upwind cell. Leonard (1991) and Després & Lagoutière (2001) recognised that when advecting a step function in one dimension the limiting can be performed exactly such that the interface is represented optimally with the available resolution. However, in order to anticipate the flux correctly it is necessary to timestep explicitly.

Explicit advection solutions are limited by the Courant-Friedrichs-Levy (CFL) condition to a Courant number less than 1 everywhere (Courant *et al.*, 1959; Donéa & Huerta, 2003). In essence, explicit timestepping restricts the flow of information into a cell to be from the neighbours with which it shares a face. The CFL condition therefore limits the timestep so that information cannot travel further than one cell in a single timestep. This makes the timestep proportional to the grid size. As the timestep being used is constant across the entire mesh this generally means that it will be limited by the smallest cell (dependent on where the highest velocities are).

For temporally variable velocities, timestep limitations can be overcome by subcycling the advection equation. For example an explicit solution run with a Courant number of 1/4 may perform 10 subcycles to cover the same period of time as an implicit solution, which is not timestep limited, run at a Courant number of 5/2. Fortunately, this has little effect on the cost of calculating the solution. Solving an explicit equation (Equation 3.24) requires only the inversion of a diagonal mass matrix, which can be done directly without incurring the cost of a linear solver. The left hand matrix in an implicit equation (Equation 2.35) has a non-trivial sparsity pattern and generally requires the use of a linear solver. Furthermore, as has already been shown, when using a higher-order face value

method several such solves are required even at low Courant numbers to apply limiting nonlinearly to the solution. This potentially makes the implicit solution more costly.

Unlike single step explicit methods, such iterative nonlinear limited methods are also not guaranteed to converge. This behaviour tends to arise because of the piecewise linear nature of the limiter boundaries. For instance an iterative solution may fall within one limiter region on one iteration and another on the next, oscillating between the two. While the resultant face value, as the sum of three valid face values, is itself potentially valid, it cannot be said to have the properties desired of the high-order face value. This problem is often overcome by introducing a smoothing parameter to the limiter (Jasak *et al.*, 1999) which transforms corners in the limiter function into smoother curves. However, this generally also has the effect of introducing more diffusion to the iterative solution.

Explicit timestepping allows the prediction of full (or empty) cells because, for an incompressible flow in one dimension (Equation 3.24) it is possible to say:

$$\alpha_i^{n+1} = \alpha_i^n + c_{i_{f_i}} \hat{\alpha}^n|_{f_i}^{HO} - c_{i_{f_{i+1}}} \hat{\alpha}^n|_{f_{i+1}}^{HO}, \qquad \text{if } \tilde{u} > 0 \tag{3.28}$$

$$\alpha_i^{n+1} = \alpha_i^n - c_{i_{f_i}} \hat{\alpha}^n|_{f_i}^{HO} + c_{i_{f_{i+1}}} \hat{\alpha}^n|_{f_{i+1}}^{HO}, \qquad \text{if } \tilde{u} < 0 \tag{3.29}$$

or in terms of the donor cell to face f_k:

$$\alpha_{c_k}^{n+1} = \alpha_{c_k}^n + c_{c_{f_{u_k}}} \hat{\alpha}^n|_{f_{u_k}}^{HO} - c_{c_{f_k}} \hat{\alpha}^n|_{f_k}^{HO}. \tag{3.30}$$

To maintain a locally smooth solution, in the sense of Equation 3.11, it is necessary to select $\hat{\alpha}^n|_{f_k}^{HO}$ such that:

$$\alpha_{c_k}^n + c_{c_{f_{u_k}}} \hat{\alpha}^n|_{f_{u_k}}^{HO} - c_{c_{f_k}} \hat{\alpha}^n|_{f_k}^{HO} \leq \alpha_{u_k}^n, \qquad \text{if } \alpha_{u_k}^n > \alpha_{d_k}^n \tag{3.31a}$$

$$\alpha_{c_k}^n + c_{c_{f_{u_k}}} \hat{\alpha}^n|_{f_{u_k}}^{HO} - c_{c_{f_k}} \hat{\alpha}^n|_{f_k}^{HO} \geq \alpha_{u_k}^n, \qquad \text{if } \alpha_{d_k}^n > \alpha_{u_k}^n. \tag{3.31b}$$

These conditions ensure that a donor value that is increasing with time does not exceed the previous timestep's value of the upwind cell (from which the donor is receiving information). Similarly, when the donor cell is decreasing in value it must not go below the previous timestep's upwind value.

To succeed Equation 3.31 clearly requires *a priori* knowledge of both the value at face f_k and at the upwind face, f_{u_k}. Examining Figure 3.5 it becomes clear that this is possible for a step function. In this example downwinding was used at faces that were surrounded by locally smooth solutions ($0 < \bar{\alpha}_c^n < 1$) while upwinding was used everywhere else ($\bar{\alpha}_c^n \leq 0$, $\bar{\alpha}_c^n \geq 1$). Only the face f_k satisfied the smoothness criterion while all other faces were upwinded. Thus in the case of a step function it is possible to assume that the flux into the donor cell is upwinded and Equation 3.31 becomes:

$$\alpha_{c_k}^n + c_{c_{f_{u_k}}} \alpha_{u_k}^n - c_{c_{f_k}} \hat{\alpha}^n|_{f_k}^{HO} \leq \alpha_{u_k}^n, \qquad \text{if } \alpha_{u_k}^n > \alpha_{d_k}^n \tag{3.32a}$$

$$\alpha_{c_k}^n + c_{c_{f_{u_k}}} \alpha_{u_k}^n - c_{c_{f_k}} \hat{\alpha}^n|_{f_k}^{HO} \geq \alpha_{u_k}^n, \qquad \text{if } \alpha_{d_k}^n > \alpha_{u_k}^n \tag{3.32b}$$

reducing the number of unknowns.

Furthermore Figure 3.5 shows that explicit downwinding produces an optimal representation of the step function for the first two timesteps of advection (Figure 3.5(b, c), the number of timesteps for which this is true will be dependent on the Courant number). Therefore it is sensible to try downwinding at the face f_k:

$$\alpha^n_{c_k} + c_{c_{f_{u_k}}} \alpha^n_{u_k} - c_{c_{f_k}} \alpha^n_{d_k} \leq \alpha^n_{u_k}, \qquad \text{if } \alpha^n_{u_k} > \alpha^n_{d_k} \tag{3.33a}$$

$$\alpha^n_{c_k} + c_{c_{f_{u_k}}} \alpha^n_{u_k} - c_{c_{f_k}} \alpha^n_{d_k} \geq \alpha^n_{u_k}, \qquad \text{if } \alpha^n_{d_k} > \alpha^n_{u_k} \tag{3.33b}$$

which, in combination with the smoothness criterion (Equation 3.11), can be converted to:

$$c_{c_{f_k}} \leq \bar{\alpha}^n_{c_k} < 1, \quad \text{or} \tag{3.34a}$$

$$0 < r \leq \frac{1 - c_{c_{f_k}}}{c_{c_{f_k}}}, \quad \text{or} \tag{3.34b}$$

$$-1 < s \leq 1 - 2c_{c_{f_k}}. \tag{3.34c}$$

Therefore explicit downwinding is safe so long as it is only used within the sensor function bounds defined by Equation 3.34.

Equation 3.32 can also be used to suggest a 'limited' value for the face f_k outside the downwinding zone. When the smoothness criteria of Equation 3.34 fail and downwinding for another timestep would result in an unbounded solution then the face value $\hat{\alpha}^n|^{HO}_{f_k}$ should be chosen such that $\alpha^{n+1}_{c_k} = \alpha^n_{u_k}$. In other words:

$$\hat{\alpha}^n|^{HO}_{f_k} = \frac{1}{c_{c_{f_k}}} \left(\alpha^n_{c_k} + \alpha^n_{u_k} \right) + \alpha^n_{u_k}, \qquad \text{when } 0 < \bar{\alpha}^n_{c_k} < c_{c_{f_k}} \tag{3.35}$$

or:

$$\bar{\hat{\alpha}}^n|^{HO}_{f_k} = \frac{1}{c_{c_{f_k}}} \bar{\alpha}^n_{c_k}, \qquad \text{when } 0 < \bar{\alpha}^n_{c_k} < c_{c_{f_k}}, \text{ or} \tag{3.36a}$$

$$\Psi = 2 \left(\frac{1 - c_{c_{f_k}}}{c_{c_{f_k}}} \right), \qquad \text{when } r > \frac{1 - c_{c_{f_k}}}{c_{c_{f_k}}}, \text{ or} \tag{3.36b}$$

$$\Theta = 2 (1 - s) \left(\frac{1 - c_{c_{f_k}}}{c_{c_{f_k}}} \right), \qquad \text{when } 1 - 2c_{c_{f_k}} < s < 1. \tag{3.36c}$$

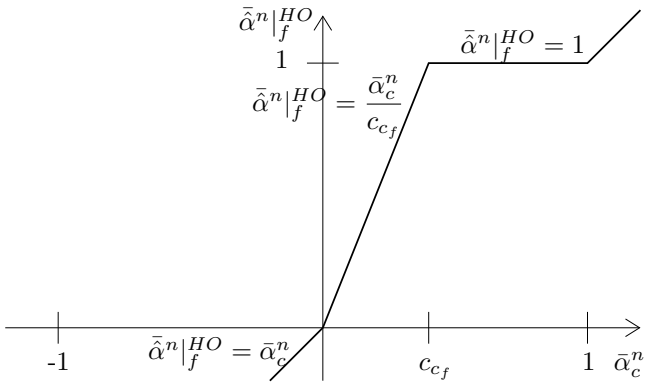
Leonard (1991) first devised this limited downwinding scheme, referring to it as HyperC. It can be plotted on any of the limiter diagrams discussed so far (see Figure 3.8) though it was originally presented on the normalised variable diagram.

Returning to the advection of a step function in Figure 3.5 it is possible to examine how the face value at f_k varies using HyperC (see Figure 3.9):

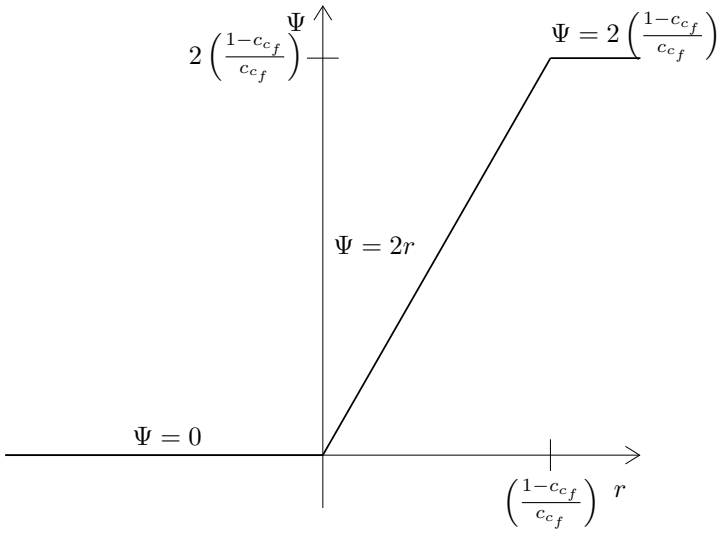
(a) as the step approaches behind f_k, $\alpha_{c_k} \neq \alpha_{u_k}$ but $\alpha_{c_k} = \alpha_{d_k}$, so downwinding begins but this produces the same result as upwinding (Figure 3.9(a));

(b) downwinding continues but as the donor cell fills up the normalised donor variable moves left across the NVD (Figure 3.9(b));

(a) HyperC - NVD



(b) HyperC - Sweby Diagram



(c) HyperC - Waterson Diagram

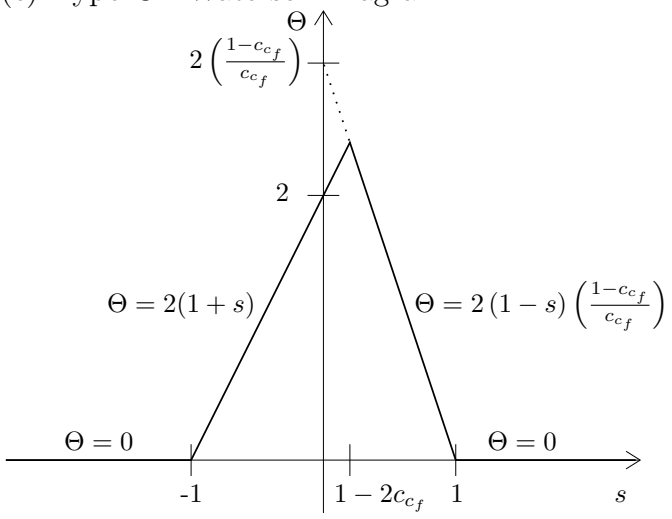


Figure 3.8: The HyperC face value function represented on (a) a normalised variable diagram, (b) the Sweby diagram, and (c) the Waterson diagram.

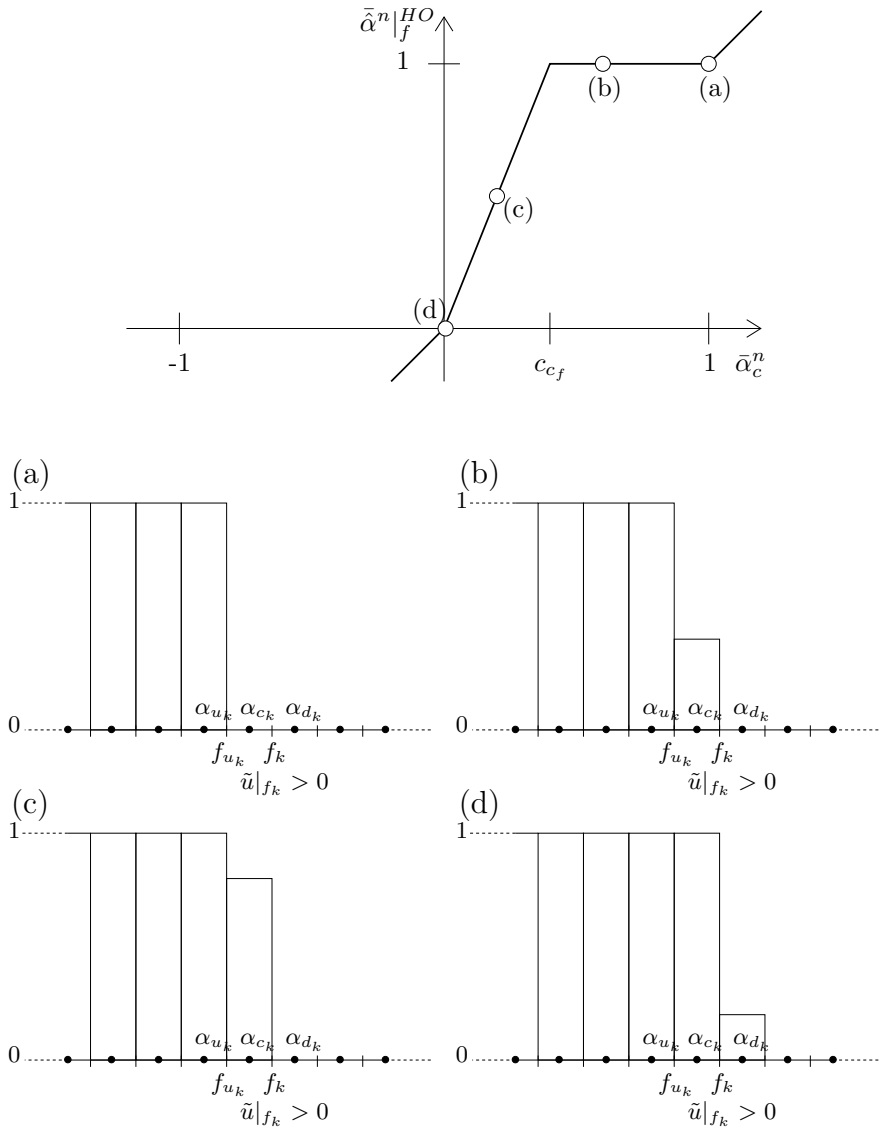


Figure 3.9: The advection of a step function using Equation 3.24 and the HyperC face value scheme. Each timestep illustrated corresponds to a marked position on the normalised variable diagram above.

(c) eventually the normalised donor value reaches the limiting zone of the NVD and the face value is set to allow sufficient flux into the next cell so that the donor cell is just filled (Figure 3.9(c));

(d) behind the face $\alpha_{c_k} = \alpha_{u_k}$ so upwinding begins again (Figure 3.9(d)).

Thus the overshoots seen in Figure 3.5(d) are avoided by the limiting step (Figure 3.9(c)). This can be seen over a longer time period in Figure 3.10 where HyperC has been used to advect a step function.

Without the use of limiter diagrams Després & Lagoutière (2001) devised the same scheme referring to it as limited downwind. They noted that it optimally represented the exact solution when advecting piecewise constant functions. In Figure 3.10 the analytical solution is represented exactly on a mesh with five times more resolution (dotted line). Comparing this with the numerical solution (circles, solid line) it can be seen that they are in close agreement. This can most strikingly be seen when the product of the number of timesteps and the Courant number results in a whole number (Figure 3.10(f,g,h)). At these points no mixed cells are present. In between these times only a single mixed cell exists at the interface. The value in this cell represents how far across that volume the interface has travelled.

Also shown in Figure 3.10 are the explicit (dashed line) and implicit ($\theta = 1/2$ and $\theta_p = 1/2$ in Equation 2.35, dot-dash line in Figure 3.10) first-order upwinding solutions. Clearly these solutions are much more dissipative than the limited downwind solution. Additionally the implicit solution is more dissipative than the explicit one. In fact, if the simulation had been run at a Courant number of 1 then the explicit first-order upwinding solution would have been equivalent to the HyperC solution (Donéa & Huerta, 2003). Effectively the solution would have jumped across the smooth region of the limiter function in a single step, from Figure 3.9(a) to (d) directly, never requiring downwinding or limiting. Unfortunately for an arbitrary velocity field and mesh this is not generally possible.

Leonard (1991) also noted the suitability of HyperC for the advection of step functions however he was interested in the advection of arbitrary functions. Instead of using the face value directly, HyperC was used as the upper boundary on a limiter. In effect HyperC represents the upper boundary of the explicit convective boundedness criterion. Any explicit face value that falls above the lines shown in Figure 3.8 will result in an

Figure 3.10: *See next page.* Eight timesteps from the advection of a step function using Equation 3.24 at a constant Courant number of 2/5 with $\tilde{u} > 0$. The HyperC solution (circles, solid line) uses upwinding where the solution is non-smooth (Equation 3.12), downwinding where possible (Equation 3.34) and limited values from Equation 3.36 otherwise. For comparison the exact solution is shown (dotted line) as well as the explicit (dashed line) and the more dissipative implicit (dot-dash line) first-order upwinding solutions.

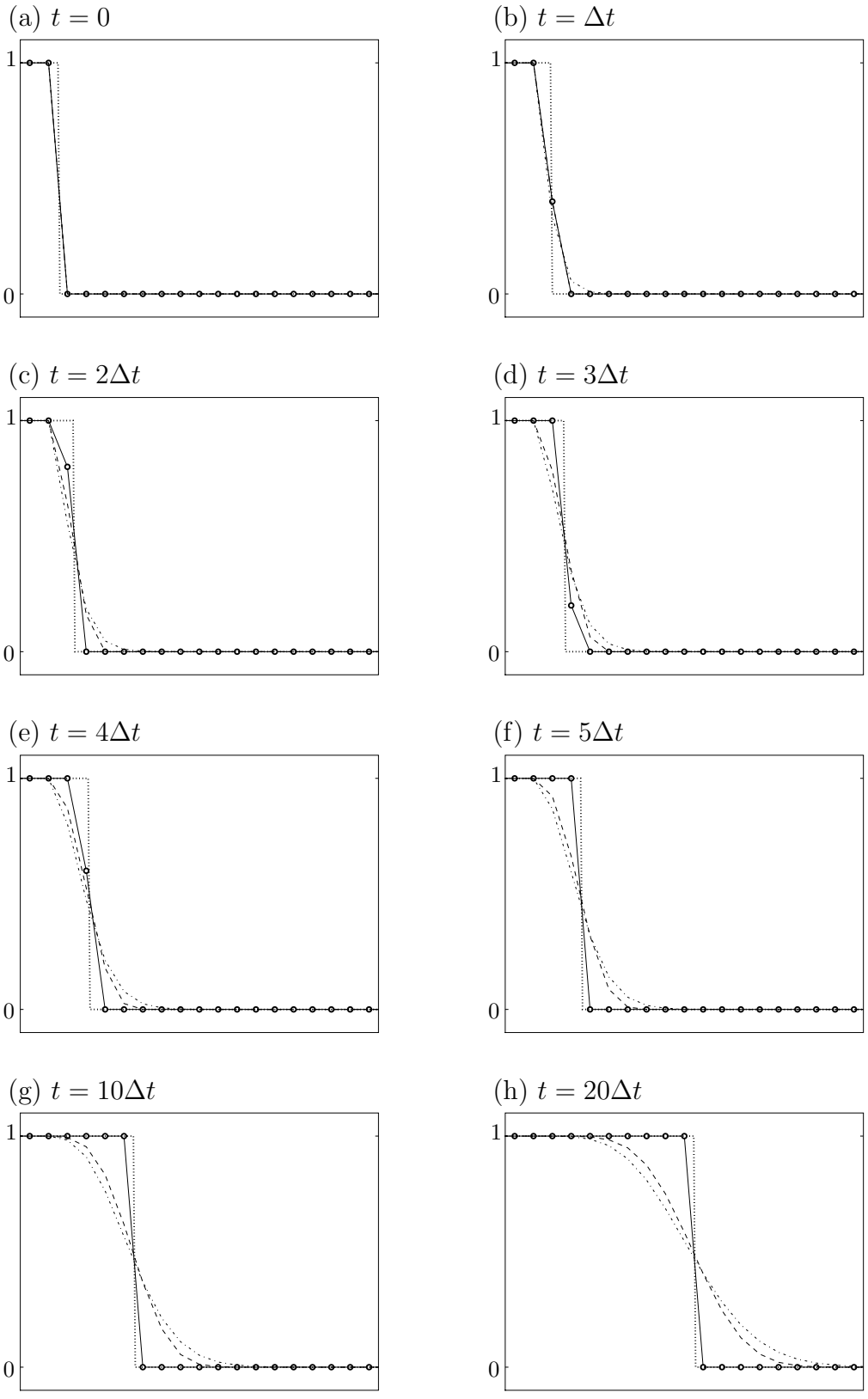


Figure 3.10: *cont.*

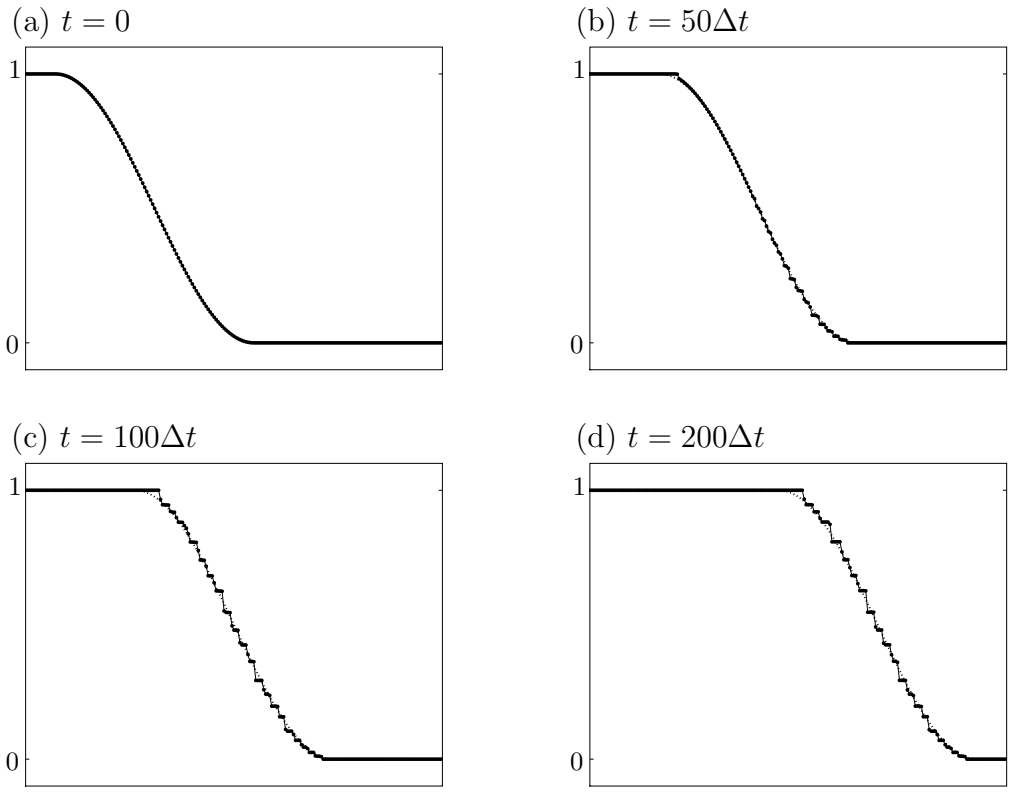


Figure 3.11: Four timesteps from the advection of a quarter period of a cosine function using Equation 3.24 at a constant Courant number of 2/5 with $\tilde{u} > 0$. The solution using HyperC (circles, solid line) becomes distorted during advection. For comparison the exact solution is shown (dotted line).

unbounded solution.

Examining HyperC represented on the Waterson diagram (Figure 3.8(c)) it is clear that it does not fulfil the often desired symmetry property $\Theta(s) = \Theta(-s)$ ($\Psi(r) = r\Psi\left(\frac{1}{r}\right)$, on the Sweby diagram, Figure 3.8(b)) except when $c_{c_{f_k}} = 1/2$. Advection schemes with the symmetry property give equal weight to both slopes, a and b, on either side of the face in the formulation of the face value. As HyperC does not provide this, it suggests that smooth functions will become distorted when advected using HyperC.

The distortion of a smooth function using HyperC can be seen in Figure 3.11. In effect the smoothly varying part of the solution has been 'staircased' by HyperC. This occurs because the assumption that only one face is surrounded by smoothly varying nodes is now incorrect. Hence assuming that the upwind face, f_{u_k}, is upwinding also breaks down (Equation 3.32). Instead if this face is also within a smooth region it will be downwinding. In the case of Figure 3.11 this results in an overestimation of the upwind flux and the commencement of limiting earlier than is necessary to prevent over-filling of the cell. Conversely when the slope is reversed the assumption of upwinding underestimates the upwind flux with the same effect, downwinding is limited to prevent under-filling.

Such over-zealous limiting coupled with downwinding at the upwind face results in a local plateau developing in the solution. As these regions satisfy the upwinding assumptions of HyperC the local step function in front of this level zone falls into an advection regime where it can be sustained. Hence the originally smooth function propagates as a staircase.

While the failure of the gradient symmetry property means that HyperC is clearly unsuited to the advection of smooth functions, it does possess another symmetry property essential to material advection. If the advected field were used to represent a volume fraction then by Equation 3.1 it should not matter which volume fraction is used to represent the interface. In other words, given an interface represented by a step function in a volume fraction, the forward facing step and the backward facing step should be advected symmetrically. HyperC satisfies this requirement because, in a two material flow, the sensor variable of one of the volume fractions is the same as that of the other. For example, in terms of normalised donor variables:

$$\bar{\alpha}_c^2 = \overline{1-\alpha_c}^1 = \bar{\alpha}_c^1. \tag{3.37}$$

Thus so long as Equation 3.1 is satisfied prior to advection then both the forward and backward facing steps experience the same regime of face value evaluation (upwinding, downwinding or limiting). This results in the normalised face values that also satisfy:

$$\bar{\hat{\alpha}}^2|_f^{HO} = \overline{1-\hat{\alpha}}^1|_f^{HO} = \bar{\hat{\alpha}}^1|_f^{HO}, \tag{3.38}$$

and hence Equation 3.1 is generally satisfied.

For one-dimensional flows HyperC is therefore suitable for material advection. In fact it produces the same result as a one-dimensional volume of fluid (VOF) method. For instance, in the case of Figure 3.9(b) $\alpha_{c_k} = 2/5$ so a VOF method would construct the interface two fifths of the way across the donor cell from f_{u_k} (see Figure 3.12(a)). As the Courant number is also 2/5 the face f_k requires information from a region 2/5 of the cell width away from it. As the interface does not fall within this region the face value is 0, the same result as with downwinding.

Using similar logic in the case of Figure 3.9(c), the interface is now 4/5 of the cell width away from face f_{u_k}. This falls halfway into the Courant flux region of face f_k (see Figure 3.12(b)). Thus the face value required to prevent over-filling of the donor cell is 1/2, the same as HyperC.

3.3 Multiple Dimensions

In one dimension, limited downwinding using HyperC provides an optimal method of advecting materials. On a structured orthogonal cube mesh this can be extended to multiple dimensions using operator splitting. Operator splitting considers a d dimensional problem as d one-dimensional problems. In essence each timestep is split into d substeps during each of which the field is advected along one of the coordinate directions. Generally the order of the directional sweeps is varied between the timesteps to avoid the accumulation of systematic error (Rudman, 1997).

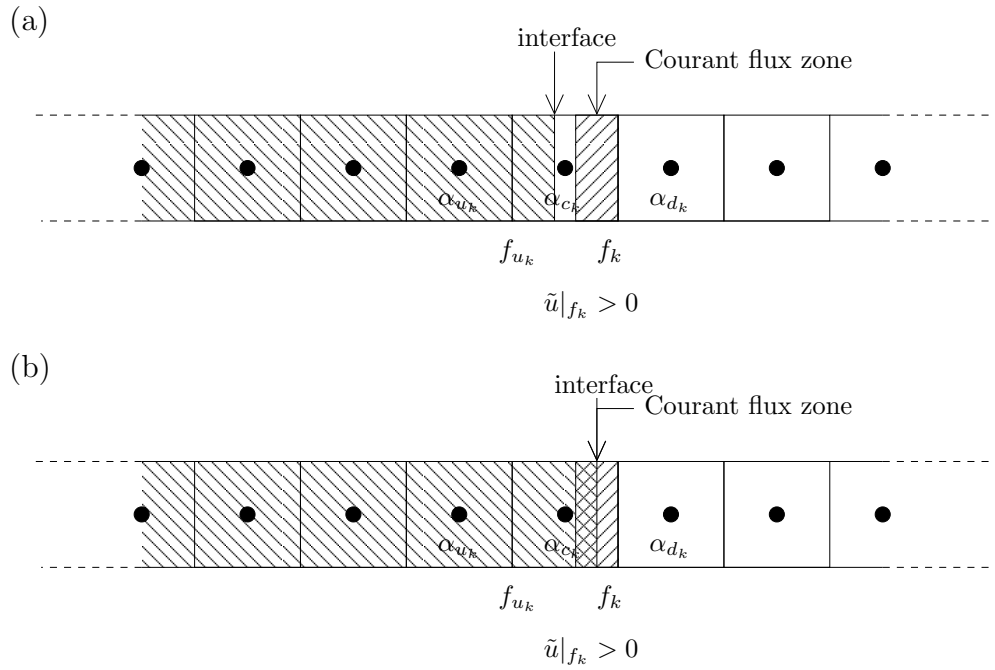


Figure 3.12: The advection of a step function using a one-dimensional volume of fluid method. The timesteps shown correspond to (a) Figure 3.9(b) and (b) Figure 3.9(c).

Operator splitting overcomes the problem of corner coupling (Thuburn, 1996), as demonstrated in Figure 3.13. Information is propagating in the direction of $\tilde{u}$ from node A to node B. As A and B share no faces, in an explicit finite volume scheme no information passes between them in a single timestep. If, however, the timestep is split into two parts, one advected by $\tilde{u}_{x_1}$ and the other by $\tilde{u}_{x_2}$ then information can be transferred between A and B via their shared neighbouring nodes.

This scheme has been applied successfully to various material advection algorithms. In particular, volume of fluid methods generally use operator splitting methods in higher dimensions. Unlike in one-dimensional models, extra freedom is introduced in determining the interface position.

The simple line in cell method (SLIC-VOF, Noh & Woodward, 1976) treats each sweep of operator splitting as a truly one-dimensional problem, considering only the neighbouring values in the direction of current advection and potentially defining different interface orientations depending on the direction of the advection sweep. It is therefore the most similar system to the scheme described in Figure 3.12 and to the three point (upwind, donor and downwind cells) stencil of HyperC using operator splitting (Jun, 2000).

In other VOF schemes the interface orientation does not depend on the direction of the advection sweep. For instance, in a VOF scheme devised by Hirt & Nichols (HN-VOF, Hirt & Nichols, 1981) a nine point stencil is used in two dimensions to determine the interface orientation. This is then used to approximate the interface, which, like in the SLIC-VOF scheme, is orientated either parallel or perpendicular to the coordinate

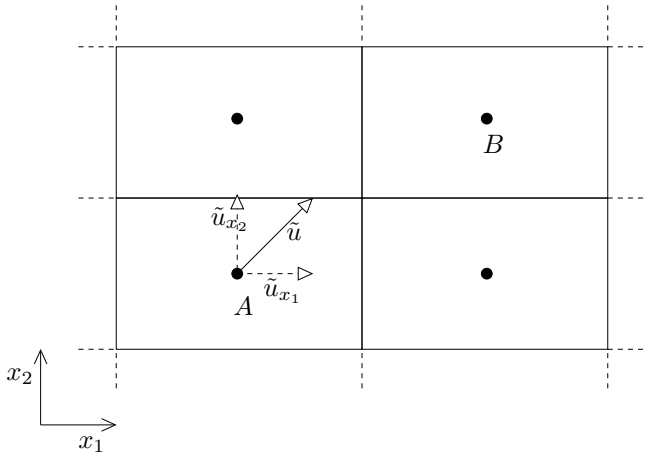


Figure 3.13: The problem of corner coupling on a structured orthogonal mesh. Information cannot propagate directly from node A to node B.

axes but is kept the same for all subtimesteps.

Youngs's scheme (Y-VOF, Youngs, 1982) also uses an extended stencil to determine the orientation of the interface. In two dimensions there are then four possible configurations for how this interface intersects with the cell edges depending on the values in the neighbouring cells. Once the correct configuration is determined a 'filled' fluid polygon can be defined using the cell edges and the interface. The area (or volume) of this polygon is then adjusted by moving the interface so that it agrees with the volume fraction value in the cell. This interface position is used to determine the fluxes through the cell faces.

Flux corrected transport VOF (FCT-VOF, Zalesak, 1979) uses a predictor-corrector approach to maintaining a sharp interface. First solving the system using first-order upwinding before calculating an anti-diffusion based on a combination of first-order upwinding and first-order downwinding to correct the first solution. In essence this is very similar to the combined low-order and iterative high and low-order scheme suggested by LeVeque (1992). In one dimension this procedure should produce the same result as HyperC, however with operator splitting in multiple dimensions Rudman (1997) found that capping the solution values was necessary to prevent unbounded behaviour.

Zalesak (1979) proposed applying such an iterative flux corrected transport scheme directly in multiple dimensions. However, Rudman (1997) showed that for sharp interfaces the results were unsatisfactory, producing non-physical isolated regions of material (so-called 'flotsam and jetsam') and distorting the shape of the interface.

Ubbink & Issa (1999) noted that introducing some degree of implicitness to the calculation meant that information could be transmitted between nodes A and B directly, overcoming the corner coupling problem. They proposed using HyperC as the high-order flux in an iterative scheme with a low-order pivot (as in Equation 2.35). Noting that this tended to wrinkle the interface (Hirt & Nichols, 1981; Lafaurie *et al.*, 1994) they therefore introduced a switch to a less compressive scheme (QUICK, Baliga & Patankar,

1998) when the interface was orientated parallel to the direction of transport.

Using a face value method like HyperC (a series of lines on the normalised variable diagram) rather than a limiter (an area on the normalised variable diagram) within an implicit iterative face value scheme increases problems with convergence and boundedness of the solution. This arises because at each iteration the sensor variable changes. It therefore becomes possible for the nature of the face value method, selected according to the sensor variable, to change during a single timestep. For instance with HyperC the behaviour may change from downwinding to limited. Furthermore, this behaviour may become oscillatory leading to non-convergence of the solution. As this means the implicit and iterative low-order solutions do not cancel it also generally means that the resulting solution is unbounded.

Despite introducing a more dissipative face value scheme such unbounded behaviour was still seen by Ubbink & Issa (1999), who introduced a correction step to smooth the resulting solution. Jasak *et al.* (1999) introduced a smoothing parameter to the face value scheme to help resolve such convergence problems. Corners between lines on the face value function were instead replaced by curves, smoothing the transition between the two different behaviours. This increases the probability of convergence of the method but, despite the extra diffusion introduced by relaxing the boundedness criterion, any such iterative scheme is still not necessarily bounded. Furthermore, Darwish & Moukalled (2006) demonstrated a number of iterative flux limited schemes showing that, despite being semi-implicit, most methods' behaviour deteriorated dramatically at higher Courant numbers. Hence their timesteps were still restricted by the Courant number.

Jun (2000) demonstrated that using HyperC explicitly on a structured quadrilateral mesh without operator splitting generated flotsam and jetsam in a similar manner to multi-dimensional flux corrected transport. However, the solution was still bounded and using operator splitting substantially improved the results.

3.3.1 Control Volume Dual Mesh

On an unstructured or non-regular mesh, operator splitting is not in general possible. The control volume edges and faces are no longer strictly aligned parallel and perpendicular to the coordinate axes or each other. Hence, it is not possible to break the fluxes down into a series of orthogonal directions. The operator splitting problem is illustrated in Figure 3.14(a) on a triangular mesh. Clearly the faces surrounding node A are not aligned with the coordinate axes or the other faces of the neighbouring cells. Hence no decomposition of the velocity will advect material in only a single direction as on an orthogonal structured mesh.

Figure 3.14(a) also demonstrates that the corner coupling problem is actually accentuated by a triangular mesh. Instead of being separated by a single cell, nodes A and B may now be separated by an arbitrary number of cells (in this example the minimum is two). Hence even if operator splitting along the coordinate axes were possible d sub-timesteps would not necessarily communicate information between A and B successfully.

It is proposed here that the problems with corner coupling and operator splitting may

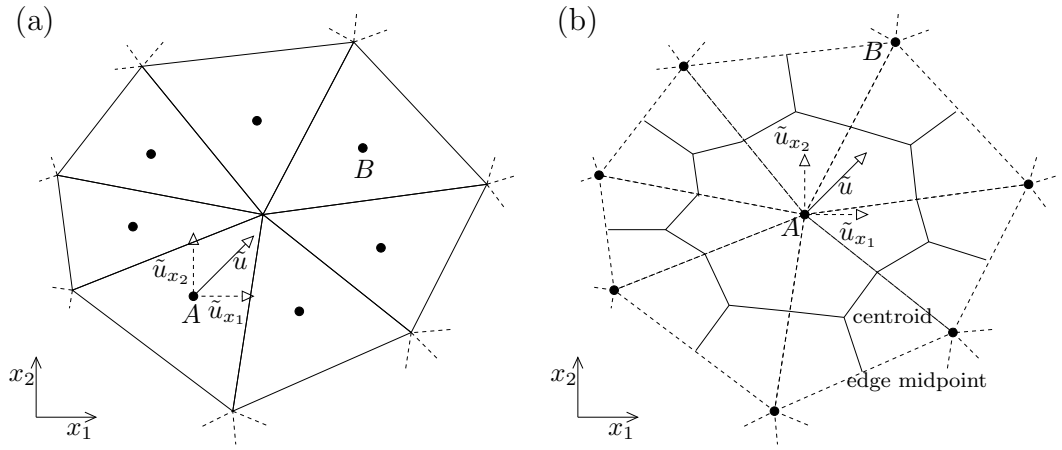


Figure 3.14: (a) The problem of corner coupling and operator splitting on unstructured simplex meshes. The cell faces are no longer aligned so no decomposition of the velocity vector advects strictly in one direction at a time. (b) A dual control volume mesh (solid lines) to the piecewise linear continuous finite element mesh (dashed lines). The solution nodes for both meshes are co-located (black circles).

be overcome on simplex meshes by adjusting the definition of a control volume used to define a cell. This means that the advection equation (2.35) will be solved on a mesh that is in some sense dual to the parent simplex mesh. In terms of the equivalence of finite element and finite volume methods, discussed in Chapter 2, the volume will no longer be coincident with the zeroth order discontinuous element on the parent mesh.

Dual control volume meshes to finite element grids have been proposed before but mostly as a means of ensuring co-location of the nodes of the finite volume and finite element solutions (Gresho *et al.*, 1998; Patil & Lakshmisha, 2009). To overcome corner coupling a suitable dual mesh should share faces between all neighbouring nodes, allowing information to propagate between them in a single advection step.

In two dimensions, a suitable mesh is constructed around a piecewise linear finite element triangular mesh. In this case a control volume edge is constructed by connecting the centroid or the circumcentre of a triangle to the midpoint of the triangle edge. Three such edges can be defined per triangle. Connecting these to those in the neighbouring triangles a control volume can be constructed around the same solution nodes as the piecewise linear continuous finite element mesh. As on fully unstructured meshes the triangles may not be acute and the circumcentre may leave the triangle, the control volume face definition between the element centroid and edge is selected here.

Figure 3.14(b) gives an example of such a control volume surrounding node A (solid lines). Clearly nodes A and B now share a face through which information can be transferred in a single timestep. Similarly node A shares a face with all the surrounding nodes.

A centroid based dual mesh is easily extended to three dimensions. Two edges of a control volume face are constructed by connecting the centroid of a tetrahedron with

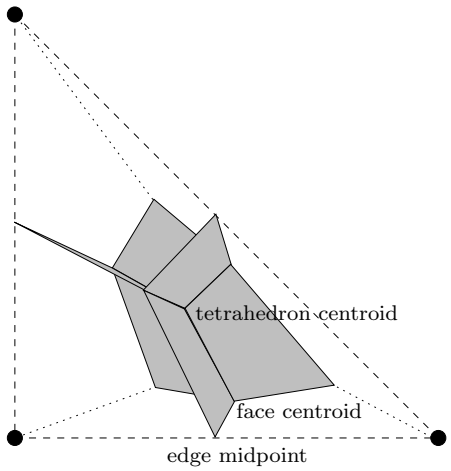


Figure 3.15: The six dual control volume mesh faces within a piecewise linear tetrahedral parent mesh element.

the centroids of the tetrahedron's faces. A further two edges are constructed in turn by connecting the tetrahedron face centroids to the tetrahedron edge midpoints. Thus a four sided control volume face is created. There are six such faces per piecewise linear continuous element of the parent mesh (see Figure 3.15).

Similarly the dual mesh can be extended to higher-order continuous element parent meshes. For instance there are nine control volume edges and 24 control volume faces per piecewise quadratic simplex element in two and three dimensions respectively. This is most easily seen in two dimensions as a piecewise quadratic triangular element may be broken down into four piecewise linear triangles, on which the same construction procedure as described above may be performed to arrive at the dual mesh for the quadratic mesh. Figure 3.16 illustrates the dual control volume meshes to both piecewise linear and piecewise quadratic elements in one and two dimensions.

Also shown in Figure 3.16 are the equivalent control volume shape functions over the support, s, for node A on the dual meshes. This highlights another difficulty implementing the centroid based dual mesh: the shape of the support is arbitrary. In other words the number of control volume faces surrounding a node depends on the number of elements surrounding that node in the parent mesh. This can be seen by comparing Figures 3.14(b) and 3.16(c) where seven and six elements surround node A leading to fourteen and twelve control volume edges respectively.

The arbitrary shape of the control volume means that assembly of the discrete equations is complicated. Consider the assembly methods discussed in Chapter 2. They all involved looping over the elements of a mesh assembling the contribution to the global equation inside or around each volume. This assembly is based on a canonical representation of the element which is transformed to the physical domain using a one-to-one mapping. As the control volume now has an arbitrary shape no mapping exists between a suitable canonical space and the physical mesh.

A simple mapping still exists between the parent element and the physical domain.

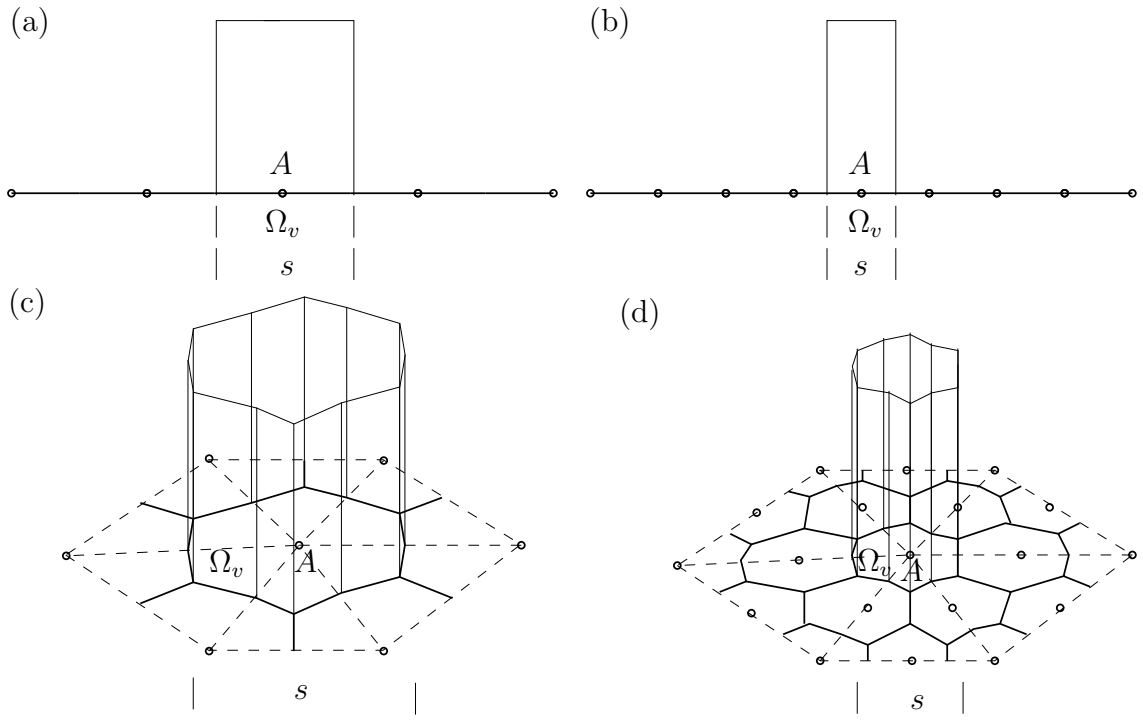


Figure 3.16: One-dimensional (a, b) and two-dimensional (c, d) schematics of piecewise constant control volume shape functions and dual meshes based on the parent (dashed lines) linear (a, c) and quadratic (b, d) finite element meshes. The shape function has value 1 at node A descending to 0 at the control volume boundaries. The support, s, coincides with the volume, Ω_v.

Furthermore the positions of the control volume faces are known within the canonical parent element and can be mapped to the physical domain using the same mapping. Each control volume face is itself a simple shape – a line in two dimensions and a quadrilateral in three – so that standard rules of quadrature can be applied across an individual face. Assembly is therefore still performed on an element-by-element basis utilising the mapping from the control volume faces in the canonical parent element to physical space. Surface quadrature is then performed on each face individually. The only volume integral required in Equation 2.35 is the mass matrix, which gives the area or volume of the control volume.

For a piecewise linear parent mesh the centroid based structure of the control volume dual mesh divides a parent element into $d+1$ equal sections. Therefore it is only necessary to know the area or volume of the parent element to find its contributions to the control volume mass matrix. This is easily obtained by lumping the mass matrix of the parent mesh. Higher-order parent mass matrices contain negative off diagonal entries, which means that lumping them does not give the correct mass contributions to the control volume mesh. For piecewise quadratic triangles lumping on the piecewise linear mesh formed by decomposing each quadratic triangle into four linear triangles does however produce the correct answer. On other meshes it is still possible to calculate the fraction of

the canonical parent element volume associated with each section of the control volume and scale these appropriately to find the control volume mass matrix. It is therefore not necessary to explicitly assemble the control volume mesh. Instead only a single canonical representation of it inside a canonical parent element is stored (like in Figure 3.15). Hence only the parent meshes are shown in subsequent figures.

The potentially arbitrary shape of the control volumes also demonstrates why, even on a mesh without corner coupling issues, interface reconstruction techniques are rejected here. Of the VOF schemes discussed earlier (Section 3.3), HN-VOF and Y-VOF consider two and four possible orientations of the interface in two dimensions respectively. The number of possible interface positions and the cost of determining the correct orientation scales with the number of faces surrounding a control volume, which in turn scales with the dimension of the problem. Additionally, implementations of the algorithm generally assume a fixed number of faces per control volume. The complexity and cost of implementing interface reconstruction for arbitrary cells was therefore considered prohibitive especially as it would necessitate constructing the entire control volume mesh.

A flux limited solution like HyperC, extended to higher dimensions, was therefore considered the best available method of advecting fields on the control volume dual mesh. However, flux limiting strategies (as discussed in Section 3.2) require a three point stencil of the donor, downwind and upwind nodes. Given a face and a direction of flow on any control volume mesh the donor and downwind nodes are clearly defined. In multiple dimensions however the definition of an upwind node is less clear.

Using operator splitting on a structured mesh the locations of the upwind nodes are defined by the current direction of transport. In an unsplit, fully multi-dimensional method the same definitions may be used by taking the upwind value for a particular face from the cell across the opposing face in the donor cell. However, given the arbitrary shape of the control volumes on an unstructured mesh there will not in general be a clearly definable opposing upwind face.

Jasak *et al.* (1999) proposed that the upwind node could be approximated by projecting the gradient of the field at the donor cell, $\alpha_{c_k,i}$, along the vector connecting the donor node to the downwind node, ζ_i:

$$\alpha_{u_k} = \alpha_{d_k} - 2\left(\alpha_{c_k,i}\zeta_i\right). \tag{3.39}$$

What is not clear from this definition is how the gradient at the donor cell, $\alpha_{c_k,i}$, is to be evaluated. Using the basis functions from the parent mesh, N_A, it is possible to evaluate $q_i = \alpha_{,i}$ at all nodes by solving:

$$\mathbf{M_q} q_i = \mathbf{C_{q_i}} \alpha \tag{3.40}$$

where

$$\mathbf{m_q}_{ab} = \int_{\Omega_e} N_a N_b \mathrm{d}V \qquad \Rightarrow \mathbf{M_q}_{AB} \qquad \text{mass matrix} \tag{3.41a}$$

$$\mathbf{c_q}_{iab} = \int_{\Omega_e} N_a N_{b,i} \mathrm{d}V \qquad \Rightarrow \mathbf{C_q}_{iAB} \qquad \text{gradient matrix} \tag{3.41b}$$

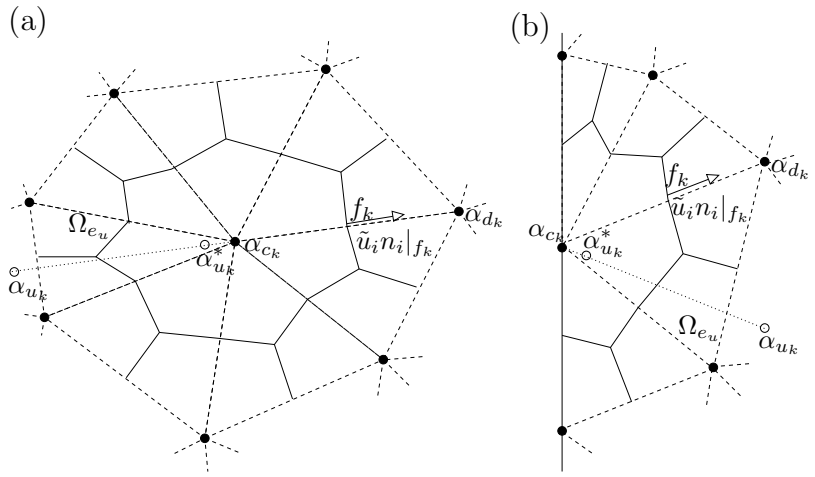


Figure 3.17: Calculation of the upwind value, α_{u_k}, on an unstructured mesh (a) internally and (b) on a boundary. The control volume mesh is shown (solid lines) around the nodes (black circles) which are co-located with the solution nodes of the parent piecewise linear continuous finite element mesh (dashed lines). An initial estimate of the upwind value, $\alpha^*_{u_k}$, is found by interpolation within the upwind parent element, Ω_{e_u}, of face k, f_k. The point-wise gradient between this estimate and the donor node, α_{c_k}, is then extrapolated the same distance between the donor and downwind, α_{d_k}, to the upwind value, α_{u_k}.

as with the Hessian calculation (see Chapter 2). q_i may then be used for the values of $\alpha_{c_k,i}$.

Calculating the gradient via Equation 3.40 effectively takes an average of the gradient over the support of the parent mesh elements surrounding the control volume node. This was found to be excessively diffusive, underestimating the value at the nodes in the actual upwind direction. A better estimate was therefore introduced by projecting off the donor instead of the downwind node:

$$\alpha_{u_k} = \alpha_{c_k} - (\alpha_{c_k,i}\zeta_i)\,, \tag{3.42}$$

and calculating the dot product, $\alpha_{c_k,i}\zeta_i$, in a single step rather than separately.

This process is illustrated in Figure 3.17(a). First the element, Ω_{e_u}, in the direction $-\zeta_i$ from the donor node, α_{c_k}, is found. This involves a search of all the elements surrounding the donor node for every node pair in the element mesh. Fortunately, this need only be performed once per parent mesh. Once Ω_{e_u} is found, the basis functions within the element are used to find an estimate of the upwind value, $\alpha^*_{u_k}$, some fraction of the distance into the element along $-\zeta_i$. The difference between this value and the donor node, α_{c_k}, can then be used to give a point-wise estimate of the gradient in that direction, which can be extrapolated to give $\alpha_{c_k,i}\zeta_i$.

Near boundaries an upwind element does not exist in the direction $-\zeta$. Therefore the projection is instead reflected off the boundary back into the domain after which the process continues as before (see Figure 3.17(b)).

The gradient calculated as in Figure 3.17 only uses information local to the upwind

Mesh	h, Average Mesh Spacing
A	0.08
B	0.04
C	0.02
D	0.01
E	0.005

Table 3.1: Average grid point spacings for meshes used in convergence analysis.

direction and was found to produce a better estimate of the values in that direction. Clearly however whether calculated using Equation 3.39 or 3.42 the upwind value is only an approximation of the solution in the vicinity of a fictitious upwind node.

Convergence Analysis

Given a parent finite element mesh and the canonical locations of the control volume faces within it, it should now be possible to assemble and solve Equation 2.35 using the elemental assembly methods described above. Furthermore, given the upwind value method described, it should also be possible to use a limited higher-order method.

To test the implementation of these methods in Fluidity a convergence analysis was performed using the method of manufactured solutions (MMS, Roache, 2002). MMS provides an easy way of generating analytical solutions against which to verify model code. Essentially a sufficiently smooth desired analytical solution is invented and a suitable source term added to the right hand side of the equation to ensure that the desired solution is actually the solution to the equation. The source is calculated by substituting the desired analytical solution in the underlying differential equation.

The numerical equation (in this case Equation 2.35 with a source term, $\int_{\Omega_v} dV S_A$, added to the right hand side) is then solved on a series of successively finer meshes. The solution on each of these is then compared to the known exact solution, α^e, and the order of convergence compared to the expected order for that method. When convergence of the solution is not seen it is an excellent indicator of an error in the model code or the numerical formulation. MMS has been shown to be highly effective at finding such problems (Salari & Knupp, 2000).

In these test cases a rectangular domain, Ω, was defined as $x_1 \in [0.1, 0.6]$, $x_2 \in [-0.3, 0.1]$. This was discretised on a series of successively finer mesh sizes, h, given in Table 3.1. As the meshes are unstructured these correspond to the average mesh spacing on the parent mesh. All the parent meshes used were equivalent to piecewise linear continuous triangular meshes.

The velocity field, $\tilde{u}_i$, was selected as:

$$\tilde{u}_1 = \sin\left(5\left(x_1^2 + x_2^2\right)\right), \tag{3.43}$$

$$\tilde{u}_2 = \cos\left(3\left(x_1^2 - x_2^2\right)\right), \tag{3.44}$$

and a desired solution field, α^e, was created:

$$\alpha^e = \sin\left(25x_1x_2\right) - \frac{2x_2}{\sqrt{x}}. \tag{3.45}$$

Using the analytical linear advection equation and the desired solution, the source can then be found as:

$$S = \alpha^e_{,t} + u_i\alpha^e_{,i}, \tag{3.46}$$

giving (using the symbolic maths package SAGE, Stein *et al.*, 2009):

$$S = \left(25x_2\cos\left(25x_1x_2\right) + \frac{x_2}{x_1^{3/2}}\right)\sin\left(5\left(x_1^2 + x_2^2\right)\right) + \left(25x_1\cos\left(25x_1x_2\right) - \frac{2}{\sqrt{x_1}}\right)\cos\left(3\left(x_1^2 - x_2^2\right)\right). \tag{3.47}$$

The velocity was placed on the piecewise linear parent mesh to allow the finite element basis functions to be used to interpolate it onto the control volume faces. This avoids the values being undefined at the faces of the control volume mesh if the velocity were control volume based.

Given the locations of the domain boundaries there are inflowing velocity vectors along the bottom and left hand edges, $\partial\Omega_{\text{in}}$. At these locations α_{in} was set equal to the exact solution, α^e.

Two face value methods were used to calculate the numerical solution, α; first-order upwinding and a Sweby limited interpolant. First-order upwinding requires only information from the donor cell so mostly tests the correct implementation of the canonical dual mesh and its mapping to the physical domain.

The Sweby limited interpolant uses the basis functions of the parent mesh to interpolate the field values onto the control volume faces. As these are in general unbounded it then uses the Sweby limiter to attempt to bound the face value. In one dimension this method is equivalent to the Sweby limited trapezoidal method. In addition to the mesh testing provided by first-order upwinding, this method tests the interpolation between the parent and dual meshes as well as the projection to the upwind value, which is required for the limiter.

The desired solution is temporally invariant. However, the initial condition, α_0, was set to zero everywhere. Therefore the numerical solution to Equation 2.35 will vary over time until a steady state is reached. The tolerance used to determine when a steady state had been reached was set as $||\alpha^{n+1} - \alpha^n||_\infty = 10^{-10}$, where n is the timestep level.

Once a steady state has been obtained on all meshes (Table 3.1) the convergence analysis may be performed. Given the error, E, on two meshes, A and B for example:

$$E_A \approx Ch^{c_p}, \tag{3.48}$$

$$E_B \approx C\left(\frac{h}{r}\right)^{c_p}, \tag{3.49}$$

Meshes	Upwinding	Sweby Limited Interpolant
A→B	0.64	1.28
B→C	0.87	1.47
C→D	0.97	1.43
D→E	0.92	1.38

Table 3.2: Order of convergence, c_p, for the advection convergence tests.

where C is a constant discretisation specific factor independent of the mesh, c_p is the order of convergence of the method and r is the refinement ratio ($r = 2$ in this case), then the ratio of errors is given by:

$$\frac{E_A}{E_B} \approx \left(\frac{Ch^{c_p}}{Ch^{c_p}}\right) r^{c_p} = r^{c_p}, \tag{3.50}$$

and the order of convergence can be calculated as:

$$c_p \approx \log_r \left(\frac{E_A}{E_B}\right). \tag{3.51}$$

This can then be compared to the expected order of convergence for a particular method. For example, as its name suggests first-order upwinding is first-order accurate. The errors were evaluated in the L_2 norm, such that for any given mesh:

$$E = ||\alpha - \alpha^e||_2 = \sqrt{\int_\Omega |\alpha - \alpha^e|^2 dV}, \tag{3.52}$$

where α is the numerical solution to Equation 2.35 on that mesh once a steady state has been attained and α^e is a representation of the exact solution on the same mesh.

The temporal discretisation used $\theta = 1/2$ and $\theta_p = 1/2$ while the timestep, Δt, was set on each mesh to give a maximum Courant number of 1/2. (The multi-dimensional Courant number is defined later.) For the Sweby limited interpolant case a tolerance of 10^{-10} was used to define convergence between the implicit and nonlinear low-order pivot solutions. This took a maximum of 4 iterations per timestep.

The results using both face value methods are presented in Figure 3.18, showing the solution using first-order upwinding, the Sweby limited interpolant solution as well as the analytical solution and the parent mesh used. Table 3.2 and Figure 3.19 give the results of the convergence analysis for the results shown in Figure 3.18.

The results for both methods are slightly oscillatory, which is probably a result of the variability in edge lengths on the unstructured mesh. The order of convergence for first-order upwinding tends toward $c_p = 1$ as expected. The order of convergence for higher-order limited methods is not generally provable, especially in higher dimensions (LeVeque, 1992), however convergence between successive meshes is seen using the Sweby limited interpolant. Additionally, the order of convergence is higher than seen with upwinding.

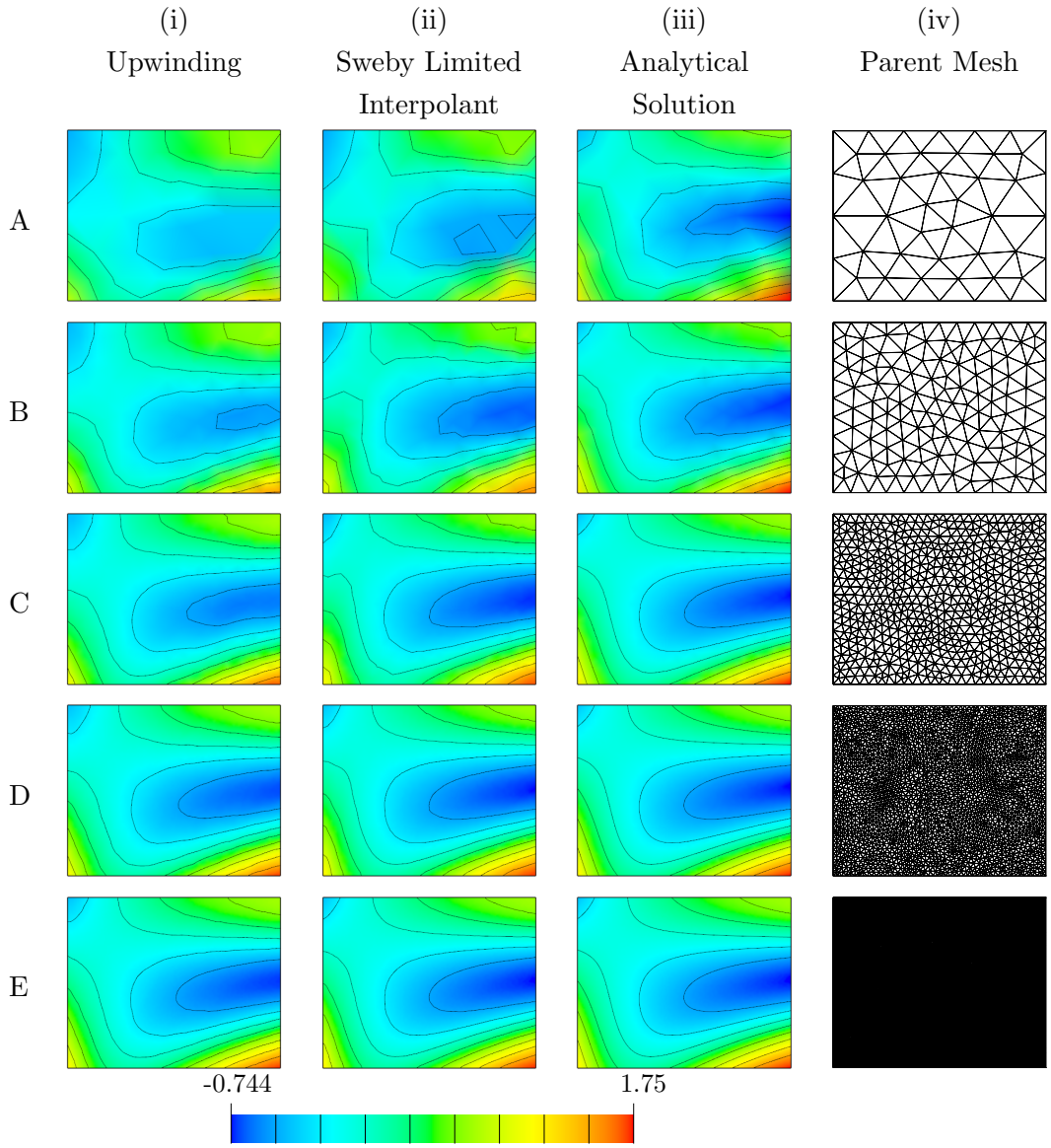


Figure 3.18: Numerical solution to Equation 2.35 with a source term given by Equation 3.47 using (i) first-order upwinding and (ii) a Sweby limited interpolant. Also shown are the expected analytical solution (iii) and the parent mesh used (iv).

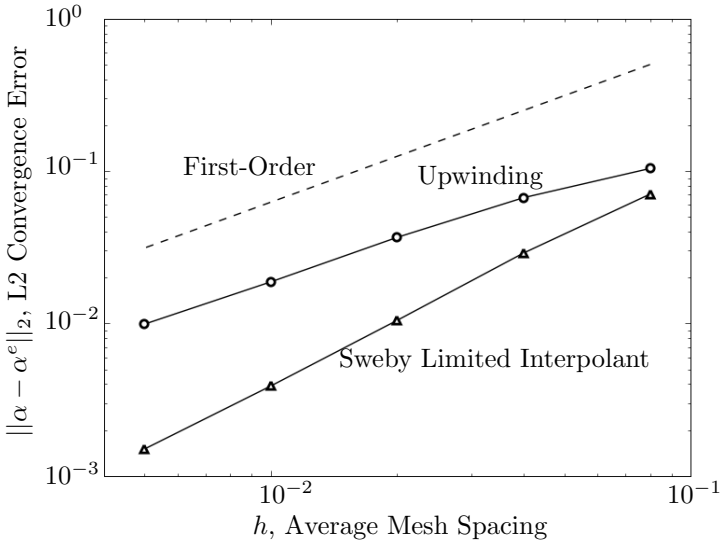


Figure 3.19: Plot of the error L_2 convergence error against average mesh spacing, h, for the advection convergence tests.

These results indicate that both the dual control volume mesh and upwind value method have been implemented correctly. The convergence analysis presented above was submitted as an automated regression test for Fluidity. This ensures that subsequent changes to the code base are tested to ensure these convergence results are unaffected.

3.3.2 Material Tracking

Having demonstrated the suitability of the control volume mesh and upwind value method in higher dimensions it is now proposed that adapting HyperC for use on this mesh will provide an adequate method for advecting materials in higher dimensions. As in one dimension, Equation 2.35 is reduced to its fully explicit form and the solution is now sought to:

$$\alpha_i^{n+1} = \alpha_i^n - \frac{\Delta t}{\int_{\Omega_{v_i}} \mathrm{d}V} \left(\underbrace{\int_{\partial\Omega_{v_i}} \tilde{u}_i n_i \hat{\alpha}^n|^{HO}_{\partial\Omega_{v_i}} \mathrm{d}s}_{\text{high-order advective flux term}} - \underbrace{\int_{\partial\Omega_{v_i}} \tilde{u}_i n_i \mathrm{d}s\, \alpha_A^n}_{\text{explicit velocity divergence term}} \right). \quad (3.53)$$

Unlike in one dimension the explicit velocity divergence term has been left in Equation 3.53 to allow for spatially variable velocity fields. Despite this it is possible to compare Equation 3.53 and its one-dimensional counterpart (Equation 3.24). The simple sums from both sides of the control volume in Equation 3.24 have been replaced by a full integral around the multi-faceted control volume. This can be broken down as:

$$-\frac{\Delta t}{\int_{\Omega_{v_i}} \mathrm{d}V} \underbrace{\int_{\partial\Omega_{v_i}} \tilde{u}_i n_i \hat{\alpha}^n|^{HO}_{\partial\Omega_{v_i}} \mathrm{d}s}_{\text{high-order advective flux term}} = \sum_{l \in \Omega_{v_i}} \left(\frac{\Delta t}{\int_{\Omega_{v_i}} \mathrm{d}V} \int_{\partial\Omega_{v_{f_l}}} |\tilde{u}_i n_i| I_{i_{f_l}} \hat{\alpha}^n|^{HO}_{\partial\Omega_{v_{f_l}}} \mathrm{d}s \right.$$
$$\left. - \frac{\Delta t}{\int_{\Omega_{v_i}} \mathrm{d}V} \int_{\partial\Omega_{v_{f_l}}} |\tilde{u}_i n_i| \left(1 - I_{i_{f_l}}\right) \hat{\alpha}^n|^{HO}_{\partial\Omega_{v_{f_l}}} \mathrm{d}s \right) \quad (3.54)$$

$$= \sum_{\Omega_e \cap i} \sum_{l \in \Omega_e} \left(\frac{\Delta t}{\int_{\Omega_{v_i}} \mathrm{d}V} \int_{\partial\Omega_{v_{f_l}}} |\tilde{u}_i n_i| I_{i_{f_l}} \hat{\alpha}^n|^{HO}_{\partial\Omega_{v_{f_l}}} \mathrm{d}s \right.$$
$$\left. - \frac{\Delta t}{\int_{\Omega_{v_i}} \mathrm{d}V} \int_{\partial\Omega_{v_{f_l}}} |\tilde{u}_i n_i| \left(1 - I_{i_{f_l}}\right) \hat{\alpha}^n|^{HO}_{\partial\Omega_{v_{f_l}}} \mathrm{d}s \right) \quad (3.55)$$

where, in Equation 3.54, $l \in \Omega_{v_i}$ is the set of control volume face indices belonging to the control volume Ω_{v_i}. Equation 3.55 produces the same result as Equation 3.54 but is displayed in the elemental form used for assembly, where $\Omega_e \cap i$ is the set of elements containing the node i and $l \in \Omega_e$ is the set of control volume face indices that lie within the element Ω_e.

Comparing Equation 3.54 with the high-order advective flux term in Equation 3.24 it can be seen that the equivalent multi-dimensional definition of the Courant number (Equation 3.5) for the face l around node i is:

$$c_{i_{f_l}} = \frac{\int_{\partial\Omega_{v_{f_l}}} |\tilde{u}_i n_i| \Delta t \mathrm{d}s}{\int_{\Omega_{v_i}} \mathrm{d}V}. \quad (3.56)$$

The HyperC face value method requires the Courant number in order to predict when downwinding will cause over or under-filling of the donor cell (by Equations 3.34).

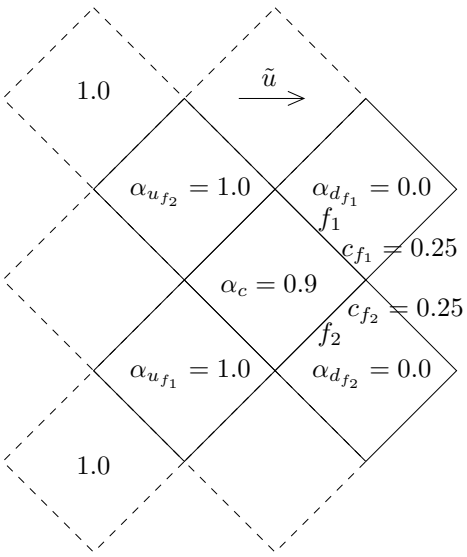


Figure 3.20: Scenario when limiting using the Courant number defined at a single face leads to unbounded behaviour.

An implementation in multiple dimensions could therefore use the projected upwind value definition discussed previously (Equation 3.42) and the Courant number at a face (Equation 3.56) in order to calculate a face value for face l according to Figure 3.8.

Despite being the most obvious extension to HyperC from one dimension, using the Courant number as defined in Equation 3.56 in order to calculate a face value is not generally bounded. This is because Equation 3.56 only considers a single face of the control volume. This is then used both to calculate the flux through face l and to estimate how much material is being transported from the upwind direction into the donor cell and hence whether over or under-filling will occur. However, there are an arbitrary number of faces on the control volume in multiple dimensions and estimating when boundedness will be violated based on a single face is not safe.

Consider the simplified quadrilateral donor control volume, α_c, in Figure 3.20. Given the direction of the velocity and the orientation of the cell, there are two faces, f_1 and f_2, with outgoing fluxes from α_c. The Courant number at each face, $c_{c_{f_1}}$ and $c_{c_{f_2}}$ using Equation 3.56, is taken as 1/4. The upwind values are known exactly from $\alpha_{u_{f_1}}$ and $\alpha_{u_{f_2}}$ and the values further upwind are smooth. This means that the value of α_c at the next time level, $n+1$, is given by:

$$\alpha_c^{n+1} = \alpha_c^n + c_{c_{f_1}}\alpha_{u_{f_1}} + c_{c_{f_2}}\alpha_{u_{f_2}} - c_{c_{f_1}}\hat{\alpha}_{f_1} - c_{c_{f_2}}\hat{\alpha}_{f_2}. \tag{3.57}$$

Taking the single face value of the Courant number (Equation 3.56) to define the downwinding region in HyperC (Equations 3.34), the normalised donor values, $\bar{\alpha}_{c_{f_1}} = \bar{\alpha}_{c_{f_2}} = 0.1$, suggest that the fluxes must be limited. Hence, the limited face values, $\hat{\alpha}_{f_1}$ and $\hat{\alpha}_{f_2}$, are both 0.6 ($\bar{\hat{\alpha}}_f = \bar{\alpha}_{c_{f_1}}/c_{c_f}$). Using Equation 3.57 the new donor value can be found:

$$\alpha_c^{n+1} = 0.9 + 0.25(1.0) + 0.25(1.0) - 0.25(0.6) - 0.25(0.6) = 1.1. \tag{3.58}$$

Clearly, this is unbounded.

In general the Courant number from a single face will be a poor estimate of the flux on the other side of the control volume and lead to unbounded behaviour. Instead, the flux may be considered over the entire control volume, i, giving an improved definition of the Courant number:

$$c_i = \frac{\sum_{l \in \Omega_{v_i}} \int_{\partial \Omega_{v_{f_l}}} |\tilde{u}_i n_i| \left(1 - I_{i_{f_l}}\right) \Delta t \mathrm{d}s}{\int_{\Omega_{v_i}} \mathrm{d}V}. \tag{3.59}$$

This is the normalised flux out of the entire control volume rather than across a single face.

Reconsidering the situation depicted in Figure 3.20, Equation 3.59 gives a Courant number of 1/2. Using this to define the downwinding region in HyperC (Equations 3.34) the fluxes through face f_1 and f_2 must still be adjusted, however the limited face values, $\hat{\alpha}_{f_1}$ and $\hat{\alpha}_{f_2}$, are now both 0.8 ($\bar{\hat{\alpha}}_f = \bar{\alpha}_{c_{f_1}}/c_c$). Using Equation 3.57 the new donor value can be found:

$$\alpha_c^{n+1} = 0.9 + 0.25(1.0) + 0.25(1.0) - 0.25(0.8) - 0.25(0.8) = 1.0. \tag{3.60}$$

Therefore, by considering the flux out of the entire control volume the cell remains bounded at the next timestep.

Using Equation 3.59 as the definition of the Courant number within HyperC effectively mitigates against a worst-case scenario. This occurs when the upwind value being used at the face currently being considered is actually being transported into the control volume at all incoming faces (as was seen in Figure 3.20). Additionally the current face may be the only face that is transporting material out of the control volume and therefore is the only one that can prevent over or under-filling of the cell. While being this cautious leads to bounded behaviour it also introduces extra diffusion into the solution, which may be unnecessary.

Consider the situation depicted in Figure 3.21. Again, a simple quadrilateral donor cell is considered, however this time not all the upwind faces are transporting the same amount of material into the cell. The normalised donor value across f_2, $\bar{\alpha}_{c_{f_2}}$, is infinite so upwinding must be used, $\hat{\alpha}_{f_2} = \alpha_c = 0.9$, while across f_1, $\bar{\alpha}_{c_{f_1}}$ is 0.1 so the face value requires limiting based on the Courant number. The result of this depends on the definition of the Courant number used to define the downwinding region (Equations 3.34). Using Equation 3.56, $c_{c_{f_1}} = 1/4$, $\hat{\alpha}_{f_1} = 0.6$ ($\bar{\hat{\alpha}}_{f_1} = \bar{\alpha}_{c_{f_1}}/c_{c_{f_1}}$) and:

$$\alpha_c^{n+1} = 0.9 + 0.25(1.0) + 0.25(0.0) - 0.25(0.6) - 0.25(0.9) = 0.775, \tag{3.61}$$

while using Equation 3.59, $c_c = 1/2$, $\hat{\alpha}_{f_1} = 0.8$ ($\bar{\hat{\alpha}}_{f_1} = \bar{\alpha}_{c_{f_1}}/c_c$) and:

$$\alpha_c^{n+1} = 0.9 + 0.25(1.0) + 0.25(0.0) - 0.25(0.8) - 0.25(0.9) = 0.725. \tag{3.62}$$

Both methods maintain a bounded solution, however using the Courant number defined at a single face (Equation 3.56) to define the downwinding region is less diffusive than the flux out of the entire control volume (Equation 3.59).

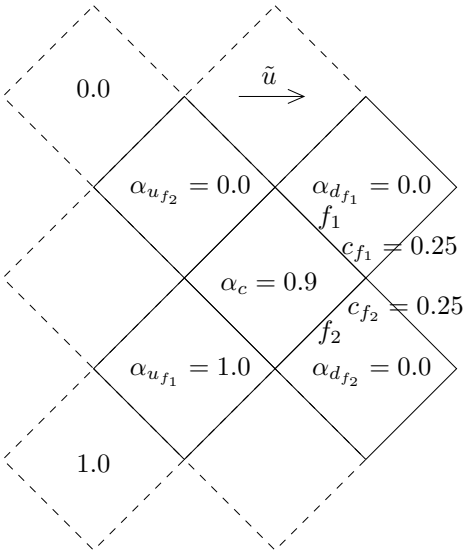


Figure 3.21: Scenario when limiting using the Courant number defined at a single face is the least diffusive option.

In general, the minimally diffusive definition of the Courant number for use within multi-dimensional HyperC lies somewhere between those given in Equations 3.56 and 3.59. For simplicity and to ensure boundedness only Equation 3.59 is used here. It is also the definition used whenever the Courant number is quoted.

Clearly the boundedness achieved with HyperC in multiple dimensions is also dependent on the definition of the upwind value. As was seen in one dimension, this serves a triple role: it is used in the detection of local extrema, as an estimate of the incoming upwind flux and as the definition of boundedness which a cell aims for when limiting.

HyperC optimally advects step functions in one dimension because the upwind value, in combination with the Courant number, gives the upwind flux correctly. In multiple dimensions however this assumption no longer holds as there are a potentially arbitrary number of faces propagating information into the donor cell, rather than just one.

The interface, which in one dimension was defined by a single mixed cell, is less clearly defined in multiple dimensions. Depending on the direction from which a face 'views' the interface it may appear to be smeared over multiple cells. For example, a face aligned nearly perpendicular to the interface may 'see' two or more mixed cells ahead of it while a face attached to the same control volume aligned parallel to the interface may 'see' only one. For this reason the upwind flux is less predictable in higher dimensions. This is further complicated by the fact that, as discussed already, on a control volume with an arbitrary number of faces there is no upwind face through which to predict a single upwind flux.

The lack of an opposing upwind face also means that the upwind value used to detect extrema is undefined. In multiple dimensions a local extremum at a particular node may be detected by comparing its value to the minimum and maximum of its neighbours. Using this definition however leads to excessively diffusive behaviour as cells

within the mixed interface zone may appear unbounded relative to their nearest, also mixed, neighbours. This results in upwinding through all the out-flowing faces exactly where HyperC should aim to prevent it.

Using the definition of the upwind value projected off the donor node (Equation 3.42) introduces a degree of directionality into extremum detection. Therefore a node in the mixed zone may appear to be a local extremum along the interface; however, it is less likely to appear to be one across the interface. This increases the chances that downwinding will continue for as long as possible across the interface, minimising the mixed region. As the worst-case scenario definition of the Courant number is being used (Equation 3.59), the projected upwind value may also be used to estimate the upwind flux.

A single definition of the upwind value, for both the flux and sensor variables, makes the implementation of HyperC in multiple dimensions much simpler, allowing each face to be considered as if it were a one-dimensional problem. Some intervention however is required to ensure that the projected upwind value is a suitable target value during limiting. Fortunately this simply involves capping the value from Equation 3.42 so that it is itself bounded within the solution values in the upwind element, Ω_{e_u}:

$$\alpha_{u_k} = \max \left(\begin{array}{c} \min \left(\begin{array}{c} \alpha_{u_k} \\ \max \left(\alpha \in \Omega_{e_u} \right) \end{array} \right) \\ \min \left(\alpha \in \Omega_{e_u} \right) \end{array} \right). \tag{3.63}$$

Without this modification a flux may be calculated that targets an unbounded value, resulting in over or undershoots in the solution.

Using Equation 3.59 and a bounded upwind value from Equation 3.42 it is therefore possible to simply extend HyperC to the control volume mesh in multiple dimensions without operator splitting. In the subsequent sections standard material tracking tests are performed to compare HyperC on an unstructured mesh against structured mesh material tracking methods. In all the examples given HyperC gave bounded solutions without additional intervention.

Unidirectional Advection

Rudman (1997) suggested a number of simple tests of material tracking algorithms. These aim to measure how well a method maintains a step function between zero and one when advecting different shapes across a domain using the linear advection equation. The tests were originally performed on all of the interface reconstruction algorithms discussed at the start of Section 3.3 (SLIC-VOF, HN-VOF, FCT-VOF and Y-VOF) on two-dimensional structured meshes using operator splitting. Some intervention to maintain boundedness was reported by Rudman (1997), who reset values that became negative to zero and values that exceeded unity back to one. The experiments are repeated here using HyperC on two-dimensional unstructured control volume meshes without operator splitting or bounding intervention.

Three different initial configurations of the advected field are considered:

- a hollow square, aligned with the coordinate axes
- a rotated hollow square, at an angle of 26.57° to the x_1-axis
- a hollow circle

as shown in Figure 3.22(a)(i–iii). Contours are shown at field values of 0.025, 0.5 and 0.975 to demonstrate the extent of the mixed zone.

The original structured mesh consisted of 200×200 cells with faces orientated along the coordinate axes. Each shape was initially a total of 40 cells wide with a distance of 10 cells between the inner and outer interfaces. Here a parent unstructured mesh was constructed in an equivalent square domain, Ω: $x_1 \in [-2, 2]$, $x_2 \in [-2, 2]$, with an average edge length of 0.02. The shapes were initially 0.8 units wide with a distance of 0.2 units between the inner and outer interfaces.

These configurations were each advected according to two unidirectional velocity fields:

- along the x_1-axis using $\tilde{u} = (1, 0)$
- at an angle to the coordinates using $\tilde{u} = (2, 1)$

for 500 timesteps at a maximum Courant number of 0.25. On the unstructured mesh this is defined using Equation 3.59 but ignoring the boundary nodes which have a higher Courant number due to their diminished area.

Using such simple velocity fields the analytical solution may be represented on the mesh, α^e, and the error, E, quantified. Rudman (1997) defined E such that:

$$E = \frac{\sum_i |\alpha_i - \alpha_i^e|}{\sum_i \alpha_{0i}}, \tag{3.64}$$

where α_i is the final solution and α_{0i} is the initial condition at node i. This was generalised here to take account of the variable cell sizes on the unstructured mesh. Hence E becomes:

$$E = \frac{\int_\Omega |\alpha - \alpha^e| \mathrm{d}V}{\int_\Omega \alpha_0 \mathrm{d}V}. \tag{3.65}$$

On a structured mesh with equal cell sizes Equation 3.65 gives the same result as Equation 3.64 so the results are still directly comparable.

These results are presented in Figure 3.22(b) using HyperC on an unstructured mesh (Figure 3.22(c)) with $\tilde{u} = (1, 0)$. The absolute difference, $|\alpha - \alpha^e|$, between the exact and numerical solutions is also shown in Figure 3.22(d). For comparison the structured mesh results from Rudman (1997) are also presented in Figure 3.23. A comparison of the errors for each algorithm is given in Table 3.3.

Examining Figure 3.22 HyperC can be seen to have preserved each of the shapes extremely well without excessive distortion or the introduction of flotsam and jetsam. In comparison, HN-VOF (Figure 3.23(c)) displays significant distortion on the interfaces not aligned with the coordinate axes while FCT-VOF (Figure 3.23(d)) appears to have introduced small pockets of non-physical material. Due to the limited resolution of the

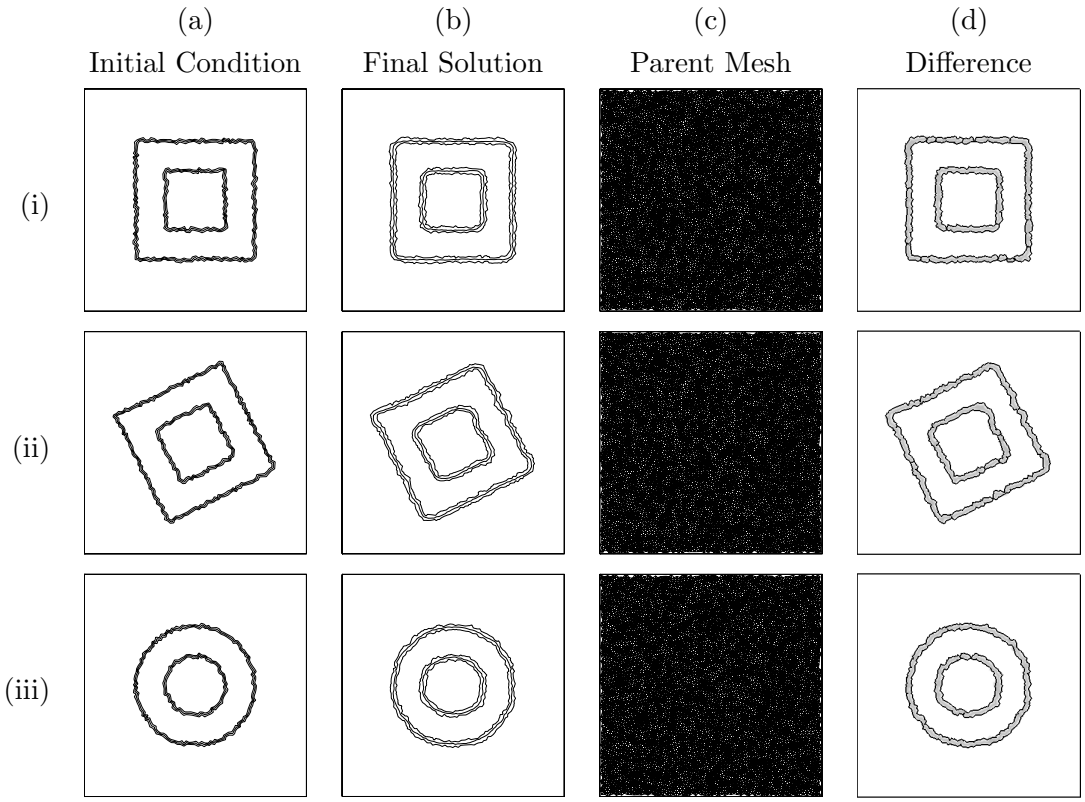


Figure 3.22: Advection of a hollow square (i), a rotated hollow square (ii) and a hollow circle (iii) on unstructured meshes using HyperC with $\tilde{u} = (1, 0)$. Contours are shown at values of 0.025, 0.5 and 0.975 for the initial condition (a) and the final result after 500 timesteps (b). The parent simplex mesh in the vicinity of the final result (c) is shown alongside the absolute difference, $|\alpha - \alpha^e|$, between the numerical and exact solutions (d). Differences greater than the 0.025 contour are shaded grey.

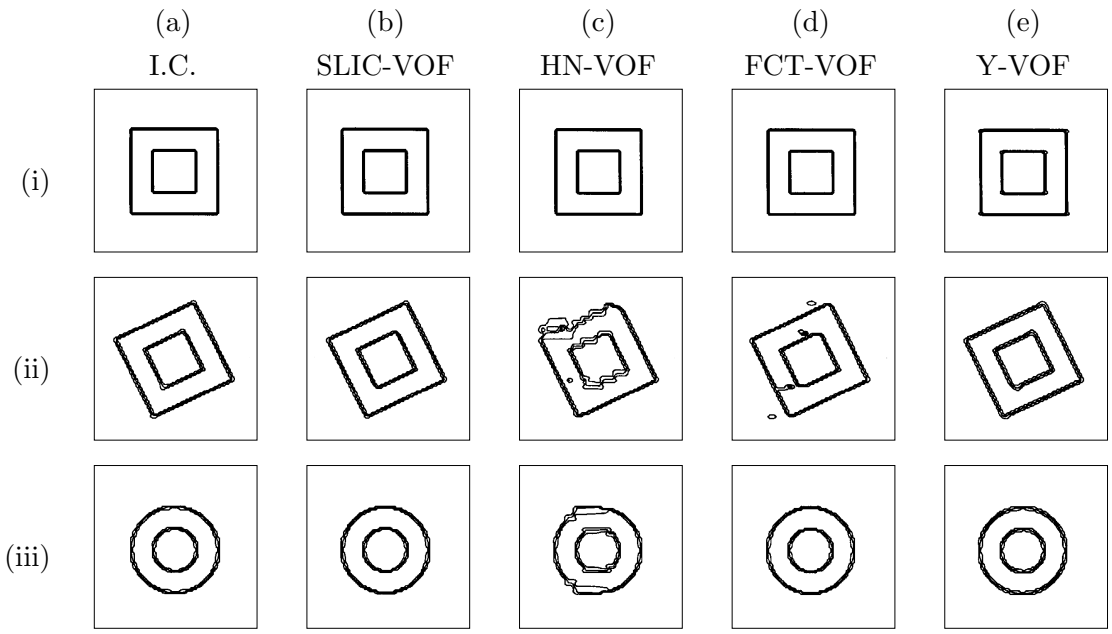


Figure 3.23: Advection of a hollow square (i), a rotated hollow square (ii) and a hollow circle (iii) on a structured mesh using operator splitting and $\tilde{u} = (1, 0)$. The initial condition, I.C. (a), and results after 500 timesteps using SLIC-VOF (b), HN-VOF (c), FCT-VOF (d) and Y-VOF (e) are shown. All images are taken from Rudman (1997) and show contours at values of 0.025, 0.5 and 0.975.

Shape	SLIC-VOF	HN-VOF	FCT-VOF	Y-VOF	HyperC
Square	8.42×10^{-8}	1.03×10^{-8}	3.89×10^{-8}	1.08×10^{-3}	5.85×10^{-2}
Rotated Square	1.46×10^{-2}	6.91×10^{-2}	2.32×10^{-2}	5.35×10^{-3}	6.77×10^{-2}
Circle	1.30×10^{-2}	4.55×10^{-2}	1.28×10^{-2}	3.08×10^{-3}	6.78×10^{-2}

Table 3.3: Errors (E by Equation 3.65) for simple advection by the velocity field $\tilde{u} = (1, 0)$. Results for SLIC-VOF, HN-VOF, FCT-VOF and Y-VOF use operator splitting on a structured mesh and are taken from Rudman (1997). HyperC results use fully two-dimensional fluxes on an unstructured mesh.

images taken from Rudman (1997) it is hard to tell how much smearing of the interface has been introduced during advection although excessive smearing will lead to a higher error.

Comparing the errors of the methods in Table 3.3 it can be seen that for shapes aligned with the coordinate axes the VOF methods produce superior results to HyperC. This is because with operator splitting these methods are in essence solving a one-dimensional problem as there is no transport in the x_2 direction. Due to the arbitrary alignment of the control volume faces HyperC is always solving a two-dimensional problem.

The advantage the structured split methods have over HyperC diminishes rapidly on shapes that are not aligned with the axes. HyperC produces errors that are of the same order of magnitude as SLIC-VOF, HN-VOF and FCT-VOF for the hollow circle and rotated square. Y-VOF, with its complex interface reconstruction method, produces better results than any other method on these shapes.

The results for advection with $\tilde{u} = (2, 1)$ using HyperC on an unstructured mesh (Figure 3.24(c)) are presented in Figure 3.24(b). Again, the absolute difference between the exact and numerical solutions is shown in Figure 3.24(d) while the structured mesh results from Rudman (1997) are presented in Figure 3.25. A comparison of the errors for each algorithm is given in Table 3.4.

This time HyperC can be seen to introduce some distortion of the interface, wrinkling the upper and lower boundaries of the hollow square (Figure 3.24(b)(i)) and distorting the curves of the circle travelling parallel to the direction of advection (Figure 3.24(b)(iii)). The rotated square, which is travelling approximately face on to the velocity, is largely unaffected (Figure 3.24(b)(ii)).

Similar or worse distortion can be seen in the structured mesh results (Figure 3.25). SLIC-VOF introduces significant wrinkling of the interface in all cases. HN-VOF distorts the shapes not aligned with the coordinate axes. FCT-VOF warps the faces of the rotated square travelling perpendicular to the advection direction. Y-VOF is again the best structured method, only smoothing the corners of the square shapes.

Comparing the errors (Table 3.4) HyperC outperforms the structured mesh methods in many of the cases. In all the others, with the exception of particularly low errors for HN-VOF and FCT-VOF when advecting the square, the order of magnitude is comparable.

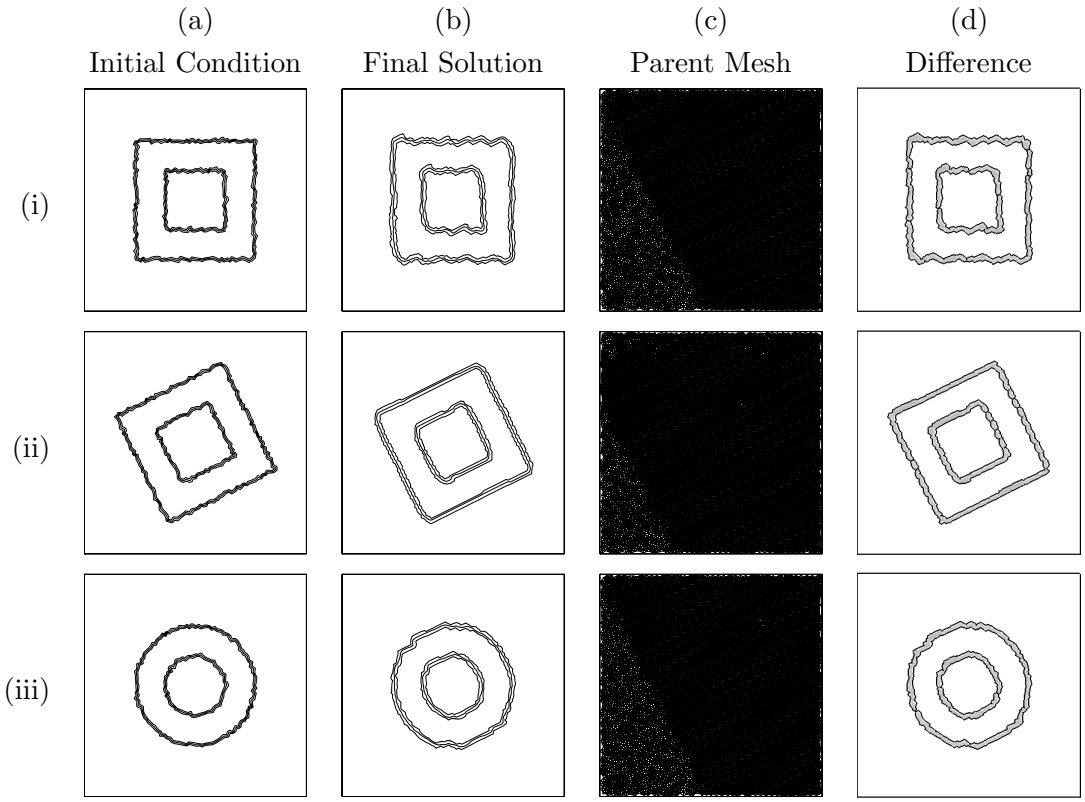


Figure 3.24: Advection of a hollow square (i), a rotated hollow square (ii) and a hollow circle (iii) on unstructured meshes using HyperC with $\tilde{u} = (2, 1)$. Contours are shown at values of 0.025, 0.5 and 0.975 for the initial condition (a) and the final result after 500 timesteps (b). The parent simplex mesh in the vicinity of the final result (c) is shown alongside the absolute difference, $|\alpha - \alpha^e|$, between the numerical and exact solutions (d). Differences greater than the 0.025 contour are shaded grey.

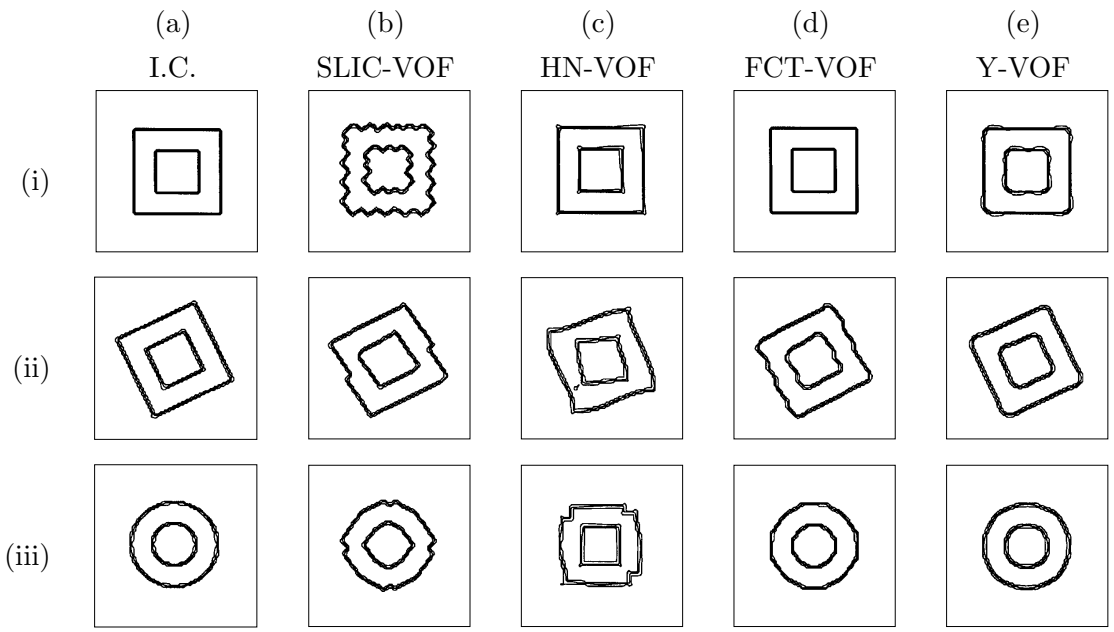


Figure 3.25: Advection of a hollow square (i), a rotated hollow square (ii) and a hollow circle (iii) on a structured mesh using operator splitting and $\tilde{u} = (2, 1)$. The initial condition (a) and results after 500 timesteps using SLIC-VOF (b), HN-VOF (c), FCT-VOF (d) and Y-VOF (e) are shown. All images are taken from Rudman (1997) and show contours at values of 0.025, 0.5 and 0.975.

Shape	SLIC-VOF	HN-VOF	FCT-VOF	Y-VOF	HyperC
Square	1.32×10^{-1}	6.86×10^{-3}	1.63×10^{-8}	2.58×10^{-2}	7.60×10^{-2}
Rotated Square	1.08×10^{-1}	1.60×10^{-1}	8.15×10^{-2}	3.16×10^{-2}	6.40×10^{-2}
Circle	9.18×10^{-2}	1.90×10^{-1}	3.99×10^{-2}	2.98×10^{-2}	7.43×10^{-2}

Table 3.4: Errors (E by Equation 3.65) for simple advection by the velocity field $\tilde{u} = (2, 1)$. Results for SLIC-VOF, HN-VOF, FCT-VOF and Y-VOF use operator splitting on a structured mesh and are taken from Rudman (1997). HyperC results use fully two-dimensional fluxes on an unstructured mesh.

Tables 3.3 and 3.4 demonstrate how dependent operator split structured mesh methods are on the geometry of the problem being solved. The errors are lowest when the interface is aligned either with the mesh faces or the advection direction. Only the most expensive and accurate interface reconstruction technique, Y-VOF, shows a certain consistency in its errors regardless of the problem.

As there is no alignment between the mesh and either the velocity field or the interface, the errors produced using HyperC on an unstructured mesh are largely problem independent. For instance the advection of the square along the x_1-axis is effectively the same problem as advecting the rotated square using $\tilde{u} = (2, 1)$. While on the structured mesh these two tests produced highly variable results, the errors using HyperC were very similar.

Since in physical scenarios it is unlikely that interfaces will align with either the advection direction or the coordinate axes, the large discrepancy in the errors between HyperC and the structured meshes for such simple cases can be discounted. Instead, the broad similarity in the errors in more general cases suggests that HyperC provides a suitable method for advecting step functions on unstructured meshes.

Solid Body Rotation

A more complicated advection problem involving solid body rotation was suggested by Zalesak (1979) and is widely used to test material advection algorithms. A circle with a slot removed from it is rotated through a complete revolution and the error is computed at the time the exact solution would return to its original position.

Rudman (1997) presented results from this test using the same four VOF methods as discussed above. Again a structured mesh with 200×200 cells was used over a domain, Ω: $x_1 \in [-2, 2]$, $x_2 \in [-2, 2]$. A circle with value 1 internally, a diameter of 50 cells and a vertical slot of width 6 cells removed from its base was centred at $(2, 2.75)$ (see Figure 3.28(a)). This was then advected for a full rotation of 2,524 timesteps at a Courant number of 0.25 under the velocity field $\tilde{u} = (x_2/2, x_1/2)$.

Here an unstructured parent mesh with an average edge length of 0.02 was used over the same domain, Ω (see Figure 3.26(b)). The circle was 1 unit in diameter and the slot had a width of 0.12 units and a length of 0.6 units (see Figure 3.26(a)). It was

SLIC-VOF	HN-VOF	FCT-VOF	Y-VOF	HyperC
8.38×10^{-2}	9.62×10^{-2}	3.29×10^{-2}	1.09×10^{-2}	5.21×10^{-2}

Table 3.5: Errors for the Zalesak slot problem after a single full rotation. Results for SLIC-VOF, HN-VOF, FCT-VOF and Y-VOF use operator splitting on a structured mesh and are taken from Rudman (1997). HyperC results use fully two-dimensional fluxes on an unstructured mesh.

Limited Trapezoidal		HyperC	
1^{st} Rotation	2^{nd} Rotation	1^{st} Rotation	2^{nd} Rotation
2.97×10^{-1}	3.44×10^{-1}	5.21×10^{-2}	5.97×10^{-2}

Table 3.6: Errors for the Zalesak slot problem after one and two full rotations using a Sweby limited trapezoidal face value method and HyperC. Errors using other face value methods are presented in Table B.1.

centred at the same point and advected by the same velocity field at a maximum Courant number of 0.25. Using Equation 3.59 as the definition of the Courant number, and again ignoring the boundary cells, meant that this took 5,048 timesteps to complete a single revolution. This suggests that the Courant number as quoted by Rudman (1997) is a simple one-dimensional definition (Equation 3.5). Taking into account operator splitting on the structured mesh the number of advective steps on both meshes are the same.

The results using HyperC on the control volume dual mesh are presented in Figure 3.26(d) while Figure 3.27(b) shows the absolute difference between these results and the exact solution. Both these figures indicate that after both one (Figures 3.26(d)(i), 3.27(b)(i)) and two (Figures 3.26(d)(ii), 3.27(b)(ii)) rotations the original shape and narrow slot are still clearly defined. Some distortion has occurred on the upper boundary where the Courant number is highest. Also, the corners of the slot are slightly blunted after advection. For comparison the results using a Sweby limited trapezoidal face value are also presented in Figures 3.26(c) and 3.27(a). These demonstrate how diffusive and unsuited such methods are to material advection.

Figure 3.28(b–e) shows the same test after a single rotation as performed by Rudman (1997) using SLIC-VOF, HN-VOF, FCT-VOF and Y-VOF. Both SLIC-VOF and HN-VOF produce significant distortion of the interface and generate flotsam and jetsam. The results using FCT-VOF show some distortion while Y-VOF gives the best results with only smoothing of the slot corners visible.

The error between the exact and final solutions can again be compared using Equation 3.65, although in this case $\alpha^e = \alpha_0$. The errors evaluated using HyperC are presented and compared to the structured mesh algorithms in Table 3.5. Again HyperC is found to be roughly comparable to the VOF methods, giving the same order of magnitude error and lying between HN-VOF and FCT-VOF in absolute terms.

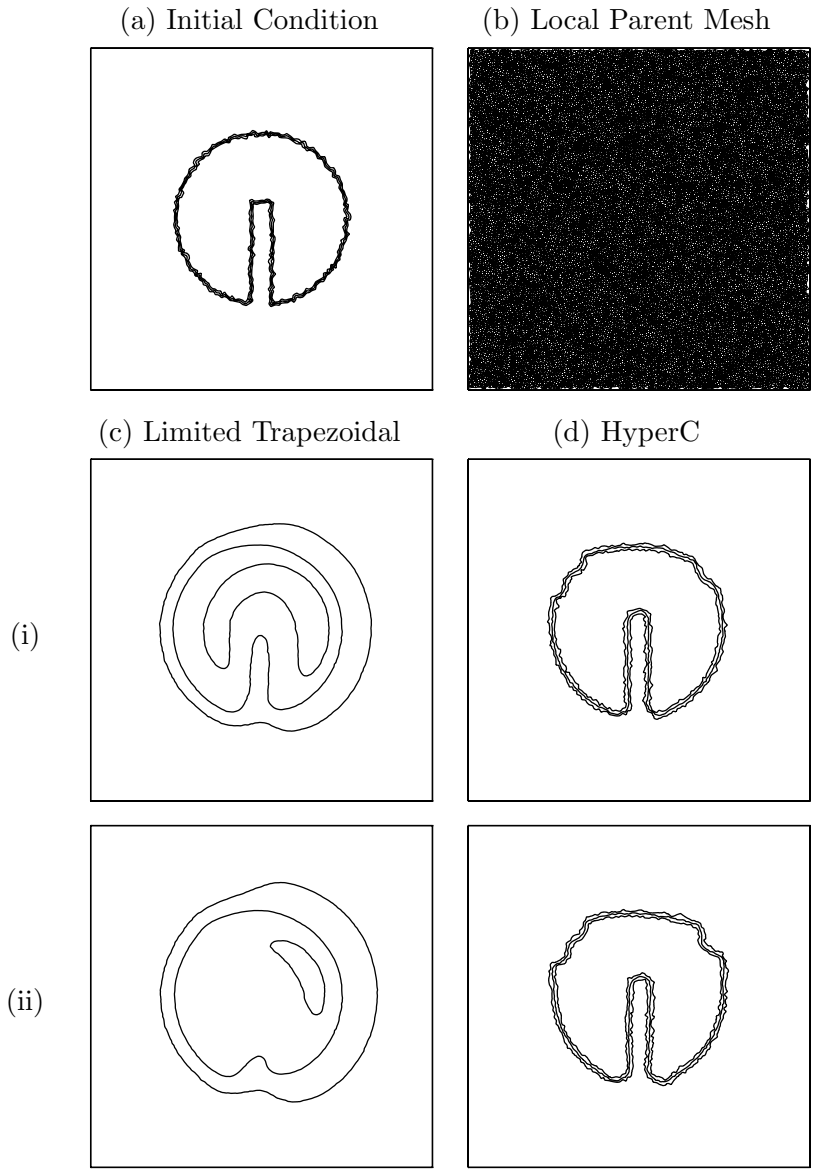


Figure 3.26: Rotation of a slotted circle (a) on an unstructured mesh (b) using a Sweby limited trapezoidal face value (c) and HyperC (d). Results after one (i) and two (ii) rotations are shown using contours at values of 0.025, 0.5 and 0.975.

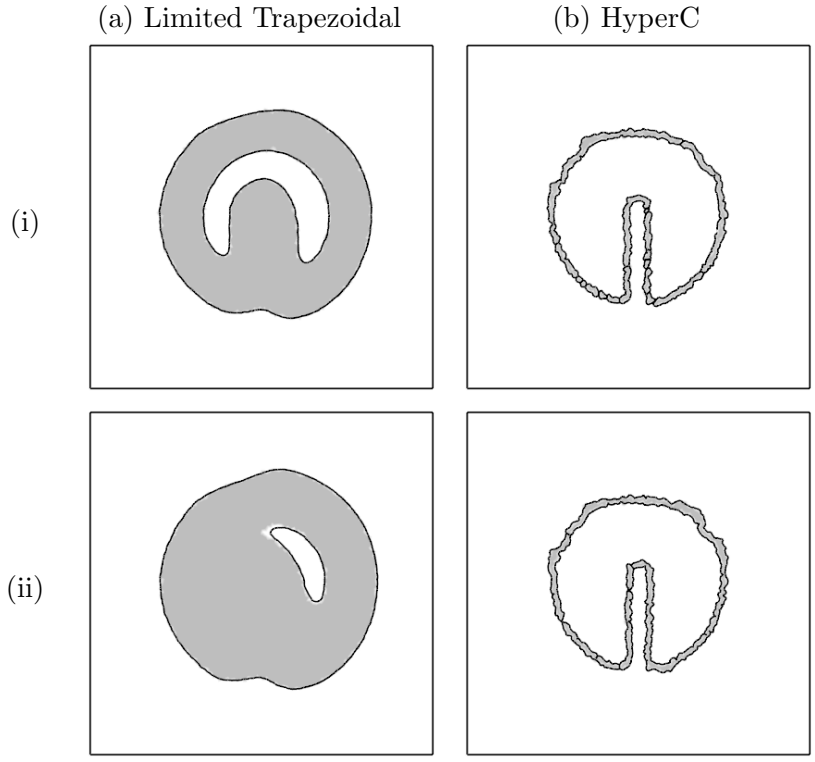


Figure 3.27: The absolute difference, $|\alpha - \alpha^e|$, between the exact solution (see Figure 3.26(a)) and the numerical solutions using a Sweby limited trapezoidal face value method (a, see Figure 3.26(c)) and HyperC (b, see Figure 3.26(d)). Values above the 0.025 contour are shaded grey after one (i) and two (ii) rotations.

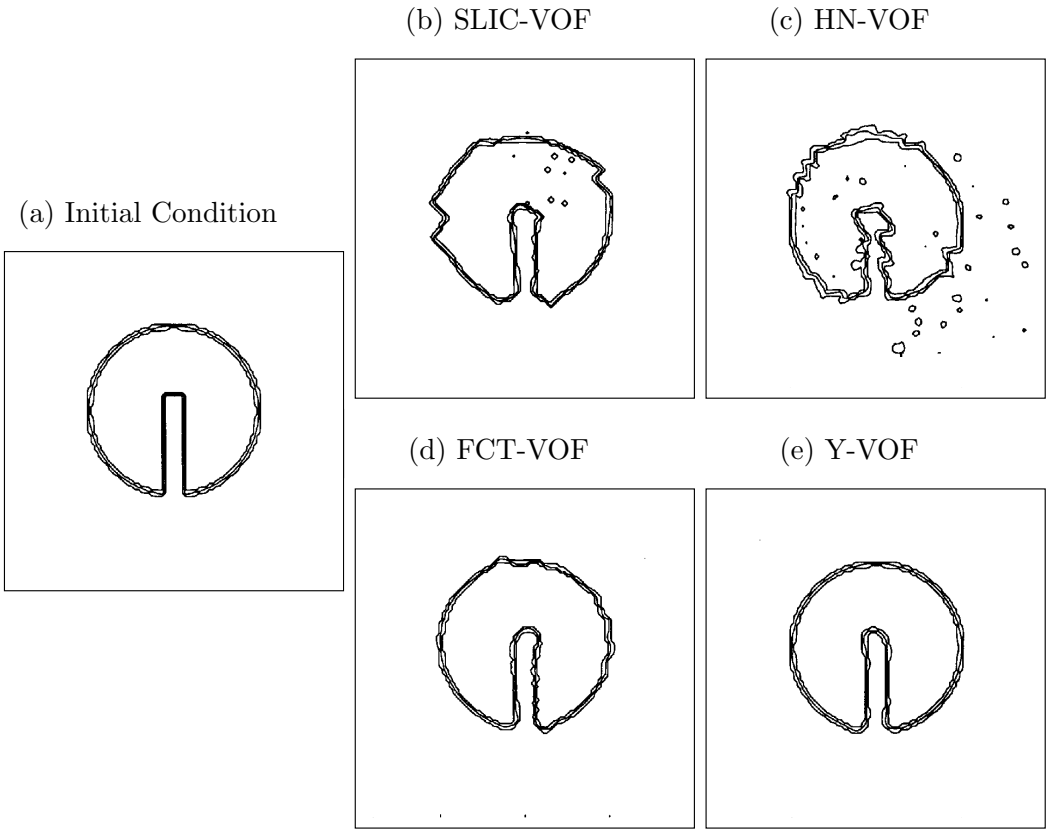


Figure 3.28: Rotation of a slotted circle on a structured mesh using operator splitting. The initial condition (a) and results using SLIC-VOF (b), HN-VOF (c), FCT-VOF (d) and Y-VOF (e) are shown after one rotation. All images are taken from Rudman (1997) and show contours at values of 0.025, 0.5 and 0.975.

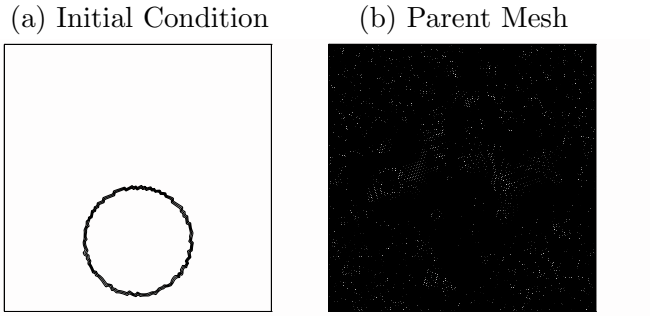


Figure 3.29: The initial condition (a) and unstructured parent mesh, containing 10,022 cells (b), used in the shearing flow advection tests.

The errors after the first and second rotation using HyperC are also shown in Table 3.6. This demonstrates that while the error increases slightly the change is not dramatic. For comparison the errors calculated using the Sweby limited trapezoidal method are also given after one and two rotations.

Shearing Flow

The results of the Zalesak slot problem demonstrate again that HyperC on an unstructured mesh is capable of performing as well as interface reconstruction methods on a structured mesh. However, Rider & Kothe (1995) pointed out that simple translation and rotation tests provide an incomplete assessment of material tracking methods. They suggested that a shearing flow, which causes topological change in the solution, would provide a better test of the algorithms. Such flows are closer to a physical situation in which stretching and break up as well as merging of the material are all possible.

Rudman (1997) suggested that a circle, with initial radius of $\pi/5$, should be advected by the velocity field $\tilde{u} = (\sin(x_1)\cos(x_2), -\cos(x_1)\sin(x_2))$ around the domain, Ω: $x_1 \in [0,\pi]$, $x_2 \in [0,\pi]$. In the original structured mesh test the domain was discretised using 100×100 cells. Here the average edge length of the unstructured parent mesh is set to $\pi/100$ and the circle is initially centred at $(\pi/2, (1+\pi)/5)$. The initial condition and parent mesh are shown in Figure 3.29.

The circle is advected forward N timesteps at a maximum Courant number of 0.25 before the flow is reversed, $\tilde{u} = (-\sin(x_1)\cos(x_2), \cos(x_1)\sin(x_2))$, and the now sheared circle is advected back for the same number of timesteps as it was advected forward, N. This should return the field to its initial condition and the errors can be calculated using Equation 3.65 with $\alpha^e = \alpha_0$.

Rudman (1997) presented results using SLIC-VOF, HN-VOF, FCT-VOF and Y-VOF after N = 250, 500, 1,000 and 2,000 steps for forward and reverse advection. Due to the discrepancies in the definitions of the Courant number on the structured and unstructured meshes, results are given here for N = 250, 500, 1,000, 2,000, 3,000 and 4,000 timesteps. Visual inspection of the forward advection results suggests that the nearest corresponding result to 2,000 timesteps on the structured mesh is 4,000 steps on

N	Trapezoidal	HyperC
250	1.10×10^{-1}	2.72×10^{-2}
500	1.36×10^{-1}	3.00×10^{-2}
1,000	1.78×10^{-1}	3.77×10^{-2}
2,000	2.91×10^{-1}	6.50×10^{-2}
3,000	4.29×10^{-1}	1.16×10^{-1}
4,000	5.60×10^{-1}	1.64×10^{-1}

Table 3.7: Errors between the 'exact' solution (see Figure 3.30(c)) and the numerical solutions using a Sweby limited trapezoidal face value and HyperC after N timesteps of forward advection.

its unstructured equivalent.

The results from the forward advection on the unstructured mesh are presented in Figure 3.30 and show that HyperC maintains the continuity of the sheared circle well until about 3,000 timesteps when mixed cells with low values start trailing the main tail of the flow. Until this point the numerical results also closely mirror the anticipated 'exact' solutions. These are provided in Figure 3.30(c) and show the results using the method of characteristics, numerically integrated to a tolerance of 10^{-12}. Using these results as α^e, Table 3.7 shows the errors in the numerical solutions by Equation 3.65. These are consistently low until about 3,000 timesteps when the shedding of low value cells from the tail begins. This can be seen particularly clearly in Figure 3.31 where the absolute difference between the numerical and 'exact' solutions is shown.

Forward advection using HyperC compares well visually with the results in Figure 3.32: SLIC-VOF and HN-VOF generate significant quantities of flotsam and jetsam after only 1,000 timesteps, which is not seen at all in the HyperC results. FCT-VOF shows some interface distortion and also develops a trail of low value cells behind the main tail. Y-VOF performs best, preserving a smooth boundary and not generating significant trailing cells.

The corresponding results from the reverse advection on the unstructured mesh are presented in Figures 3.33 and 3.34. HyperC successfully returns to a reasonable approximation of the initial condition up to about 3,000 timesteps. After this point the previously trailing mixed cells can be seen around the boundary causing disruption and

Figure 3.30: *See next page.* Results for the forward advection of Figure 3.29(a) in a shearing flow, $\tilde{u} = (\sin(x_1)\cos(x_2), -\cos(x_1)\sin(x_2))$, after $N = 250$ (i), 500 (ii), 1,000 (iii), 2,000 (iv), 3,000 (v) and 4,000 (vi) timesteps using a Sweby limited trapezoidal face value (a) and HyperC (b). Contours are shown at values of 0.025, 0.5 and 0.975. The 'exact' solution, found by numerically integrating the characteristic equations to a tolerance of 10^{-12}, is also shown.

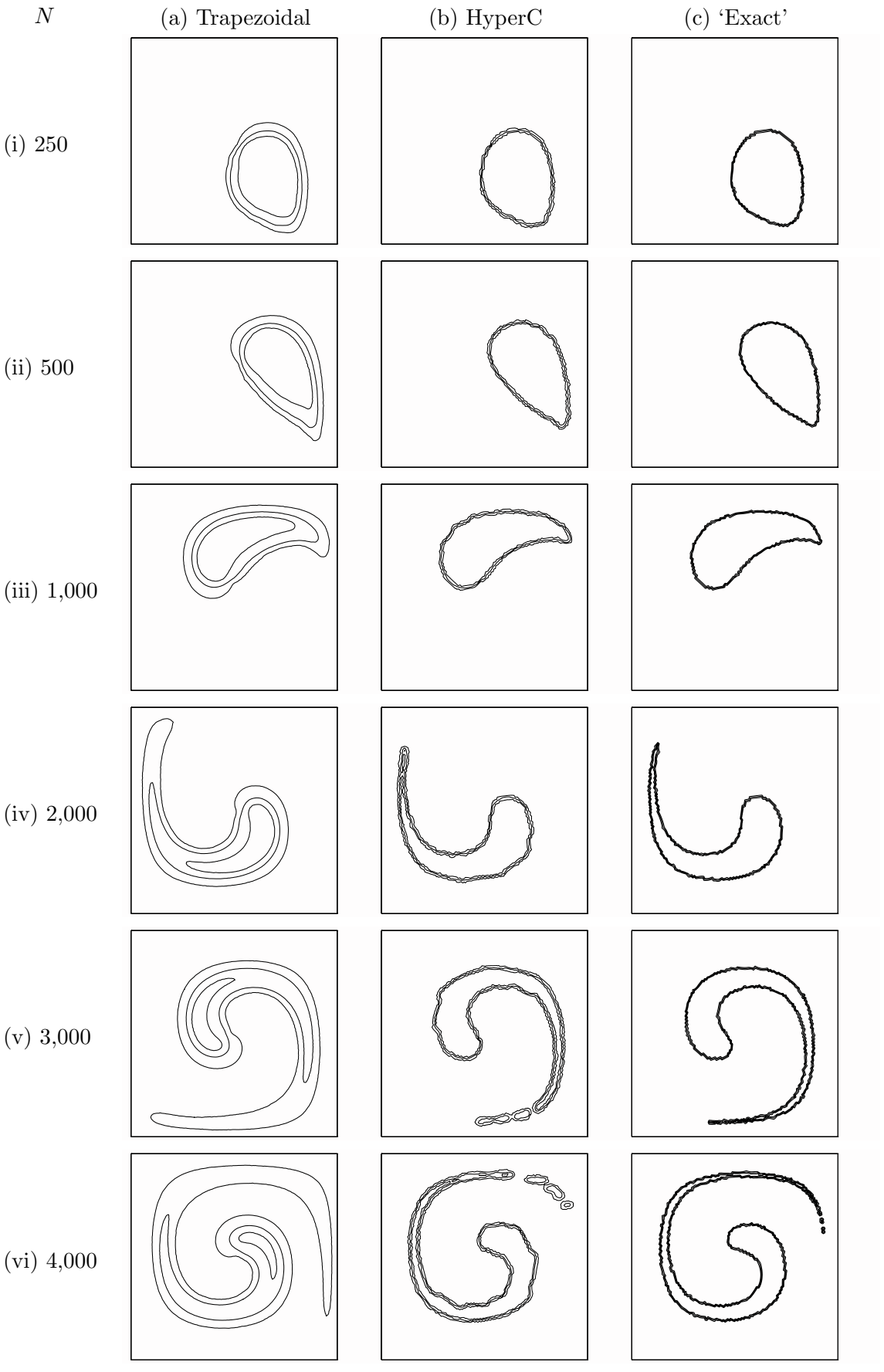


Figure 3.30: *cont.*

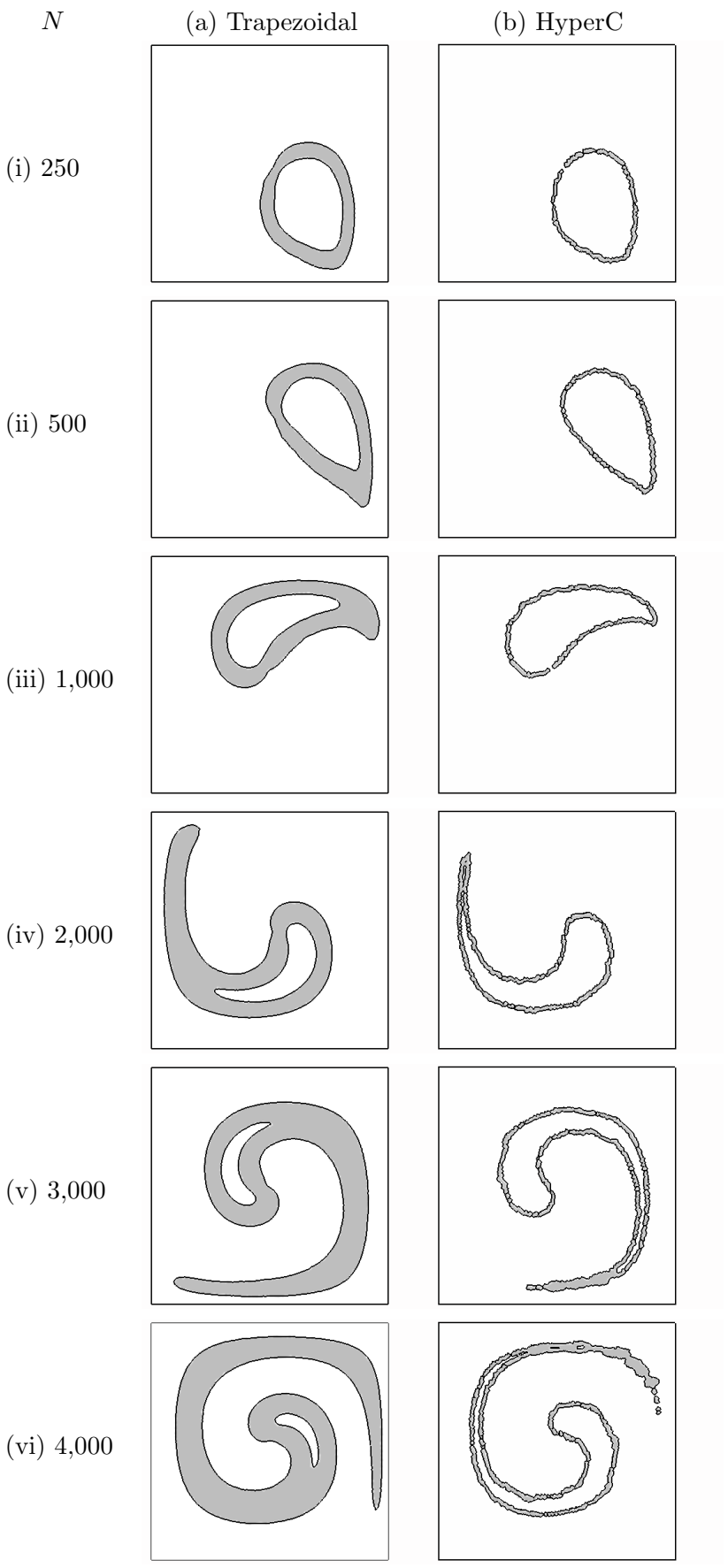


Figure 3.31: *See next page.*

Figure 3.31: *cont.* The absolute difference, $|\alpha - \alpha^e|$, between the 'exact' solution (see Figure 3.30(c)) and the numerical solutions using a Sweby limited trapezoidal face value method (a, see Figure 3.30(a)) and HyperC (b, see Figure 3.30(b)). Values above the 0.025 contour are shaded grey. The results are shown after $N = 250$ (i), 500 (ii), 1,000 (iii), 2,000 (iv), 3,000 (v) and 4,000 (vi) timesteps of forward advection.

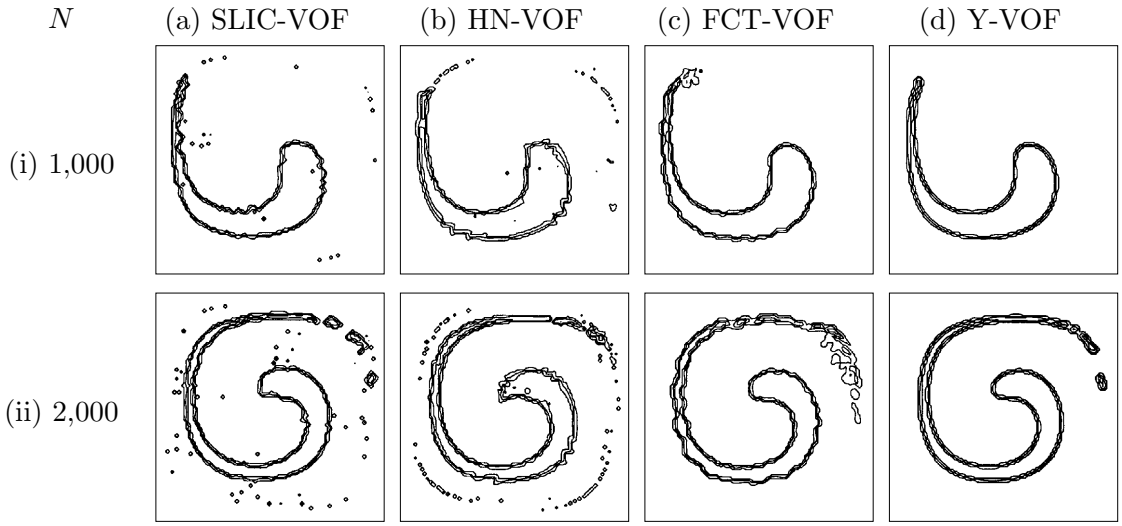


Figure 3.32: Results for the forward advection in a shearing flow, $\tilde{u} = (\sin(x_1)\cos(x_2), -\cos(x_1)\sin(x_2))$, after $N = 1{,}000$ (i) and 2,000 (ii) timesteps using SLIC-VOF (a), HN-VOF (b), FCT-VOF (c) and Y-VOF (d). All images are taken from Rudman (1997) and show contours at values of 0.025, 0.5 and 0.975.

distortion of the interface.

Again, this compares well with the corresponding structured results in Figure 3.35. The flotsam and jetsam produced by SLIC-VOF and HN-VOF have completely contaminated the solution. The mixed cells generated by FCT-VOF have similarly contaminated the final solution with evidence of their existence present in both the 1,000 and 2,000 timestep results. Y-VOF shows minimal disruption with only a small deviation after 2,000 timesteps.

Table 3.8 compares the errors between the final and initial solutions for both the structured and unstructured mesh results. HyperC can be seen to be comparable to all the methods, with the same order of magnitude error as seen using interface reconstruction. The only exception to this is with Y-VOF which has particularly low errors for small numbers of timesteps ($N \leq 1,000$).

All methods accumulate increasing errors with increasing numbers of timesteps. As shearing continues the tail of the deformed circle begins to be thinner than the available mesh resolution. This leads to the formation of separate isolated patches with all the algorithms. How each method deals with these under-resolved regions dictates whether

N	SLIC-VOF	HN-VOF	FCT-VOF	Y-VOF	HyperC
250	2.72×10^{-2}	3.24×10^{-2}	1.94×10^{-2}	2.61×10^{-3}	2.14×10^{-2}
500	3.30×10^{-2}	4.00×10^{-2}	2.35×10^{-2}	5.12×10^{-3}	2.67×10^{-2}
1,000	4.59×10^{-2}	6.66×10^{-2}	3.14×10^{-2}	8.60×10^{-3}	3.36×10^{-2}
2,000	9.02×10^{-2}	1.09×10^{-1}	1.44×10^{-1}	3.85×10^{-2}	4.03×10^{-2}
3,000					6.23×10^{-2}
4,000					8.06×10^{-2}

Table 3.8: Errors for a shearing flow which is reversed after N timesteps and compared to the initial condition. Results for SLIC-VOF, HN-VOF, FCT-VOF and Y-VOF use operator splitting on a structured mesh and are taken from Rudman (1997). HyperC results use fully two-dimensional fluxes on an unstructured mesh. Errors for other face value methods and at various resolutions and Courant numbers are presented in Tables B.2-B.3.

they form smeared mixed cells or more coherent clumps. Flux limiting methods that adjust which behaviour is achieved on an unstructured mesh will be discussed in the next section.

Figures 3.30, 3.31, 3.33 and 3.34 also display the results using a Sweby limited trapezoidal face value method. If the field being advected was a volume fraction then this method would result in a particularly high degree of mixing between the materials – the uppermost contour (0.975) has disappeared in all but the shortest simulations. This point is emphasised by comparing the fractional area of the domain that is filled with mixed cells when using a limited trapezoidal method and HyperC. This is shown in Figure 3.36 for two definitions of mixed cells: those which have values that fall between the uppermost and lowermost contours, 0.025 and 0.975 (Figure 3.36(a)), and those that have values that fall between 10^{-8} and $1 - 10^{-8}$. By either measure HyperC is seen to produce far fewer mixed cells, as is desired in a material tracking algorithm.

All the advection tests for translation, rotation and shearing flow show that HyperC on an unstructured mesh is comparable to the operator split structured mesh VOF methods. Additionally, it never generated flotsam or jetsam and the magnitude of its errors are independent of the problem set-up. It is therefore proposed that it offers a suitable method for advecting materials on unstructured meshes where operator splitting and interface reconstruction are not practical.

Figure 3.33: *See next page.* Results for the reverse advection of Figure 3.30(a–b)(i–vi) in a shearing flow, $\tilde{u} = (-\sin(x_1)\cos(x_2), \cos(x_1)\sin(x_2))$, for the same number of timesteps that each was advected forward, $N = 250$ (i), 500 (ii), 1,000 (iii), 2,000 (iv), 3,000 (v) and 4,000 (vi), using a Sweby limited trapezoidal face value (a) and HyperC (b). Contours are shown at values of 0.025, 0.5 and 0.975.

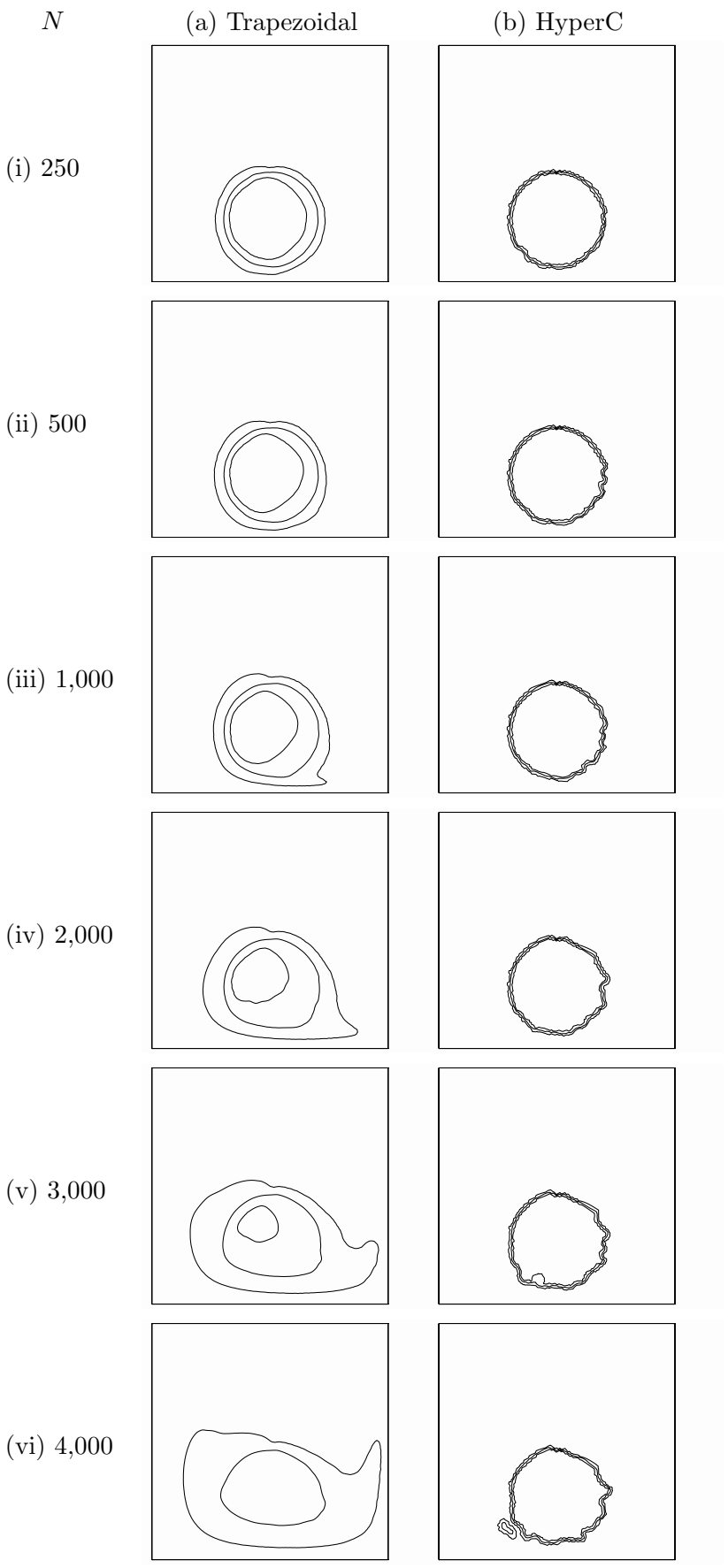


Figure 3.33: *cont.*

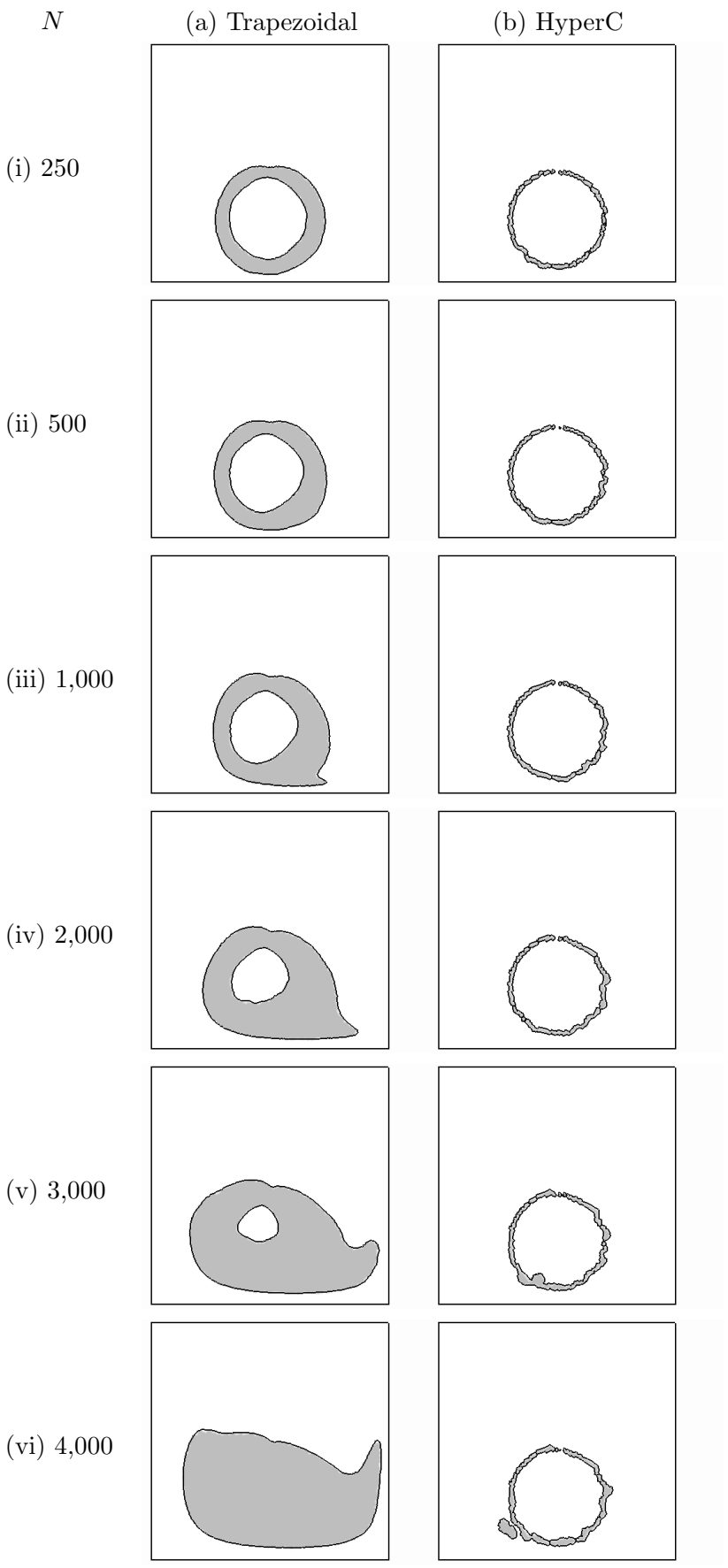


Figure 3.34: *See next page.*

Figure 3.34: *cont.* The absolute difference, $|\alpha - \alpha^e|$, between the exact solution (see Figure 3.29(a)) and the numerical solutions using a Sweby limited trapezoidal face value method (a, see Figure 3.33(a)) and HyperC (b, see Figure 3.33(b)). Values above the 0.025 contour are shaded grey. The results are shown after $N = 250$ (i), 500 (ii), 1,000 (iii), 2,000 (iv), 3,000 (v) and 4,000 (vi) timesteps of forward and reverse advection.

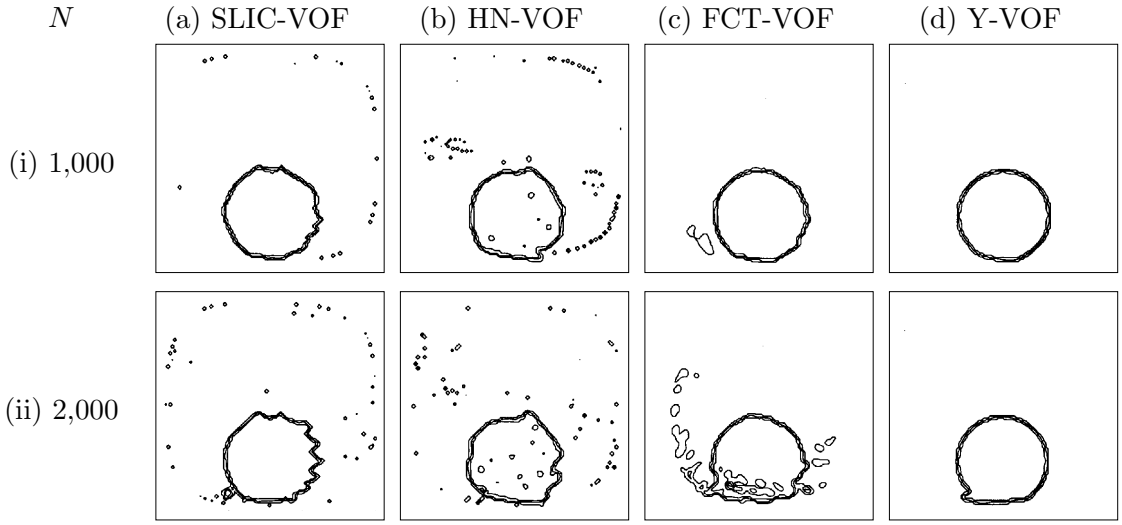


Figure 3.35: Results for the reverse advection in a shearing flow, $\tilde{u} = (-\sin(x_1)\cos(x_2), \cos(x_1)\sin(x_2))$, after $N = 1{,}000$ (i) and 2,000 (ii) timesteps using SLIC-VOF (a), HN-VOF (b), FCT-VOF (c) and Y-VOF (d). All images are taken from Rudman (1997) and show contours at values of 0.025, 0.5 and 0.975.

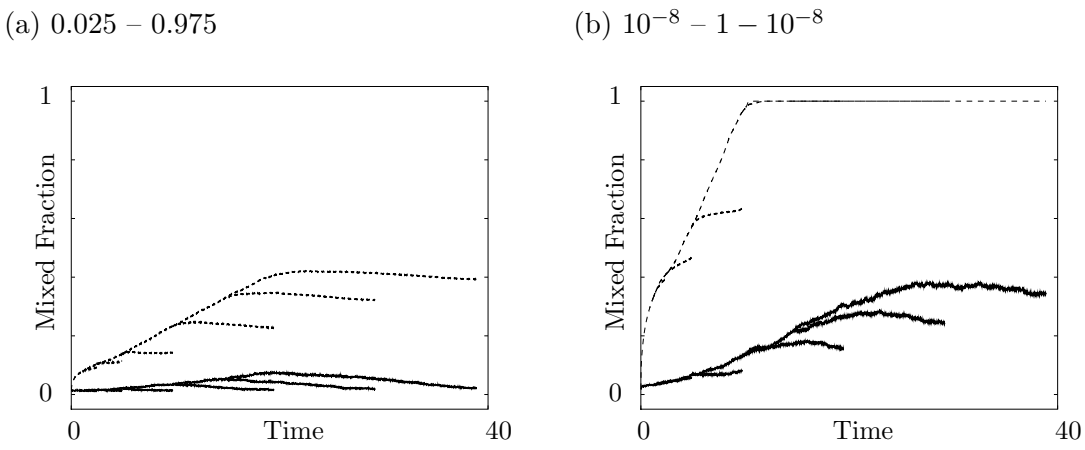


Figure 3.36: Fractional area of the domain which consists of mixed cells over time for both the forward and reverse runs of the shearing flow test. Two ranges of mixing are shown: 0.025 – 0.975 (a) and 10^{-8} – $1 - 10^{-8}$ (b) for a Sweby limited trapezoidal face value (dashed) and HyperC (solid).

3.4 Adjusting the Boundedness Criteria: A New Compressive Advection Method

In Section 3.3.2 the triple role of the upwind value in HyperC was discussed. It is used to estimate the upwind flux, detect extrema and as a target defining boundedness when calculating the optimum face value. The use of the upwind value from the neighbouring cell (in one dimension) or an estimate of the value in the upwind direction (in multiple dimensions) gives HyperC a total variation diminishing (TVD) property.

TVD schemes aim to ensure that an advected field stays bounded in the sense that $\min(\alpha^n) \leq \alpha^{n+1} \leq \max(\alpha^n)$, where n is the timestep level. In the case of HyperC this means that a donor cell in a forward facing step is not allowed to exceed (or drop below in a backward facing step) the value upwind of it. The result of this bounding restriction can be seen in Figure 3.30. Once the tail of the sheared circle is too fine to be represented on the available mesh resolution the volume fraction drops below the original maximum bound of 1. TVD ensures that it will always remain below it during forward advection.

In certain circumstances an alternative definition of boundedness may be desired. For instance when advecting material volume fractions the physical bounds on the field are known *a priori* to be $0 \leq \alpha \leq 1$ and cells that fall in between these values are non-physical mixtures of materials. Therefore it may be beneficial to define boundedness based on the initial bounds of the field, $\min(\alpha_0) \leq \alpha^{n+1} \leq \max(\alpha_0)$, rather than its current state. Such a scheme satisfies a total variation bounded (TVB) criterion.

Modifying HyperC in order to satisfy a TVB rather than a TVD criterion clearly requires the modification of the upwind value. In Equation 3.33 the upwind value was introduced as a target value, thereby defining the boundedness criterion. This can be easily modified to use a target upwind value, α_{tu_k}:

$$\alpha^n_{c_k} + c_{c_{f_{u_k}}} \alpha^n_{u_k} - c_{c_{f_k}} \alpha^n_{d_k} \leq \alpha_{tu_k}, \qquad \text{if } \alpha^n_{u_k} > \alpha^n_{d_k},\ \alpha_{tu_k} = \max(\alpha_0), \tag{3.66a}$$

$$\alpha^n_{c_k} + c_{c_{f_{u_k}}} \alpha^n_{u_k} - c_{c_{f_k}} \alpha^n_{d_k} \geq \alpha_{tu_k}, \qquad \text{if } \alpha^n_{d_k} > \alpha^n_{u_k},\ \alpha_{tu_k} = \min(\alpha_0). \tag{3.66b}$$

This simple modification enables downwinding to continue until the donor cell reaches the target value rather than the actual upwind value. In the case of an isolated top hat function with a maximum value less than its initial bounds the front will not stop downwinding until the cell behind it has exceeded its current bounds and attempted to regain its original maximum.

The end result of using Equation 3.66 to define the downwinding region is a series of local spikes rather than a contiguous region of material. This is demonstrated in Figure 3.37, where an isolated top hat function has an initial maximum value of 1/2. The downwinding region of HyperC has been modified to incorporate a maximum target bound of one at the leading step (when the slope, $\alpha^n_{u_k} - \alpha^n_{d_k}$, is negative) and a minimum target bound of zero at the backward facing step (when the slope, $\alpha^n_{u_k} - \alpha^n_{d_k}$, is positive). During advection (Figure 3.37(b–h)) this results in the forward facing step exceeding its initial bound of 1/2, though it is still constrained to be less that 1. However, rather than transforming the top hat function into a field bounded strictly between 0 and 1, it leads

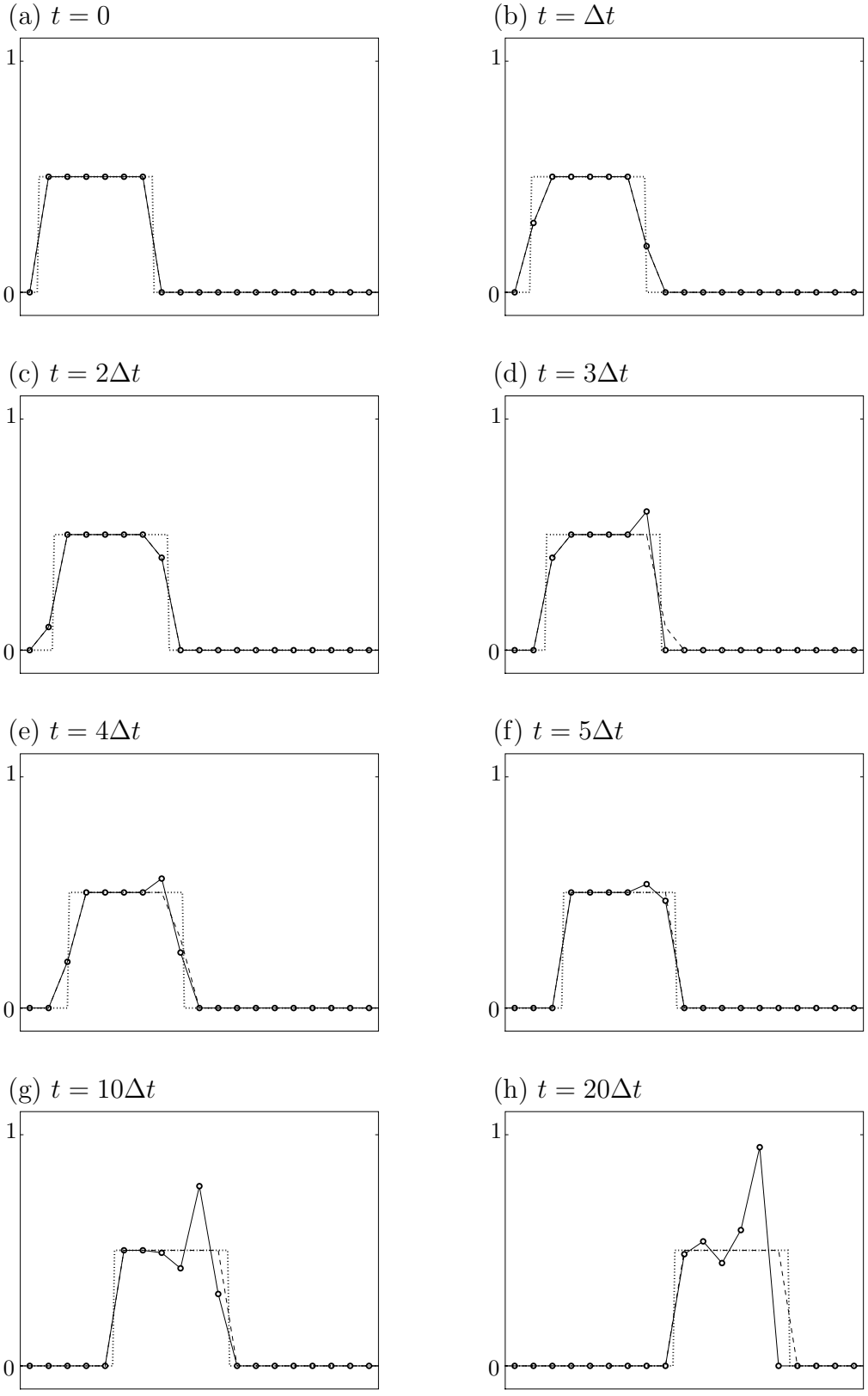


Figure 3.37: *See next page.*

Figure 3.37: *cont.* Eight timesteps from the advection of a step function using the downwinding limit described in Equations 3.66 and Equation 3.24 at a constant Courant number of 2/5 with $\tilde{u} > 0$. For comparison the solution using HyperC (dashed line) and the exact solution (dotted line) are also shown.

to a series a oscillations in the solution. This is a basic continuity problem. The cell at the leading edge is waiting for sufficient material to regain its original bounds but no extra material is being transported into it to achieve this. Furthermore, once it exceeds the current maximum value it will be a local extremum and subjected to upwinding, which causes it to start advecting again.

It therefore becomes obvious that, in order to maintain a contiguous region of material, the flux rate behind a front must be increased and the definition of an extremum must be changed. These are the two other functions of the upwind value. The downwind value also controls the flux rate from cells that are downwinding as well as being a part of the sensor variable detecting extrema. Therefore, as well as a new target upwind value:

$$\alpha_{tu_k} = \max(\alpha_0) \quad \text{if } \alpha^n_{u_k} > \alpha^n_{d_k}, \tag{3.67a}$$

$$\alpha_{tu_k} = \min(\alpha_0) \quad \text{if } \alpha^n_{d_k} > \alpha^n_{u_k}, \tag{3.67b}$$

a new target downwind value, α_{td_k}, is also introduced such that:

$$\alpha_{td_k} = \min(\alpha_0) \quad \text{if } \alpha^n_{u_k} > \alpha^n_{d_k}, \tag{3.68a}$$

$$\alpha_{td_k} = \max(\alpha_0) \quad \text{if } \alpha^n_{d_k} > \alpha^n_{u_k}. \tag{3.68b}$$

The relative values between the target upwind, donor and target downwind nodes can be used to determine if a solution around a particular face is locally smooth:

$$\alpha_{tu_k} > \alpha_{c_k} > \alpha_{td_k}, \tag{3.69a}$$

$$\alpha_{td_k} > \alpha_{c_k} > \alpha_{tu_k}, \tag{3.69b}$$

or contains a local extremum:

$$\alpha_{c_k} \geq \alpha_{tu_k} \geq \alpha_{td_k}, \tag{3.70a}$$

$$\alpha_{c_k} \leq \alpha_{tu_k} \leq \alpha_{td_k}, \tag{3.70b}$$

$$\alpha_{c_k} \geq \alpha_{td_k} \geq \alpha_{tu_k}, \tag{3.71a}$$

$$\alpha_{c_k} \leq \alpha_{td_k} \leq \alpha_{tu_k}. \tag{3.71b}$$

Using normalised variables, Equation 3.69 can be rewritten as:

$$\bar{\alpha}_{tu_k} < \bar{\alpha}_{c_k} < \bar{\alpha}_{td_k} \tag{3.72}$$

while Equations 3.70 and 3.71 become:

$$\bar{\alpha}_{c_k} \geq \bar{\alpha}_{td_k} \quad \text{and} \tag{3.73}$$

$$\bar{\alpha}_{c_k} \leq \bar{\alpha}_{tu_k}. \tag{3.74}$$

Thus the discrete solution can be considered smooth when the normalised donor value lies between the normalised target upwind and normalised target downwind values.

These criteria can also be represented by defining a modified normalised donor variable:

$$\check{\alpha}_{c_k} = \frac{\alpha_{c_k} - \alpha_{tu_k}}{\alpha_{td_k} - \alpha_{tu_k}}. \tag{3.75}$$

Then Equation 3.69 can be rewritten as:

$$0 < \check{\alpha}_{c_k} < 1, \tag{3.76}$$

while Equations 3.70 and 3.71 become:

$$\check{\alpha}_{c_k} \geq 1 \quad \text{and} \tag{3.77}$$

$$\check{\alpha}_{c_k} \leq 0. \tag{3.78}$$

Using this modified normalised donor variable the smoothness criterion looks almost identical to those used in standard normalised variable diagrams (see Figure 3.3(a)).

The modification of the smoothness criterion means that values are now considered smooth that lie within the original bounds of the solution rather than the current bounds. This means that higher-order fluxes can be attempted at a larger range of values. For instance, in the case of a volume fraction using a HyperC-like scheme downwinding can be attempted in more regions. However, the downwinding must use an appropriately decreased (for a forward facing step or increased for a backward facing step) flux in order to maintain a contiguous field. Therefore, within the smooth region, downwinding using the target downwind value is only attempted when it is safe to do so:

$$\alpha^n_{c_k} + c_{c_{f_{u_k}}} \alpha_{tu_k} - c_{c_{f_k}} \alpha_{td_k} \leq \alpha_{tu_k}, \qquad \text{if } \alpha^n_{u_k} > \alpha^n_{d_k},\ \alpha_{tu_k} = \max(\alpha_0),\ \alpha_{td_k} = \min(\alpha_0), \tag{3.79a}$$

$$\alpha^n_{c_k} + c_{c_{f_{u_k}}} \alpha_{tu_k} - c_{c_{f_k}} \alpha_{td_k} \geq \alpha_{tu_k}, \qquad \text{if } \alpha^n_{d_k} > \alpha^n_{u_k},\ \alpha_{tu_k} = \min(\alpha_0),\ \alpha_{td_k} = \max(\alpha_0). \tag{3.79b}$$

Here the definition of safe assumes that the upwind face will be propagating the maximum (or minimum) target upwind value possible. In combination with the smoothness criterion (Equation 3.72) this can be converted to normalised variables giving:

$$\left(1 - c_{c_{f_k}}\right) \bar{\alpha}_{tu_k} + c_{c_{f_k}} \bar{\alpha}_{td_k} \leq \bar{\alpha}^n_{c_k} < \bar{\alpha}_{td_k} \tag{3.80}$$

or, using the modified normalised variable:

$$c_{c_{f_k}} \leq \check{\alpha}^n_{c_k} < 1. \tag{3.81}$$

Within these regions targeted downwinding:

$$\hat{\alpha}^n|^{HO}_{f_k} = \alpha_{td_k}, \tag{3.82}$$

or, using normalised variables,

$$\bar{\hat{\alpha}}^n|_{f_k}^{HO} = \bar{\alpha}_{td_k} \quad \text{or} \tag{3.83}$$

$$\check{\hat{\alpha}}^n|_{f_k}^{HO} = 1 \tag{3.84}$$

is safe.

In the remainder of the smooth zone ($\bar{\alpha}_{tu_k} < \bar{\alpha}_{c_k}^n < \left(1 - c_{c_{f_k}}\right)\bar{\alpha}_{tu_k} + c_{c_{f_k}}\bar{\alpha}_{td_k}$ or $0 < \check{\alpha}_{c_k}^n < c_{c_{f_k}}$) the downwind flux must be limited to ensure the total variation bounded property. Therefore the face value must be selected so that the target upwind value is attained by the donor cell at the next timestep, $\alpha_{c_k}^{n+1} = \alpha_{tu_k}$:

$$\hat{\alpha}^n|_{f_k}^{HO} = \frac{1}{c_{c_{f_k}}}\left(\alpha_{c_k}^n + \alpha_{tu_k}\right) + \alpha_{tu_k}, \qquad \text{when } 0 < \check{\alpha}_{c_k}^n < c_{c_{f_k}}, \tag{3.85}$$

or, using normalised variables:

$$\bar{\hat{\alpha}}^n|_{f_k}^{HO} = \frac{\bar{\alpha}_{c_k}^n + \left(c_{c_{f_k}} - 1\right)\bar{\alpha}_{tu_k}}{c_{c_{f_k}}}, \quad \text{when } \bar{\alpha}_{tu_k} < \bar{\alpha}_{c_k}^n < \left(1 - c_{c_{f_k}}\right)\bar{\alpha}_{tu_k} + c_{c_{f_k}}\bar{\alpha}_{td_k}, \text{ or} \tag{3.86a}$$

$$\check{\hat{\alpha}}^n|_{f_k}^{HO} = \frac{1}{c_{c_{f_k}}}\check{\alpha}_{c_k}^n, \qquad \text{when } 0 < \check{\alpha}_{c_k}^n < c_{c_{f_k}}. \tag{3.86b}$$

Outside of the smooth zone the same logic as applied before implies that only upwinding ($\hat{\alpha}^n|_{f_k}^{HO} = \alpha_{c_k}^n$) will smooth out extrema and return the donor value to within the target bounds. This, in combination with the target downwind flux and the Courant limited face value, makes up a new TVB scheme, referred to here as UltraC. The behaviour of the face value, $\hat{\alpha}^n|_{f_k}^{HO}$, using UltraC is represented in Figure 3.38(a) on a normalised variable diagram. For comparison the HyperC scheme is also shown.

Clearly UltraC extends the definition of boundedness beyond the area used by HyperC, representing the upper boundary of the TVB rather than the TVD criterion. However, the bounds on UltraC are variable and as the actual downwind, $\alpha_{d_k}^n$, and upwind, $\alpha_{u_k}^n$, values tend towards the target downwind, $\alpha_{td_k}^n$, and target upwind, $\alpha_{tu_k}^n$, values HyperC and UltraC become indistinguishable. This means that for a region of the field that is already bounded by the initial maximum and minimum UltraC will behave in exactly the same manner as HyperC. The similarity between the two schemes is further emphasised on a modified normalised variable diagram (Figure 3.38(b)) where UltraC looks virtually identical to HyperC on a standard normalised variable diagram (Figure 3.8(a)). This similarity makes UltraC particularly easy to implement by modifying the HyperC code.

The major difference between the two schemes arises when the field values fall between the original bounds. Figure 3.39 shows eight timesteps from the advection of a top hat function using both UltraC and HyperC. At the start of advection the top hat lies in the bounds $[0, 0.5]$ but is considered to have originally been bounded by $[0, 1]$. This is analogous to an isolated piece of material that has become mixed with the surrounding material through numerical diffusion. HyperC aims to maintain the bounds from the start of the simulation while UltraC attempts to return the function to the original bounds.

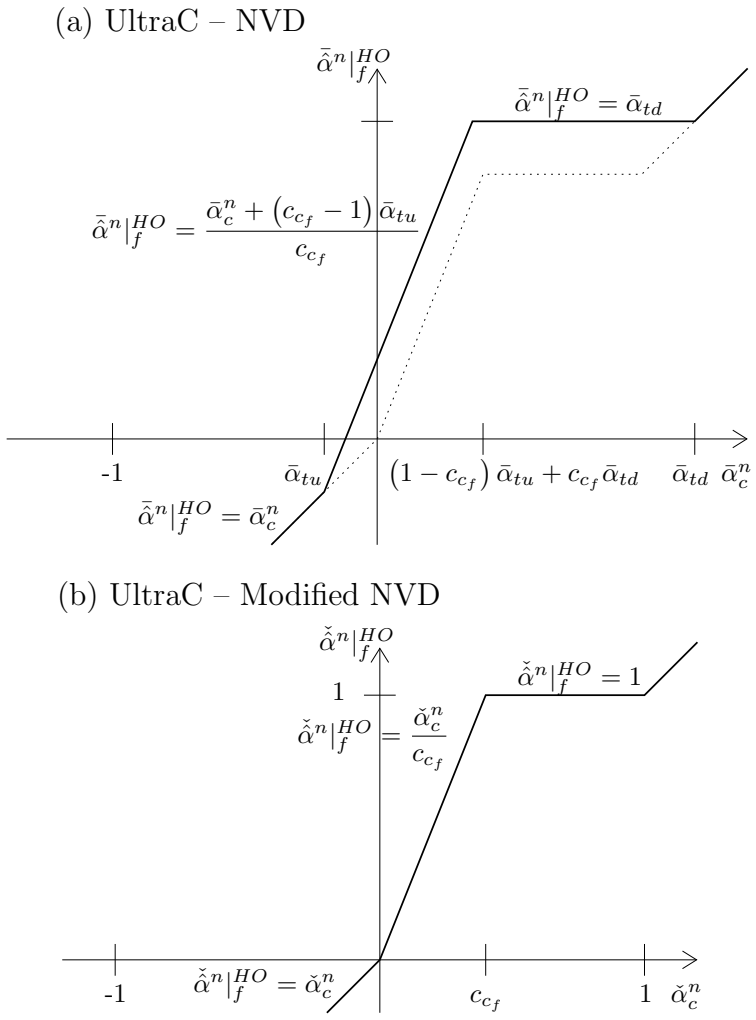


Figure 3.38: The UltraC face value function (solid line) represented on (a) a normalised variable diagram, (b) a modified normalised variable diagram. The HyperC scheme is also shown in (a) as a dotted line.

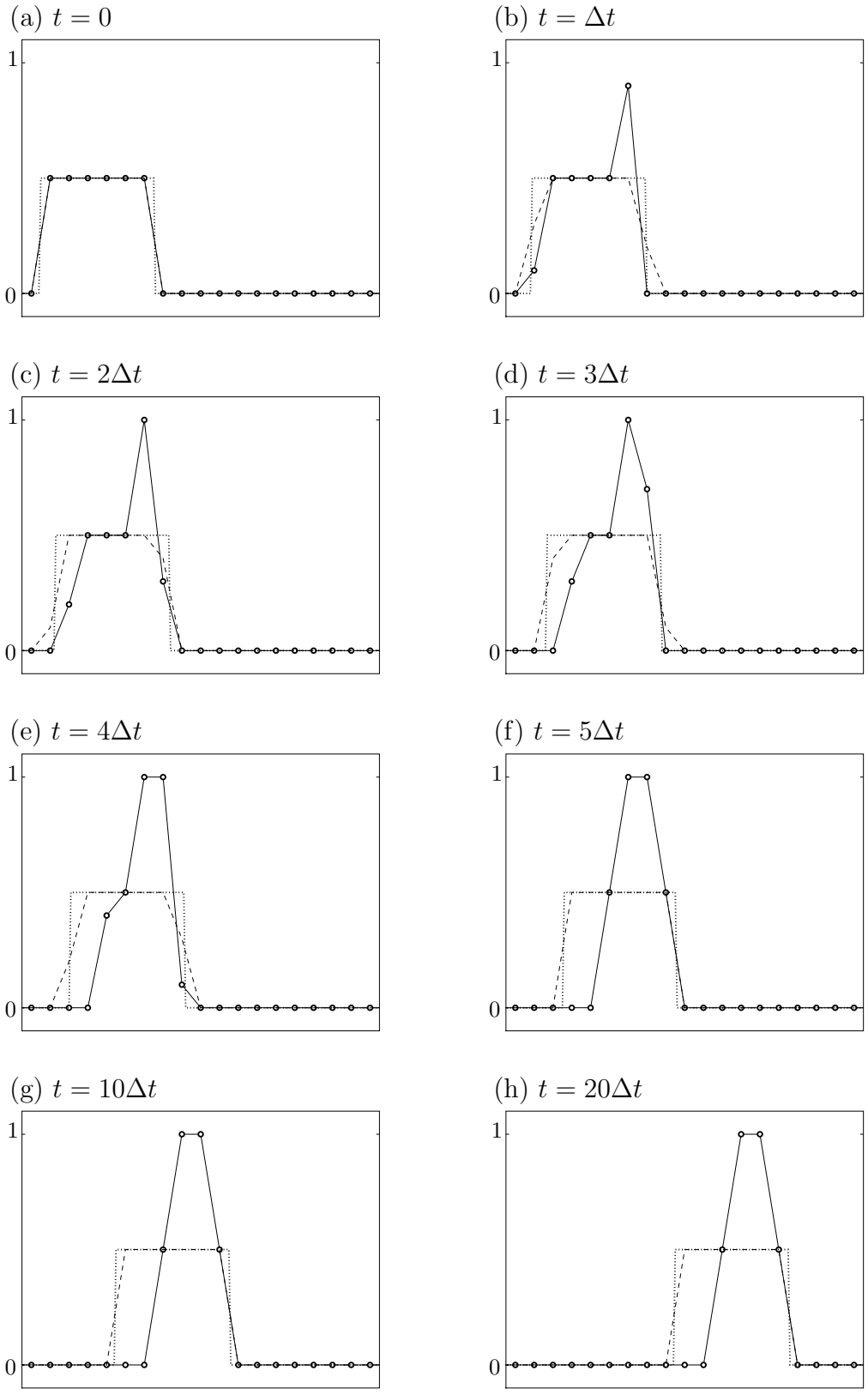


Figure 3.39: *See next page.*

Figure 3.39: *cont.* Eight timesteps from the advection of a step function using Equation 3.24 at a constant Courant number of 2/5 with $\tilde{u} > 0$ using UltraC (hollow circles, solid line). The field begins the simulation with a maximum value of 1/2 but is considered to be bounded up to 1. For comparison the solution using HyperC (dashed line) and the exact solution (dotted line) are also shown.

At the leading step of the solution field, UltraC considers targeted downwinding safe until the value behind the step reaches the target upwind value (1). This means that a spike develops (Figure 3.39(b)). Due to the extended definition of smoothness however, UltraC still considers this to be a smooth region and downwinding continues. HyperC in contrast is attempting to maintain the initial bounds ($[0, 0.5]$) and therefore prevents the development of such a spike by upwinding the front straight away. Similarly, in the main flat body of the function, HyperC uses upwinding while UltraC considers it to be safe to use targeted downwinding. This has the effect of increasing the flux rate and speeding up the advection of the back end of the function. This is necessary to maintain the continuity of the field.

Once the targeted maximum value is about to be reached at the leading step, UltraC switches to a limited target downwind flux and allows the front to continue (Figure 3.39(c)). The main body of the top hat is still advecting at an increased rate using targeted downwinding to catch up with the front. Targeted downwinding and limited downwinding then continue at the new higher step much like in HyperC (Figure 3.39(d–e)) allowing it to move forward. Eventually a top hat function is attained which is bounded by $[0, 1]$ rather than $[0, 0.5]$ (Figure 3.39(f)). This continues to advect exactly as if it were using HyperC (Figure 3.39(g–h)).

Clearly UltraC has a dramatic effect on the shapes of functions it advects, modifying the fluxes to restore the field to within the original bounds. While HyperC respected the initial bounds of the field, Figure 3.11 showed that it could still have a dramatic effect on the shape of some functions. Effectively a smooth function was turned into a staircase of small step functions, which HyperC could then preserve the shape of individually. Figure 3.40 shows six timesteps from the advection of the same smooth function as in Figure 3.11, this time using UltraC rather than HyperC. Instead of turning the smoothly varying function into a series of small step functions UltraC has modified it into a single step function advecting at the inflection point of the original smooth function. Initially, any point less than half way between the targeted bounds of $[0, 1]$ is set to use targeted downwinding. This prevents the lower curve from moving. The upper points are also considered smooth but due to the use of the target upwind value as an estimate of the incoming flux they end up advecting forward using a limited downwind flux (Figure 3.40(b)). As all points use the same targeted values to set this flux the upper curve moves forward contiguously.

This continues until the same estimate of the flux leads the midpoint to believe it is now unsafe to use targeted downwinding. A limited flux is then allowed and the

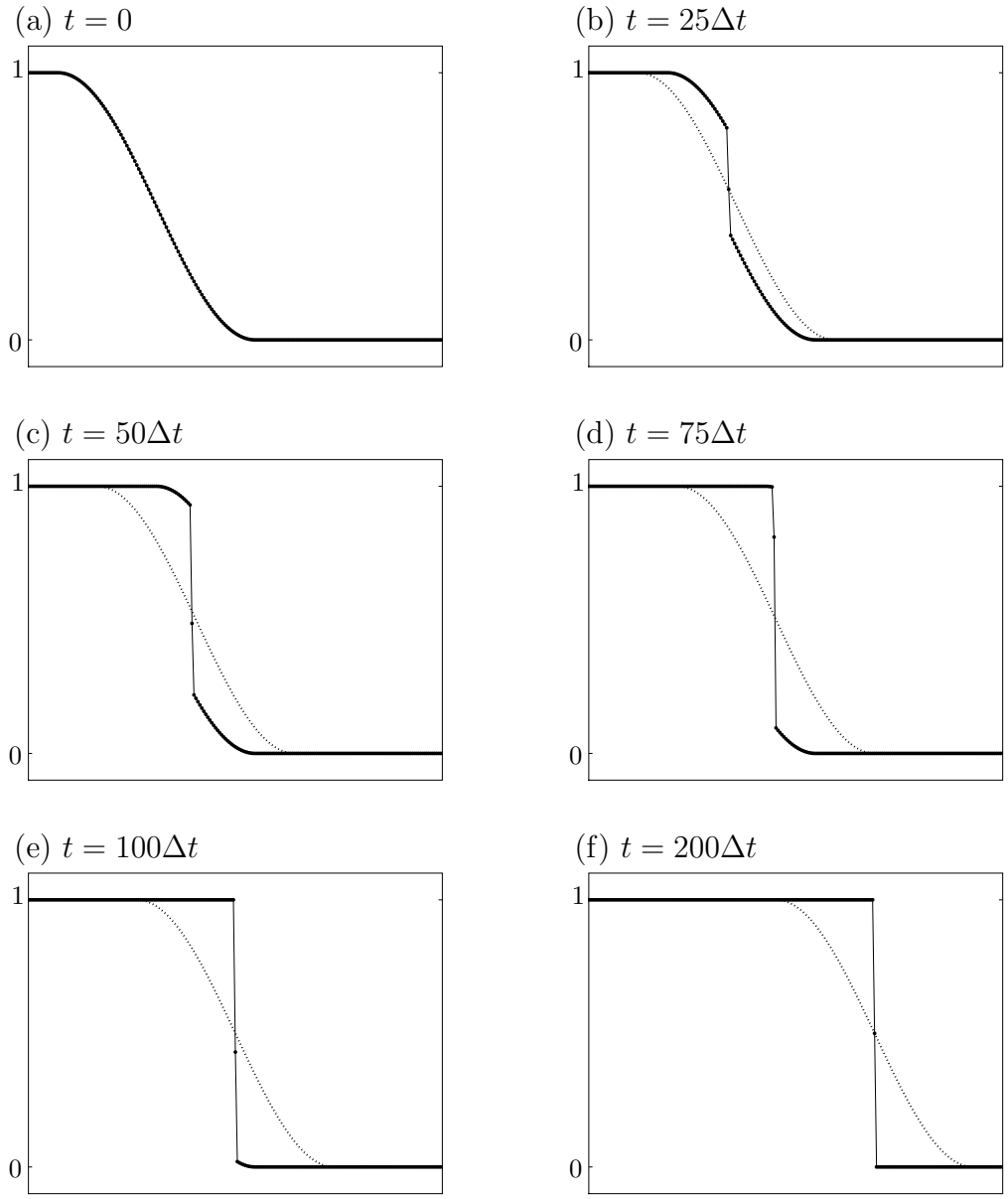


Figure 3.40: Six timesteps from the advection of a quarter period of a cosine function using Equation 3.24 at a constant Courant number of 2/5 with $\tilde{u} > 0$. The solution using UltraC (circles, solid line) becomes distorted during advection. For comparison the exact solution is shown (dotted line).

developing step function moves forward (Figure 3.40(c)). The curve behind the step is consumed by the increased flux of material while the curve in front of the step is still prevented from moving, until the step function reaches it, by the diminished flux of material (Figure 3.40(d–e)). This continues until a step function from zero to one is optimally represented on the resolution available (Figure 3.40(f)). Advection then proceeds as it would with HyperC.

For smoothly varying functions that are meant to be advected as such, UltraC, just like HyperC, clearly produces non-physical behaviour. However, if the field is meant to be minimally dissipative and strictly bounded the sharpening caused by UltraC may be considered acceptable. For instance, it may restore a volume fraction, which has been smeared by low resolution or an interpolation between two adapted meshes, to physical values. It may be thought of as acting like a transient surface tension around diffuse interfaces. It should be stressed that this is not a physical surface tension. In particular it has no effect unless the interface is being advected (i.e. non-zero advective velocity) whereas a real surface tension initiates motion at any interface not in equilibrium.

As has been seen in the last two examples the transient surface tension behaviour of UltraC depends on increasing and decreasing the fluxes in order to cause low valued regions to coalesce. This apparently non-physical behaviour can be justified numerically for some fields but may also produce incorrect behaviour. Take for example Figure 3.41(a), which shows an isolated top hat function within the bounds $[0, 0.5]$. As in Figure 3.39, this top hat is considered to have been originally bounded by $[0, 1]$, however the function is now only represented on a single grid point.

As the top hat in Figure 3.41 is below the target maximum bound, UltraC (hollow circles, solid line in Figure 3.41(b)) increases the fluxes behind the front in an attempt to increase its value. However, because it is only one grid point wide, there is insufficient material to change the bounds so increasing the fluxes only causes UltraC to advect the function at an artificially increased velocity (Figure 3.41(c–h)).

As UltraC also decreases fluxes in order to create sharp interfaces the opposite behaviour is equally possible. In other words an isolated top hat may also be advected at an artificially decreased velocity rather than at an increased velocity. This behaviour can be seen in Figure 3.42 (hollow circles, solid line) where the same top hat function as in Figure 3.41 is stationary when advected using UltraC.

The change in behaviour between Figures 3.41 and 3.42 is caused by defining the target upwind, α_{tu_k}, and target downwind, α_{td_k}, values differently for the case when

Figure 3.41: *See next page.* Eight timesteps from the advection of an isolated top hat function using Equation 3.24 at a constant Courant number of 2/5 with $\tilde{u} > 0$. The results using UltraC (hollow circles, solid line), UltraC-P (hollow squares, solid line), UltraC-S (hollow triangles) and HyperC (dashed line) are shown in comparison to the exact solution (dotted line).

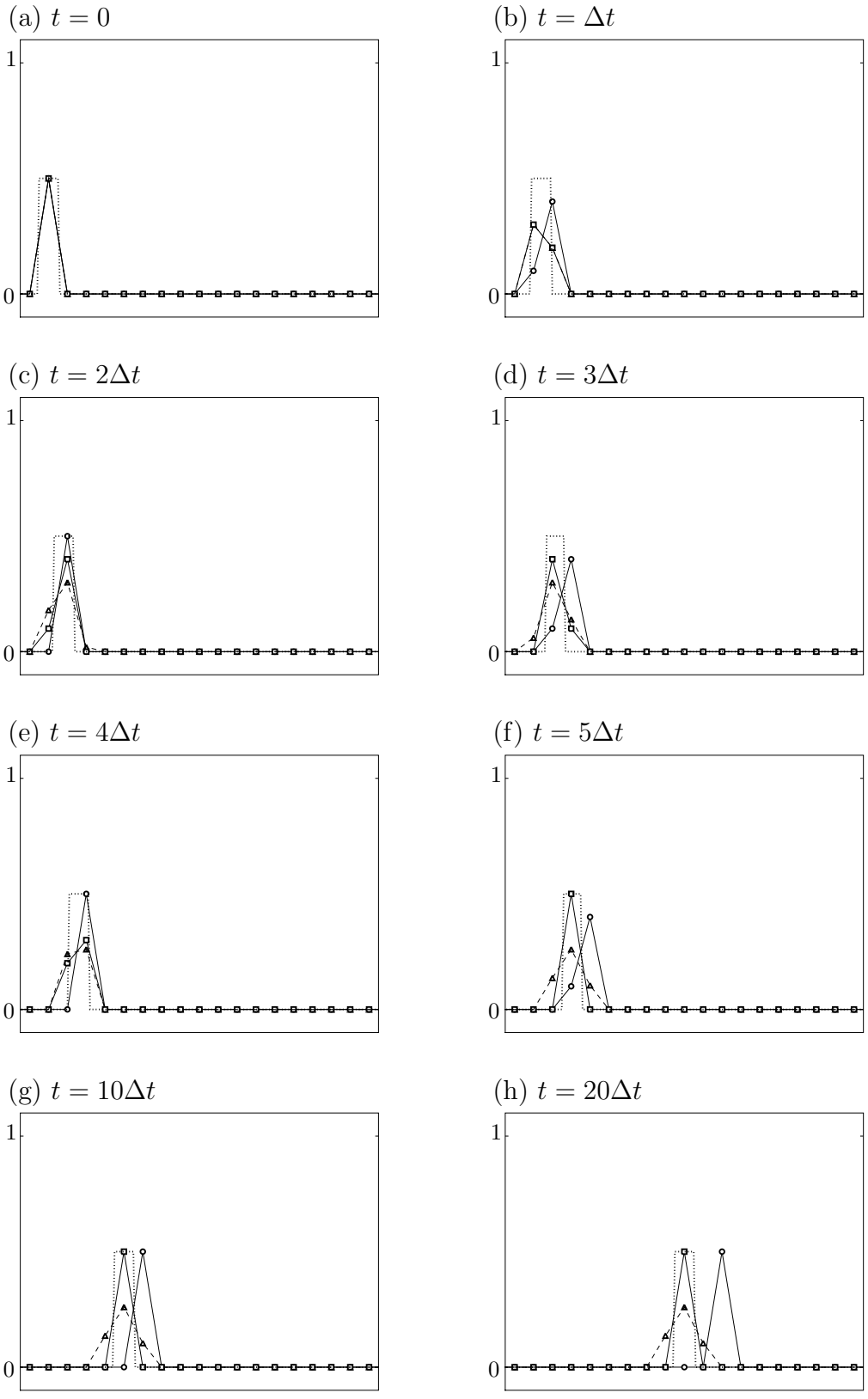


Figure 3.41: *cont.*

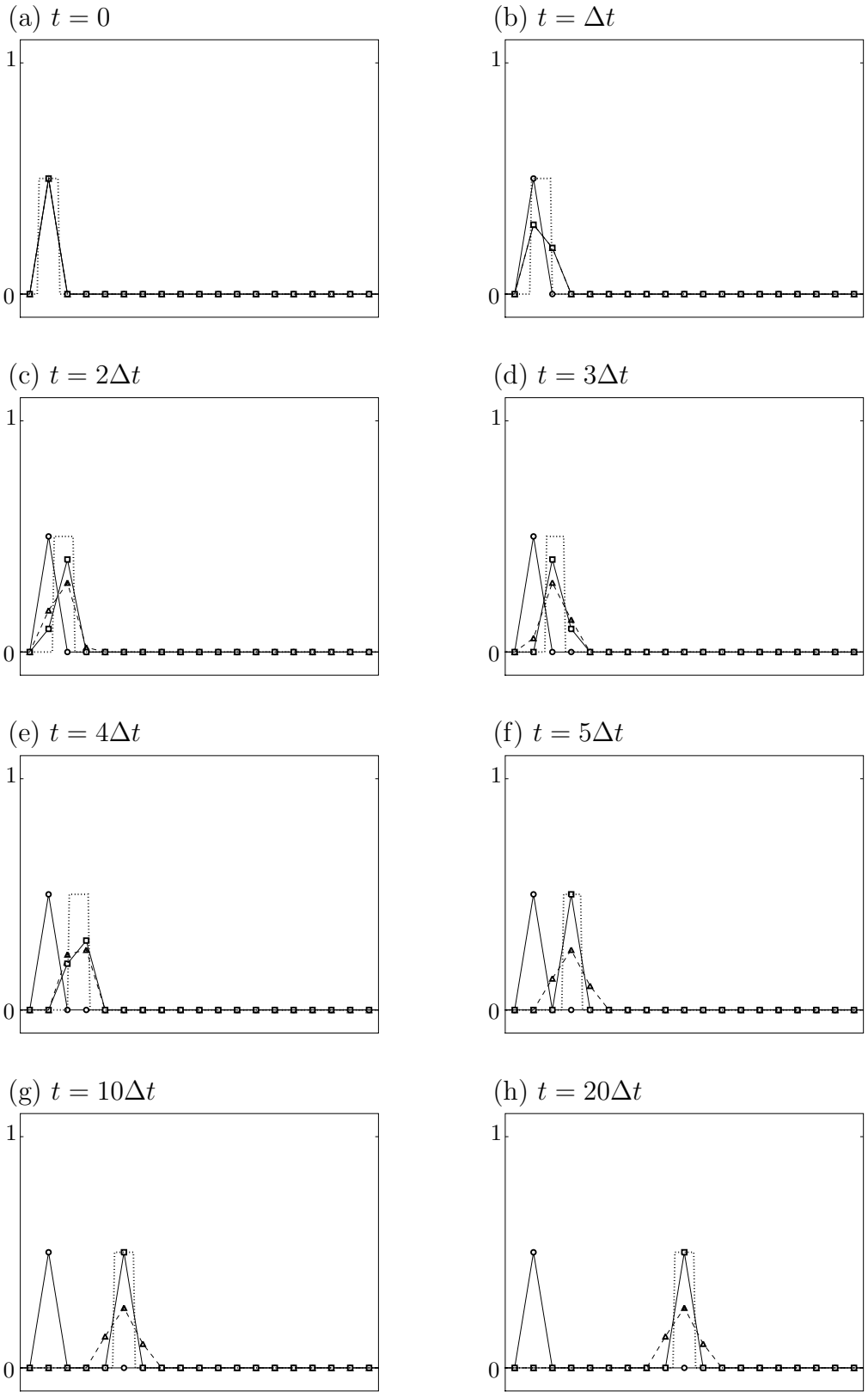


Figure 3.42: *See next page.*

Figure 3.42: *cont.* Eight timesteps from the advection of an isolated top hat function using Equation 3.24 at a constant Courant number of 2/5 with $\tilde{u} > 0$. The results using UltraC (hollow circles, solid line), UltraC-P (hollow squares, solid line), UltraC-S (hollow triangles) and HyperC (dashed line) are shown in comparison to the exact solution (dotted line).

$\alpha^n_{d_k} = \alpha^n_{u_k}$. In Figure 3.41:

$$\alpha_{tu_k} = \min(\alpha_0) \quad \text{if } \alpha^n_{u_k} = \alpha^n_{d_k}, \text{ and} \tag{3.87a}$$

$$\alpha_{td_k} = \max(\alpha_0) \quad \text{if } \alpha^n_{d_k} = \alpha^n_{u_k}. \tag{3.87b}$$

Therefore targeted downwinding at the front of the top hat causes it to advect at an increased rate. Similarly in Figure 3.42:

$$\alpha_{tu_k} = \max(\alpha_0) \quad \text{if } \alpha^n_{u_k} = \alpha^n_{d_k}, \text{ and} \tag{3.88a}$$

$$\alpha_{td_k} = \min(\alpha_0) \quad \text{if } \alpha^n_{d_k} = \alpha^n_{u_k}. \tag{3.88b}$$

Hence targeted downwinding at the front of the top hat causes it to remain stationary, awaiting an increase in the bounds which will not happen.

It is possible to fix this erroneous behaviour by introducing a function which attempts to detect if there will be sufficient material to satisfy the new bounds. In the event that there isn't, UltraC can be modified to prevent incorrect advection velocities. In its simplest form, ignoring all local area or volume variations, the potential function takes the form:

$$\alpha_{p_k} = \alpha^n_{u_k} + \alpha^n_{c_k} + \alpha^n_{d_k}. \tag{3.89}$$

If the potential value is greater than the maximum target bound, $\alpha_{p_k} > \max(\alpha_0)$, then UltraC can proceed as normal. However if the potential value is lower than this then UltraC may be unsafe and should be modified. Two alternative modifications are presented.

The target values may be limited by the potential value:

$$\left.\begin{aligned} \alpha_{tu_k} &= \min(\alpha_{p_k}, \max(\alpha_0)) \\ \alpha_{td_k} &= \min(\alpha_0) \end{aligned}\right\} \text{if } \alpha^n_{u_k} > \alpha^n_{d_k}, \tag{3.90}$$

or:

$$\left.\begin{aligned} \alpha_{tu_k} &= \min(\alpha_0) \\ \alpha_{td_k} &= \min(\alpha_{p_k}, \max(\alpha_0)) \end{aligned}\right\} \text{if } \alpha^n_{d_k} > \alpha^n_{u_k}. \tag{3.91}$$

This scheme is referred to as UltraC-P.

Alternatively the potential value may be used to simply switch UltraC back to the behaviour of HyperC when $\alpha_{p_k} > \max(\alpha_0)$:

$$\left.\begin{aligned} \alpha_{tu_k} &= \alpha^n_{u_k} \\ \alpha_{td_k} &= \alpha^n_{d_k} \end{aligned}\right\} \text{if } \alpha_{p_k} < \max(\alpha_0). \tag{3.92}$$

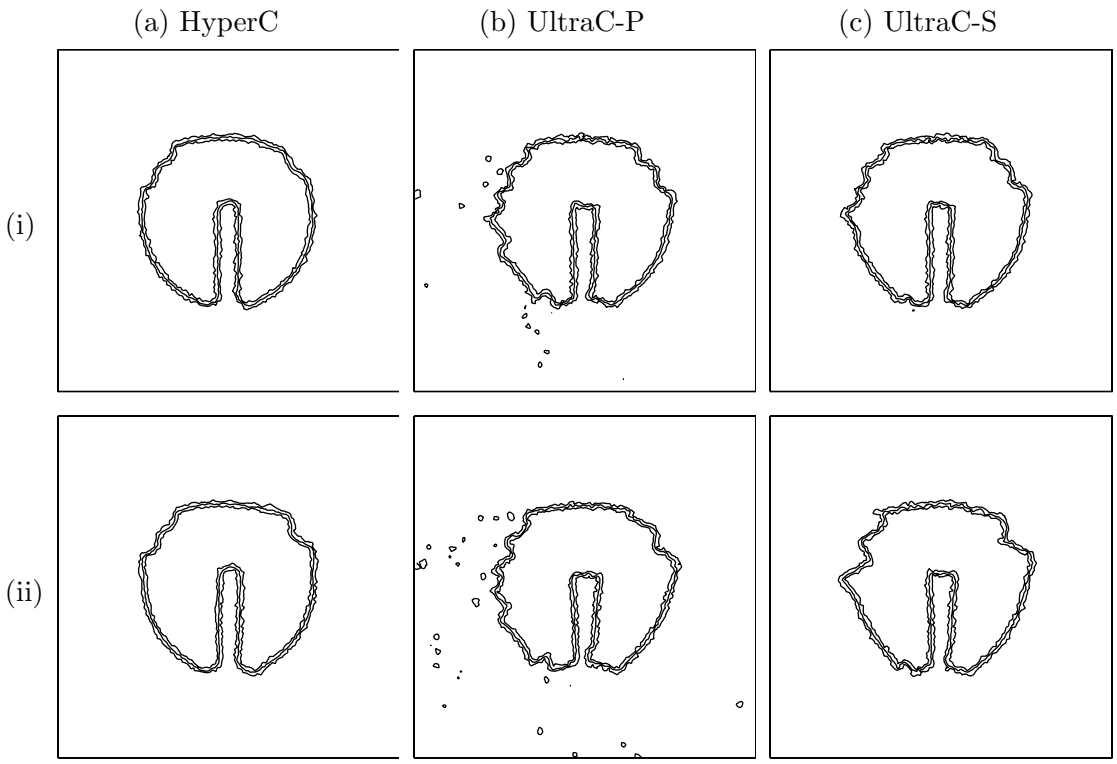


Figure 3.43: Rotation of the slotted circle shown in Figure 3.26(a) on the mesh shown in Figure 3.26(b) using HyperC (a), UltraC-P (b) and UltraC-S (c). Results after one (i) and two (ii) rotations are shown using contours at values of 0.025, 0.5 and 0.975.

This scheme is referred to as UltraC-S.

The results from both UltraC-P (hollow squares, solid line) and UltraC-S (hollow triangles) are also shown in Figures 3.41 and 3.42. In both cases the schemes fix the erroneous advection velocities of UltraC. UltraC-P maintains a sharper representation of the exact solution while, as expected UltraC-S behaves in exactly the same way as HyperC (dashed line) for this problem.

3.4.1 Multi-Dimensional Analysis

UltraC (and its variants) can be extended to the dual control volume mesh in multiple dimensions in exactly the same way as was done for HyperC: using a bounded projected upwind value and an extended definition of the Courant number. It can then be tested in exactly the same manner as HyperC was tested in Section 3.3.2. A sample of these tests is presented below.

Solid Body Rotation

The Zalesak solid body rotation was performed using the same initial condition and mesh as before (Figure 3.26(a, b)). The slotted circle was rotated twice around the origin using UltraC-P and UltraC-S. Comparing these results (Figure 3.43(b, c)) to those produced

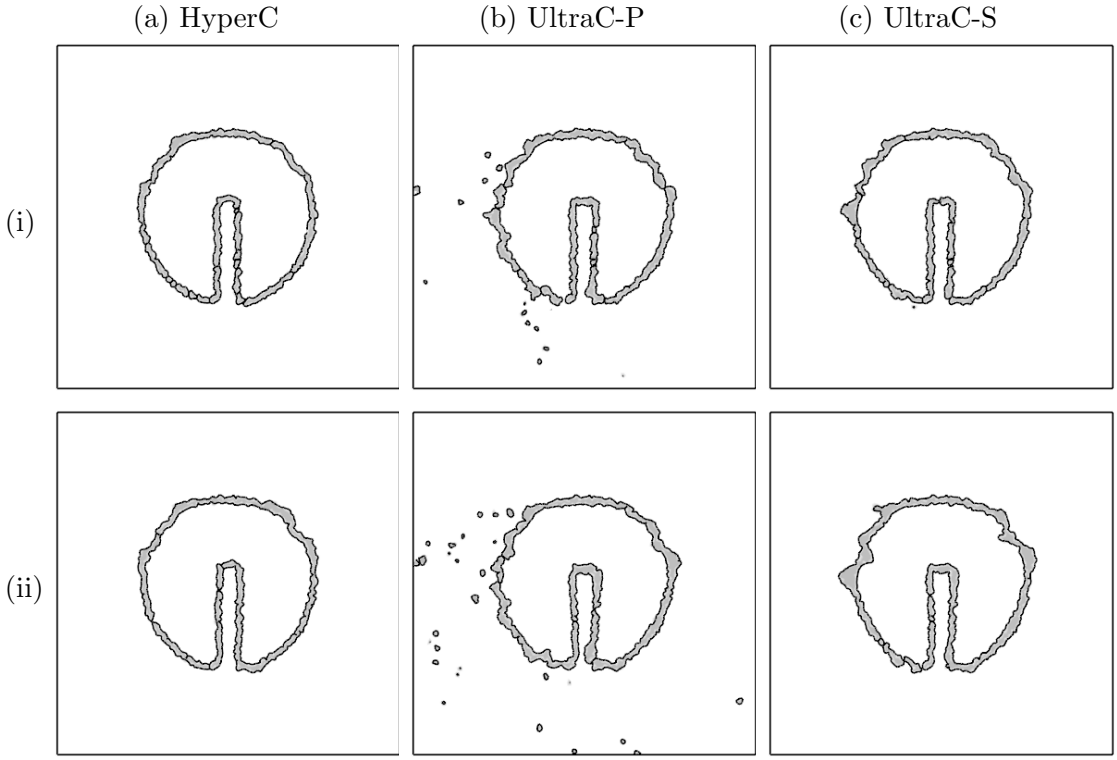


Figure 3.44: The absolute difference, $|\alpha - \alpha^e|$, between the exact solution (see Figure 3.26(a)) and the numerical solutions using HyperC (a, reproduced from Figure 3.27(b)), UltraC-P (b, see Figure 3.43(b) and UltraC-S (c, see Figure 3.43(c)). Values above the 0.025 contour are shaded grey after one (i) and two (ii) rotations.

HyperC		UltraC-P		UltraC-S	
1st Rotation	2nd Rotation	1st Rotation	2nd Rotation	1st Rotation	2nd Rotation
5.21×10^{-2}	5.97×10^{-2}	2.01×10^{-1}	1.04×10^{0}	7.80×10^{-2}	9.90×10^{-2}

Table 3.9: Errors for the Zalesak slot problem after one and two full rotations using HyperC, UltraC-P and UltraC-S. Errors using other face value methods are presented in Table B.1.

using HyperC (Figure 3.43(a)) and to the exact solution (Figure 3.44) it becomes obvious that UltraC has poorer shape preserving properties in multiple dimensions than HyperC. The wrinkling of the outer interface noted previously using HyperC has been accentuated by UltraC.

UltraC-P (Figures 3.43(b), 3.44(b)) has also generated flotsam and jetsam during rotation, in a similar manner to the SLIC-VOF and HN-VOF methods in Figure 3.28(b, c). This is because in Equations 3.90 and 3.91 only the maximum target values were limited by the potential value. The minimum target values may still cause material to be advected at an artificially reduced velocity. Although in a multi-dimensional numerical simulation a situation where $\alpha^n_{u_k} = \alpha^n_{d_k}$ exactly is unlikely, rounding errors can lead to a situation where, when $\alpha^n_{u_k} \approx \alpha^n_{d_k}$, the face value is chosen as if $\alpha^n_{u_k} > \alpha^n_{d_k}$ rather than $\alpha^n_{u_k} < \alpha^n_{d_k}$ or vice versa. This can lead to the flotsam and jetsam seen in Figure 3.43(b).

In this respect UltraC-S outperforms UltraC-P in two dimensions. This can also be seen by calculating the errors according to Equation 3.65 (see Table 3.9). UltraC-S has a lower error than UltraC-P and it does not increase as much between rotations. However, due to the poorer shape preserving qualities of both UltraC methods, they both perform worse than HyperC.

Shearing Flow

In Figure 3.30 HyperC was seen to perform poorly at maintaining the original bounds of the advected field once the field had been sheared to a width equivalent to the mesh resolution. UltraC has been designed to attempt to maintain stricter bounds on the field so the shearing flow tests were repeated using UltraC-P and UltraC-S.

The same initial condition and mesh were used as in Figure 3.29 and the circle was again advected forward for $N = 250$, 500, 1,000, 2,000, 3,000 and 4,000 timesteps at a Courant number of 0.25 before being advected back for an equal number of timesteps. The

Figure 3.45: *See next page.* Results for the forward advection of Figure 3.29(a) in a shearing flow, $\tilde{u} = (\sin(x_1)\cos(x_2), -\cos(x_1)\sin(x_2))$, after $N = 250$ (i), 500 (ii), 1,000 (iii), 2,000 (iv), 3,000 (v) and 4,000 (vi) timesteps using HyperC (a), UltraC-P (b) and UltraC-S (c). Contours are shown at values of 0.025, 0.5 and 0.975.

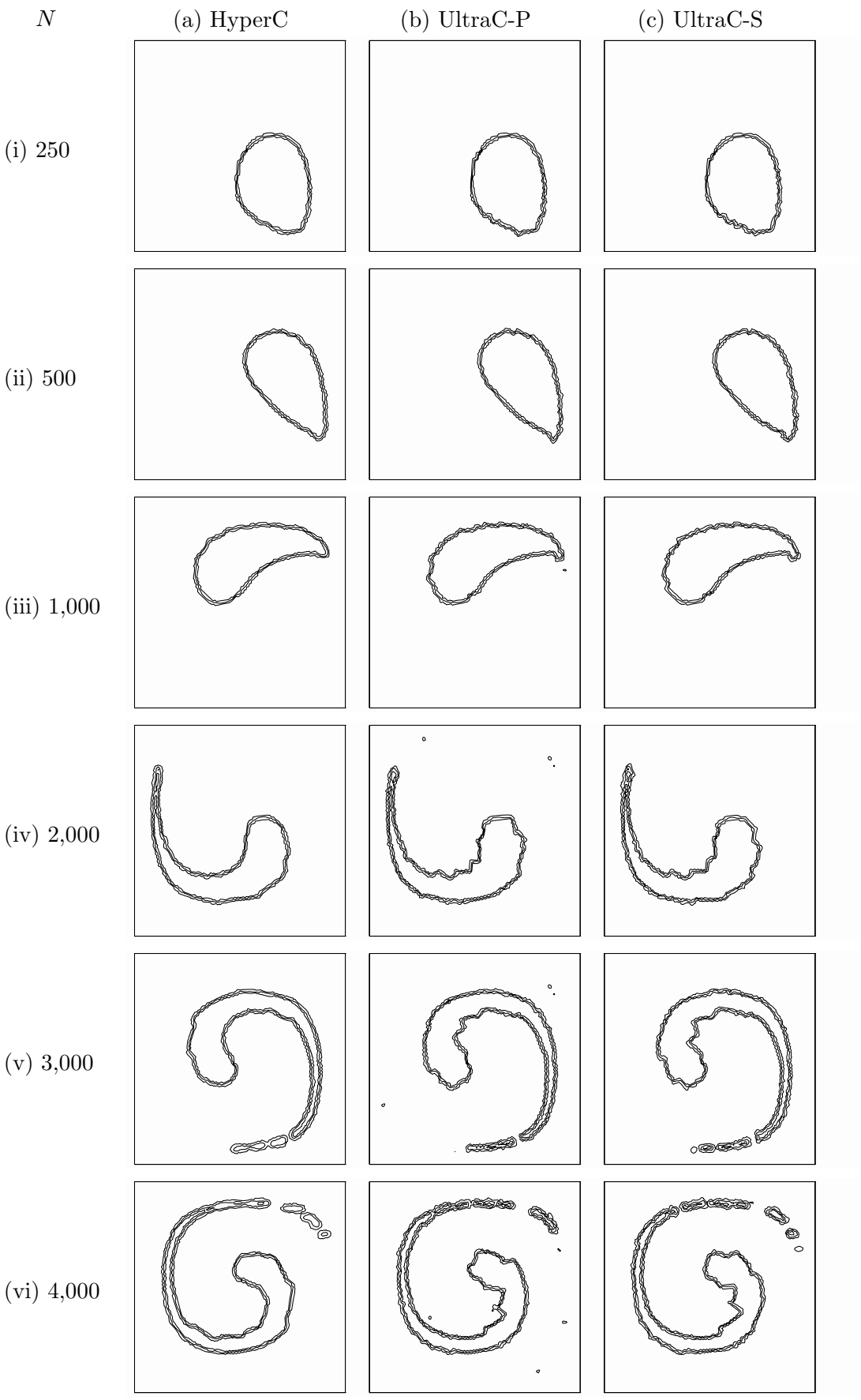


Figure 3.45: *cont.*

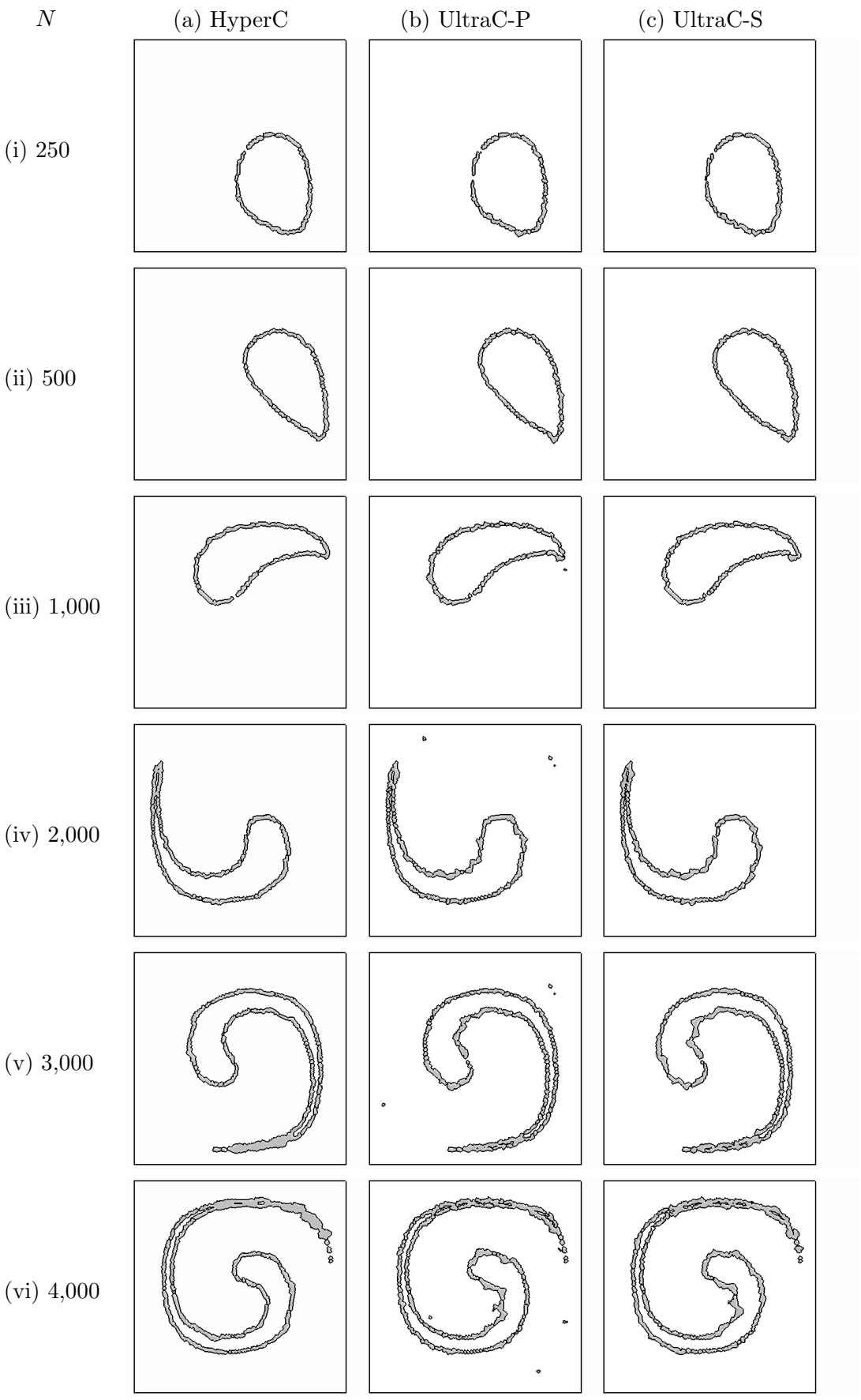


Figure 3.46: *See next page.*

Figure 3.46: *cont.* The absolute difference, $|\alpha - \alpha^e|$, between the 'exact' solution (see Figure 3.30(c)) and the numerical solutions using HyperC (a, reproduced from Figure 3.31(b)), UltraC-P (b, see Figure 3.45(b)) and UltraC-S (c, see Figure 3.45(c)). Values above the 0.025 contour are shaded grey. The results are shown after $N = 250$ (i), 500 (ii), 1,000 (iii), 2,000 (iv), 3,000 (v) and 4,000 (vi) timesteps of forward advection.

N	HyperC	UltraC-P	UltraC-S
250	2.72×10^{-2}	3.33×10^{-2}	3.23×10^{-2}
500	3.00×10^{-2}	3.30×10^{-2}	3.38×10^{-2}
1,000	3.77×10^{-2}	4.62×10^{-2}	4.31×10^{-2}
2,000	6.50×10^{-2}	9.34×10^{-2}	9.40×10^{-2}
3,000	1.16×10^{-1}	1.28×10^{-1}	1.41×10^{-1}
4,000	1.64×10^{-1}	1.78×10^{-1}	1.97×10^{-1}

Table 3.10: Errors between the 'exact' solution (see Figure 3.30(c)) and the numerical solutions using HyperC, UltraC-P and UltraC-S after N timesteps of forward advection.

results from the forward advection using UltraC-P and UltraC-S are presented in Figure 3.45(b, c) and are shown alongside those for HyperC (Figure 3.45(a)) for comparison. These are compared to the 'exact' solution from Figure 3.30(c) in Figure 3.46.

The poorer shape preserving qualities of UltraC are again visible in Figures 3.45–3.46 with increased wrinkling of the interface relative to HyperC. Additionally some flotsam and jetsam can be seen to be generated by UltraC-P. However, after $N = 2,000$ timesteps HyperC begins to fail to be able to represent the tail of the sheared circle on the resolution available. As they use the same mesh, UltraC is also not able to preserve a continuous interface at the tail but the adjusted fluxes ensure that the isolated patches that are shed from the tail are maintained, for the most part, between the bounds $[0, 1]$.

In Figure 3.46 the tighter bounding leads to a thinner zone of difference between the exact and numerical solutions using UltraC in the vicinity of the sheared tail. However, the increased numerical wrinkling of the interface tends to widen this zone elsewhere. Using Equation 3.65, Table 3.10 shows that the poorer shape preserving qualities tend to dominate the error, which is higher using UltraC than with HyperC.

The same effects can be seen during the reverse advection in Figures 3.47–3.48. While

Figure 3.47: *See next page.* Results for the reverse advection of Figure 3.45(a–c)(i–vi) in a shearing flow, $\tilde{u} = (-\sin(x_1)\cos(x_2), \cos(x_1)\sin(x_2))$, after $N = 250$ (i), 500 (ii), 1,000 (iii), 2,000 (iv), 3,000 (v) and 4,000 (vi) timesteps using HyperC (a), UltraC-P (b) and UltraC-S (c). Contours are shown at values of 0.025, 0.5 and 0.975.

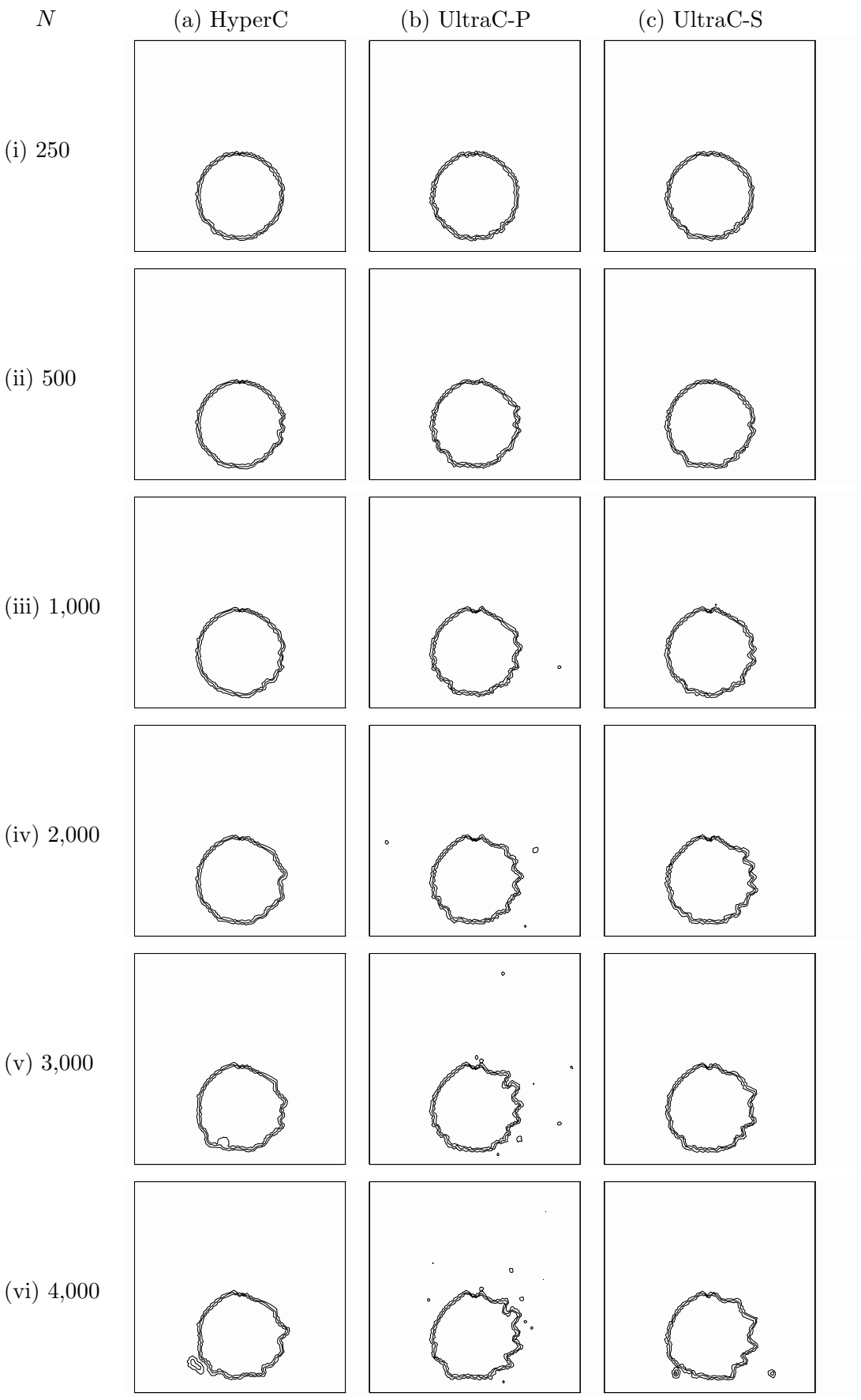


Figure 3.47: *cont.*

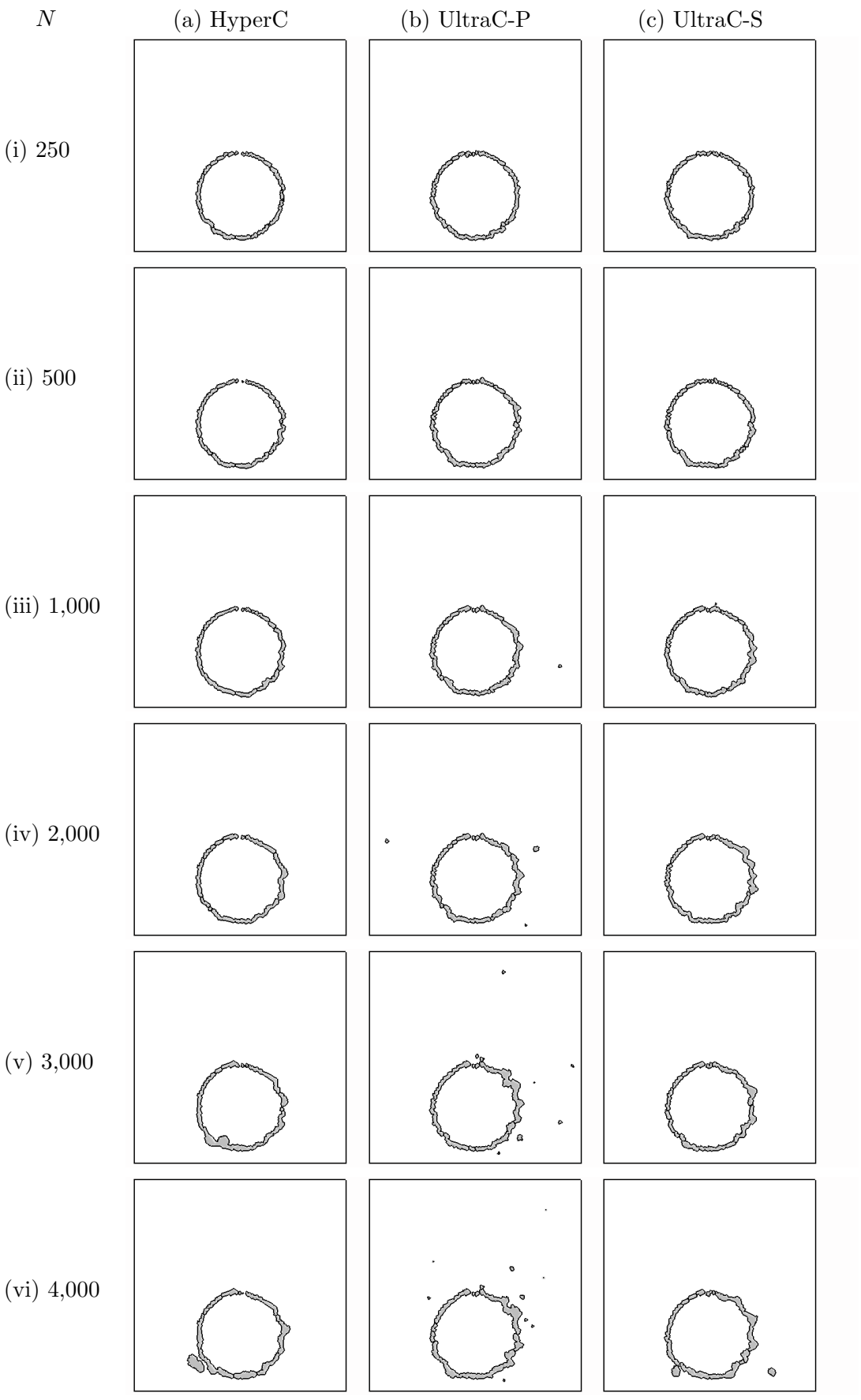


Figure 3.48: *See next page.*

Figure 3.48: *cont.* The absolute difference, $|\alpha - \alpha^e|$, between the exact solution (see Figure 3.29(a)) and the numerical solutions using HyperC (a, reproduced from Figure 3.34(b)), UltraC-P (b, see Figure 3.47(b)) and UltraC-S (c, see Figure 3.47(c)). Values above the 0.025 contour are shaded grey. The results are shown after $N = 250$ (i), 500 (ii), 1,000 (iii), 2,000 (iv), 3,000 (v) and 4,000 (vi) timesteps of forward and reverse advection.

N	HyperC	UltraC-P	UltraC-S
250	2.14×10^{-2}	3.22×10^{-2}	3.10×10^{-2}
500	2.67×10^{-2}	4.38×10^{-2}	3.78×10^{-2}
1,000	3.36×10^{-2}	5.42×10^{-2}	4.73×10^{-2}
2,000	4.03×10^{-2}	6.35×10^{-2}	6.10×10^{-2}
3,000	6.23×10^{-2}	7.94×10^{-2}	6.23×10^{-2}
4,000	8.06×10^{-2}	9.79×10^{-2}	8.91×10^{-2}

Table 3.11: Errors for a shearing flow which is reversed after N timesteps and compared to the initial condition. Results for HyperC, UltraC-P and UltraC-S are shown. Errors for other face value methods and at various resolutions and Courant numbers are presented in Tables B.2-B.3.

the trailing isolated patches remain at low values with HyperC they preserve their original bounds using UltraC. However, the original shape of the circle is not generally achieved by UltraC at the end of the simulation. Furthermore UltraC-P again shows flotsam and jetsam which increases with an increasing number of timesteps.

The poor shape preserving qualities of UltraC can be seen in the errors, calculated using Equation 3.65, in Table 3.11. UltraC-P initially performs as well as UltraC-S however with increasing shedding of flotsam and jetsam this trend is reversed. As before, both methods produce higher errors than HyperC.

The lower errors of HyperC seen in Table 3.11 are at the expense of increased production of non-physical mixed cells. This can be seen in Figure 3.49 where the fractional area of domain that has values between 0.025 and 0.975 and between 10^{-8} and $1 - 10^{-8}$ is shown. By both measures UltraC produces fewer mixed cells than HyperC.

3.5 Mesh Adaptivity

The fundamental problem seen with all material tracking techniques during the shearing flow test case (Figures 3.30, 3.32 and 3.45) occurs when the tail thickness approaches the available mesh resolution. At this point the field value tends to drop below the original maximum, separate from the main body of material, or both. As material advection algorithms treat mixed interface regions in a special way it is not surprising that during

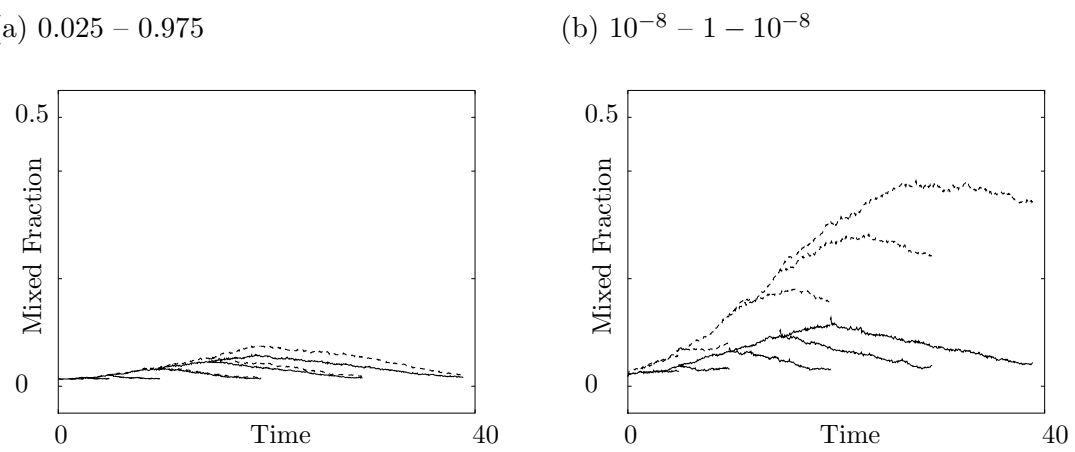


Figure 3.49: Fractional area of the domain which consists of mixed cells over time for both the forward and reverse runs of the shearing flow test. Two ranges of mixing are shown: 0.025 – 0.975 (a) and 10^{-8} – $1-10^{-8}$ (b) for HyperC (dashed) and UltraC (solid).

the reverse advection stage these isolated patches do not merge back into the original shape without introducing significant errors.

An alternative approach to dealing with the sheared tail is to attempt to increase the resolution in its vicinity with the aim of preventing it from becoming under-resolved. If the resolution is also decreased where it is not needed, away from the interface, this has the potential to substantially reduce the number of nodes required to perform the simulation, lowering the computational requirements.

The shearing flow test case was repeated using a variety of adaptive meshes. Only the material being sheared was adapted to, with a desired error of $\hat{\epsilon} = 0.11$ (see Section 2.3). Effectively this means that the mesh attempts to put sufficient resolution into the domain to achieve an 11% 'error' in the solution. A lower value of $\hat{\epsilon}$ would result in higher resolution around the interface while a higher value would lead to lower resolution. However, as the only variation in the field occurs at the interface over a single grid point the discrete Hessian defining the metric (Equation 2.50) tends to always find a high gradient regardless of the desired error. It was therefore found necessary to limit the minimum edge lengths, $\hat{l}_{i\min}$, to 0.01 in all dimensions (i.e. approximately a third of the average grid spacing used in the fixed mesh runs). The maximum edge lengths, $\hat{l}_{1\max}$, were also limited to 1.5.

Using the initial conditions shown in Figure 3.50 the simulation was run forward using HyperC with the shearing velocity field, $\tilde{u} = (\sin(x_1)\cos(x_2), -\cos(x_1)\sin(x_2))$, at a Courant number of 0.25 as in the fixed mesh example (Figure 3.30). However, as the Courant number is mesh dependent the number of timesteps taken in the fixed mesh case does not correspond to the number of timesteps on the adaptive mesh. Therefore the simulation was stopped at times roughly equivalent to N = 250, 500, 1,000, 2,000, 3,000 and 4,000 timesteps in the fixed mesh simulations. At these points the flow was reversed, $\tilde{u} = (-\sin(x_1)\cos(x_2), \cos(x_1)\sin(x_2))$, and the field was advected back to its

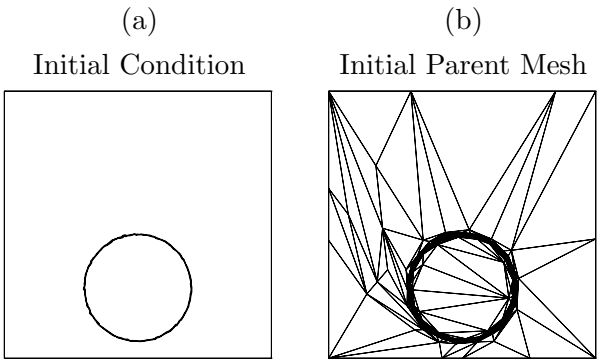


Figure 3.50: The initial condition (a) and initial adapted parent mesh (b) used in the adaptive shearing flow advection test.

initial position.

The mesh was adapted every 10 timesteps to maintain sufficient resolution as the field was advected. As discussed in Section 2.4, when this mesh optimisation takes place the field must be interpolated to the new mesh. Consistent interpolation was found to be too dissipative, smearing the interface during advection. Additionally it is not conservative. Projection between the meshes is conservative but introduces unbounded spikes to the solution.

The field being advected is not piecewise linear continuous, instead it is represented on a control volume mesh that is dual to a linear mesh. The nodes of these two meshes are co-located and the integral of the nodal values on the control volume mesh is the same as on the piecewise linear mesh. This fact was exploited in the assembly of the control volume mass matrix, where the lumped mass matrix from the linear mesh was used (see Section 3.3). As the linear lumped mass and full mass matrices share the same conservation properties the same logic that applied to the lumped and bounded projection methods (see Section 2.4) can also be applied to fields on the control volume mesh dual to the linear mesh.

Using the lumped mass during projection does indeed bound the field and is conservative. However, it was also found to be too dissipative, adversely affecting the sharpness of the interface. Hence, the minimally dissipative and conservative bounded projection method was used.

For fields with rapidly changing values within isolated regions, such as the one advected here, the minimally dissipative bounded projection algorithm was found to have slower convergence rates than those reported in Farrell *et al.* (2009), where smoothly varying fields were tested. The algorithm depends on smearing deviations between nearest neighbours (it is thus a locality preserving method) at each iteration until sufficient capacity is found to absorb the unboundedness. Convergence is therefore not monotonic and will be particularly slow where the absorptive capacity is isolated within a small region, for example around an interface. For this reason a less severe boundedness tolerance of 10^{-6} was enforced. After this the local smoothing of deviations using the lumped

mass matrix was stopped. This typically took between 2,000 and 3,000 iterations.

To ensure strict boundedness after a step of mesh optimisation a non-local redistribution of any remaining deviations was implemented. First, the deviation, α_{dev}, is recalculated:

$$\alpha_{\text{dev}} = \begin{cases} \alpha - \alpha_{\max} & \text{if } \alpha > \alpha_{\max}, \\ \alpha - \alpha_{\min} & \text{if } \alpha < \alpha_{\min}, \\ 0 & \text{if } \alpha_{\min} < \alpha < \alpha_{\max}, \end{cases} \tag{3.93}$$

where $\alpha_{\max}$ and $\alpha_{\min}$ are the maximum and minimum values of the lumped projection solution on a local patch of nodes (as in Section 2.4). The absorptive capacities of the nodes are also calculated using these maximum and minimum values, such that:

$$\alpha_{\text{-}} = \begin{cases} \alpha - \alpha_{\max} & \text{if } \alpha > \alpha_{\max}, \\ 0 & \text{if } \alpha < \alpha_{\min}, \\ \alpha - \alpha_{\min} & \text{if } \alpha_{\min} < \alpha < \alpha_{\max}, \end{cases} \tag{3.94a}$$

$$\alpha_{+} = \begin{cases} 0 & \text{if } \alpha > \alpha_{\max}, \\ \alpha_{\min} - \alpha & \text{if } \alpha < \alpha_{\min}, \\ \alpha_{\max} - \alpha & \text{if } \alpha_{\min} < \alpha < \alpha_{\max}, \end{cases} \tag{3.94b}$$

where $\alpha_{\text{-}}$ is the capacity of a node to decrease its value and α_{+} is the capacity of a node to increase its value. Some potential capacity has been ignored here. For instance, when $\alpha > \alpha_{\max}$ the decrease capacity, $\alpha_{\text{-}}$, could be set to $\alpha - \alpha_{\min}$, potentially reducing that node's value to the minimum bound. However this was found to introduce too much diffusion to the solution and so the upper bound was chosen. Similarly, the lower bound was chosen for the increase capacity, α_{+}.

Using these node-wise capacities, the deviations can then be redistributed by looping over the nodes moving their overshoots and undershoots to cells with increase and decrease capacities respectively. Note that to be conservative this process must take account of the ratio between the areas/volumes of each cell. After each node-wise redistribution the deviation, α_{dev}, and increase and decrease capacities, $\alpha_{\text{-}}$ and α_{+}, must be recalculated. This process takes at most n_s^2 redistributions, where n_s is the number of solution nodes. However, as only a few nodes are likely to have deviations, in practice it takes far fewer. The process is non-local and material may be redistributed anywhere on the mesh. Its use is therefore restricted to the small deviations ($< 10^{-6}$) remaining after the local bounding process.

The result of advecting the circle in Figure 3.50 forward in a shearing flow is shown in Figure 3.51 alongside the parent adapted mesh at the corresponding timesteps. The mesh can be seen to remain focussed around the interface, which also remains sharp with the contours shown staying closely packed. The tail of the sheared circle is also seen to remain contiguous and within the original bounds ($[0, 1]$) of the field. However, some interaction between the mesh and the advection can also be seen. The interface appears to develop corners corresponding to the orientation of the edges of the mesh.

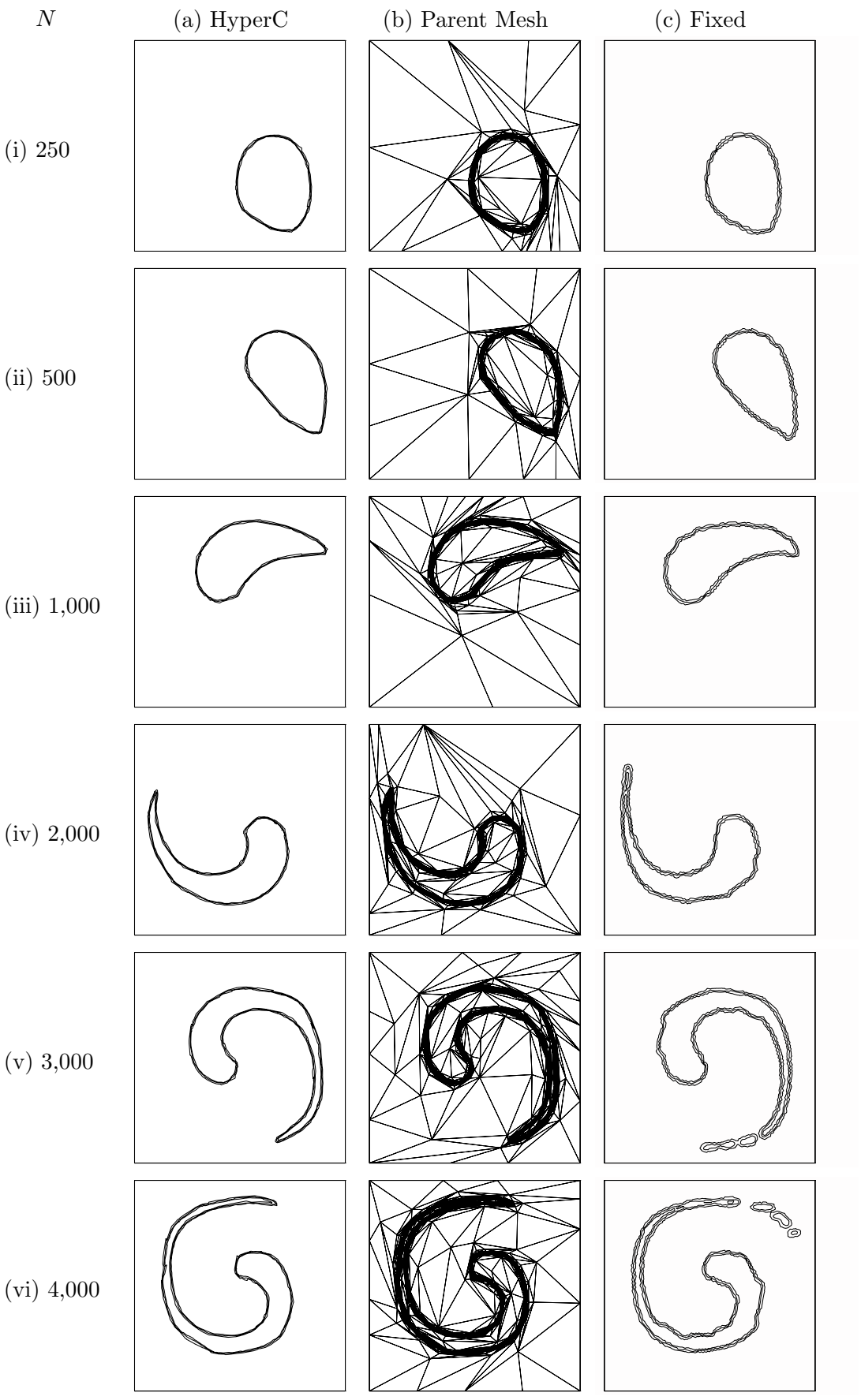


Figure 3.51: *See next page.*

Figure 3.51: *cont.* Results for the forward advection of Figure 3.50(a) in a shearing flow, $\tilde{u} = (\sin(x_1)\cos(x_2), -\cos(x_1)\sin(x_2))$, after $N = 250$ (i), 500 (ii), 1,000 (iii), 2,000 (iv), 3,000 (v) and 4,000 (vi) equivalent timesteps using HyperC. Contours are shown at values of 0.025, 0.5 and 0.975 (a) alongside the adapted parent mesh at each time (b). For comparison the results on a fixed mesh from Figure 3.30 are shown in (c).

This distortion of the interface becomes more extreme during the reverse advection (see Figure 3.52). Only for the shortest runs, equivalent to $N = 250$ and 500 (Figure 3.52(a)(i, ii)), does the interface return to a shape resembling the initial circle. Over longer time periods the final solution bears little relation to the initial condition other than approximate location.

The poor results seen in Figures 3.51 and 3.52 are a result of the mesh optimisation process being based on the current state of the field α. Hence the mesh is only optimal for that field straight after the optimisation takes place. However the field is then advected for a further 10 timesteps, during which time the mesh is no longer optimal for the field. In particular, the anisotropic mesh optimisation aligns the elements of the parent mesh along the interface. This results in faces on the dual control volume mesh also being aligned along the interface. While this reduces the number of nodes required to represent the field, it also influences the subsequent advection. In essence, the adapted mesh guides the advection of the field, hence distorting the interface. Repeated mesh optimisation and advection steps reinforce these distortions in a feedback loop.

Ideally, the feedback cycle would be broken by adapting the mesh to both the current and future solutions, placing resolution where it is necessary to achieve a desired error at some point in the future. Such goal based adaptivity (Power *et al.*, 2006) is however expensive and beyond the scope of this project. Two easier methods are instead used here: metric advection and edge length gradation.

As the field is being advected by a known velocity field it is possible to advect the metric forward in time at the same rate. This advected metric gives an estimate of the mesh requirements of the field in the future and can be merged with the current metric using the same superposition technique illustrated in Figure 2.6.

For simplicity tensor twisting terms are ignored and each symmetric component of

Figure 3.52: *See next page.* Results for the reverse advection of Figure 3.51(a–b)(i–vi) in a shearing flow, $\tilde{u} = (-\sin(x_1)\cos(x_2), \cos(x_1)\sin(x_2))$, after $N = 250$ (i), 500 (ii), 1,000 (iii), 2,000 (iv), 3,000 (v) and 4,000 (vi) equivalent timesteps using HyperC. Contours are shown at values of 0.025, 0.5 and 0.975 (a) alongside the absolute difference, $|\alpha - \alpha^e|$, between the numerical solutions and the exact solution (b, see Figure 3.50(a)) on the super mesh (see Figure 3.53(b)). Differences greater than the 0.025 contour are shaded grey.

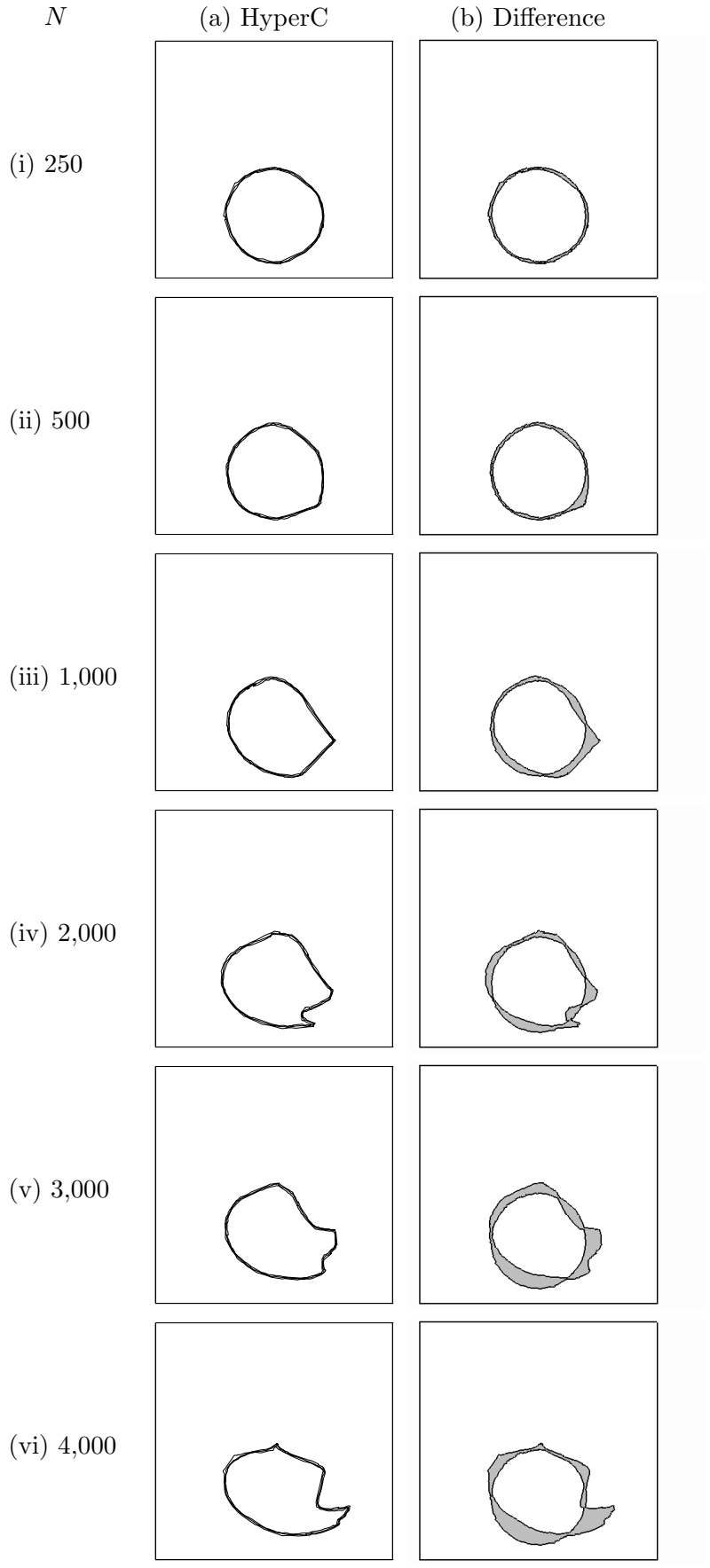


Figure 3.52: *cont.*

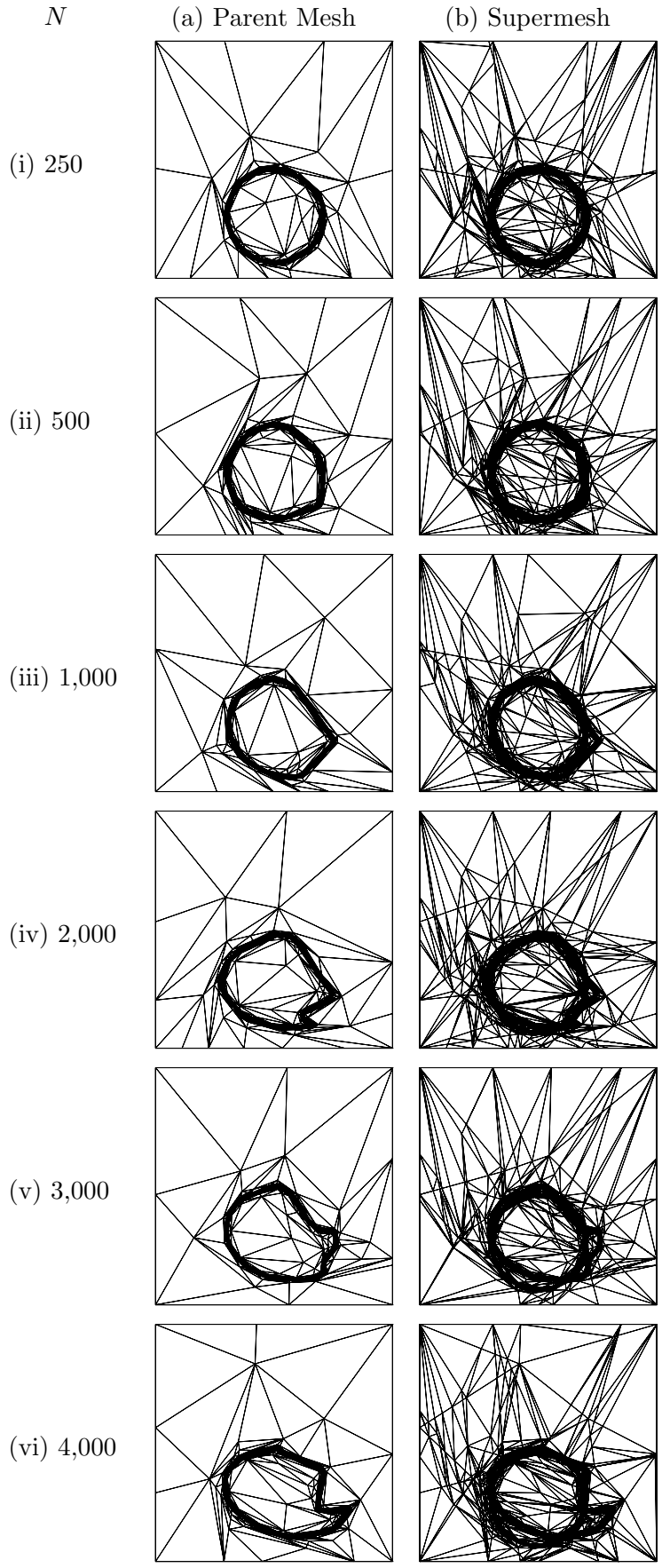


Figure 3.53: *See next page.*

Figure 3.53: *cont.* (a) Adapted parent meshes corresponding to the timeframes shown in Figure 3.52(i–vi), the final timestep in the reverse advection of Figure 3.51(a)(i–vi) in a shearing flow, $\tilde{u} = (-\sin(x_1)\cos(x_2), \cos(x_1)\sin(x_2))$, after $N = 250$ (i), 500 (ii), 1,000 (iii), 2,000 (iv), 3,000 (v) and 4,000 (vi) equivalent timesteps using HyperC. (b) The supermesh of the meshes shown in (a) with the initial mesh from Figure 3.50(b), which was used to calculate the errors.

the metric, $M_{\alpha\,ij} \forall i, j \geq i$, is advected individually. Equation 2.35 is solved in Ω using first-order upwinding over the time period $[t, t + \gamma T_{\text{adapt}}]$, where t is the current time, T_{adapt} is an estimate of the amount of time until the next step of mesh optimisation and γ is a scaling factor to increase the distance the metric is advected. As the mesh is optimised every ten timesteps $T_{\text{adapt}} \approx 10\Delta t$. To ensure sufficient resolution ahead of the interface $\gamma = 3$ was found to give the best results. The current metric is used as the initial condition at the beginning of the advection.

As first-order upwinding is used in Equation 2.35 it may be solved implicitly with $\theta = 1/2$ and $\theta_p = 1/2$. This eases the timestep restriction so a new larger timestep, Δt_{metric}, may be defined. Generally this was taken so that the maximum Courant number was 2.5. The shearing test uses a simulation timestep, Δt, such that a maximum Courant number of 0.25 is achieved. Hence approximately three timesteps of metric advection, Δt_{metric}, are required to advect the metric γT_{adapt} forward in time. To ensure an even distribution of resolution over the advected distance, rather than resolution focussed around the initial interface and around the interface at $t + \gamma T_{\text{adapt}}$, the metrics are merged after each timestep, Δt_{metric}.

Superposing successive metrics removes some of the anisotropy of the initial mesh, limiting the effect it can have on the interface shape during advection. Furthermore metric advection, using a diffusive scheme like first-order upwinding, automatically introduces a degree of edge length gradation into the mesh. As the bounds of the metric decay with advection, the equivalent edge length increases thus smoothing the transition from high resolution to low resolution regions.

Gradation is also enforced directly by restricting the rate at which the edge lengths are allowed to change between elements to a gradient of 1 in all directions. This is imposed by looping over all node pairs in the mesh and adjusting the requested edge lengths until the gradient restriction is satisfied (Li, 2003).

The updated initial condition and parent mesh for the shearing flow test case are shown in Figure 3.54. Using metric advection with $\gamma = 3$ and an edge length gradient restriction of 1, the number of nodes has increased from 429 in Figure 3.50(b) to 1,360 in Figure 3.54(b). However this is still far fewer than the 11,524 nodes used in the fixed mesh equivalent (Figure 3.29(b)).

The results from the forward advection of Figure 3.54(a) under a shearing flow, $\tilde{u} = (\sin(x_1)\cos(x_2), -\cos(x_1)\sin(x_2))$, are shown in Figure 3.55 after an amount of time equivalent to 250, 500, 1,000, 2,000, 3,000 and 4,000 timesteps of the fixed mesh

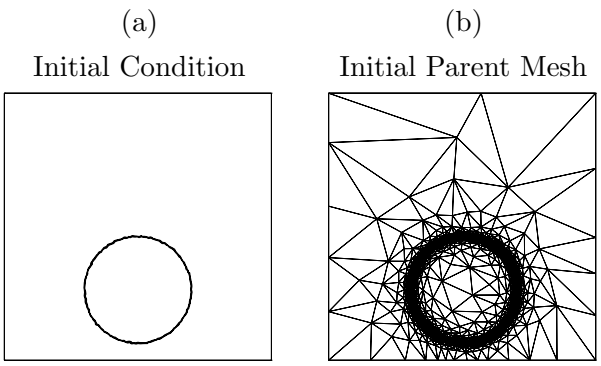


Figure 3.54: The initial condition (a) and initial adapted parent mesh using metric advection and gradation (b) used in the adaptive shearing flow advection test.

experiment (Figure 3.30). As in the results using a simple adaptive mesh (Figure 3.51) the mesh can be seen to follow the movement of the interface closely and the contours remain tightly packed throughout the simulation. However, metric advection and gradation prevent the interaction between the advection and adapted mesh from warping the interface, which remains smooth throughout the forward advection.

Using the improved adapted mesh the reverse advection is also much more successful at returning the sheared shape back to the initial conditions (see Figure 3.56). Only after the equivalent of 3,000 timesteps does significant distortion become evident. However, even at this stage the contours are still closely packed and there are no isolated mixed cells.

The enhanced results using the improved adaptive mesh as opposed to the simple adaptive mesh can be seen by calculating the errors using Equation 3.65, with $\alpha^e = \alpha_0$ as before. However, on an adaptive mesh this is complicated by the fact that the final solution, α, and the initial condition, α_0, are on different meshes (compare Figure 3.57(a)(vi) and Figure 3.54(b) for instance). Therefore $\int_\Omega |\alpha - \alpha_0| \mathrm{d}V$ cannot easily be computed on either mesh. Instead the supermesh of the initial and final meshes is constructed (see Figures 3.53(b) and 3.57(b)). As this contains all the nodes of both meshes, the initial and final fields may be interpolated conservatively onto the supermesh where the absolute difference between them, $|\alpha - \alpha^e|$, can be found (see Figures 3.52(b) and 3.56(b)). Equation 3.65 can then be evaluated on the supermesh.

Figure 3.55: *See next page.* Results for the forward advection of Figure 3.54(a) in a shearing flow, $\tilde{u} = (\sin(x_1)\cos(x_2), -\cos(x_1)\sin(x_2))$, after $N = 250$ (i), 500 (ii), 1,000 (iii), 2,000 (iv), 3,000 (v) and 4,000 (vi) equivalent timesteps using HyperC. Contours are shown at values of 0.025, 0.5 and 0.975 (a) alongside the adapted parent mesh at each time (b). For comparison the results on a fixed mesh from Figure 3.30 are shown in (c).

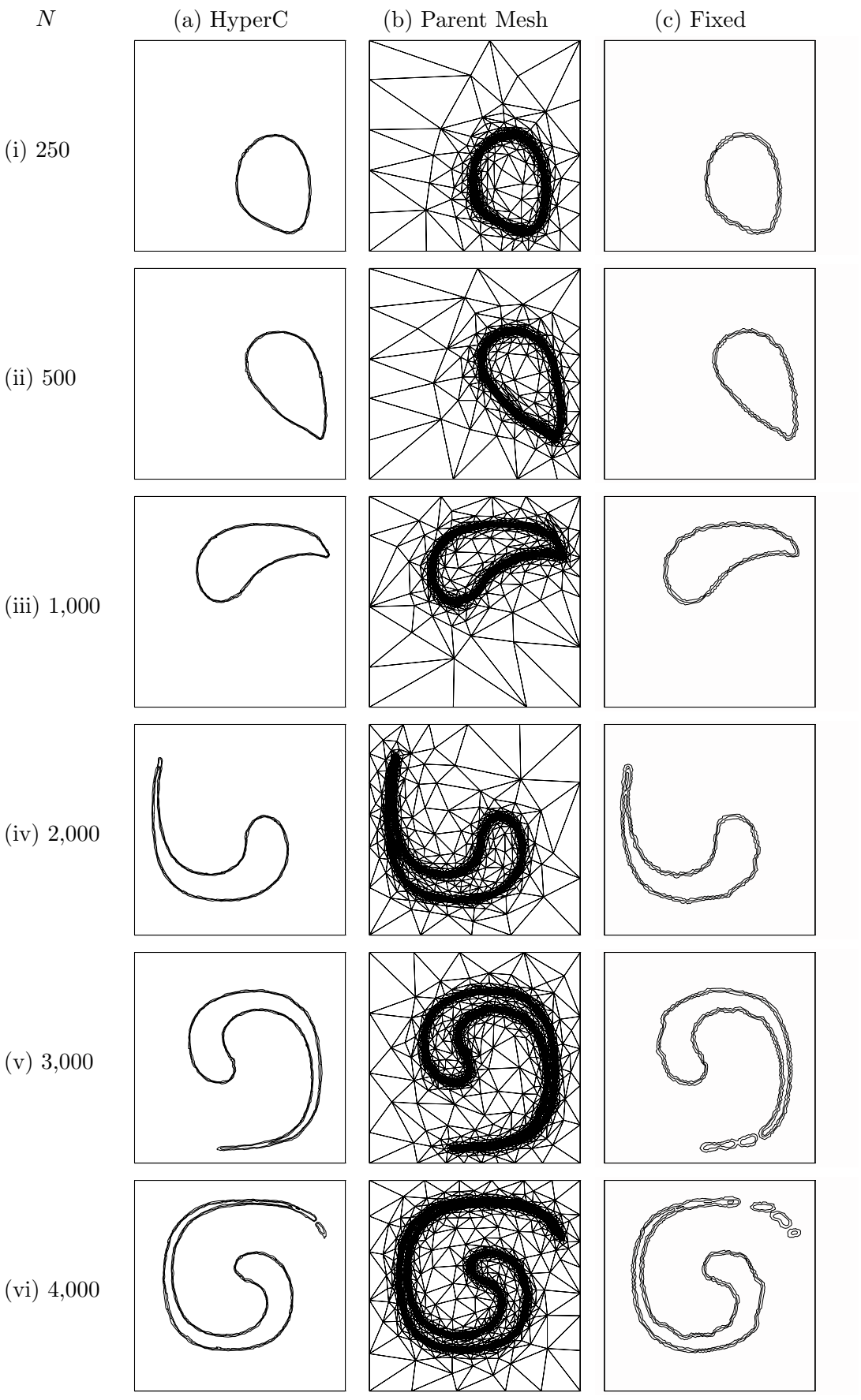


Figure 3.55: *cont.*

	Fixed Mesh		Simple Adaptive Mesh		Improved Adaptive Mesh	
N	Error	Nodes	Error	Max. Nodes	Error	Max. Nodes
250	2.14×10^{-2}	11,524	3.61×10^{-2}	485	2.00×10^{-2}	1,360
500	2.67×10^{-2}	11,524	6.21×10^{-2}	485	1.99×10^{-2}	1,360
1,000	3.36×10^{-2}	11,524	1.54×10^{-1}	507	3.98×10^{-2}	1,360
2,000	4.03×10^{-2}	11,524	1.66×10^{-1}	757	3.99×10^{-2}	1,834
3,000	6.23×10^{-2}	11,524	3.00×10^{-1}	1,018	6.09×10^{-2}	2,371
4,000	8.06×10^{-2}	11,524	2.99×10^{-1}	1,229	1.16×10^{-1}	2,956

Table 3.12: Errors for a shearing flow which is reversed after the equivalent of N timesteps and compared to the initial condition. Results for HyperC on an unstructured fixed mesh, a simple adaptive mesh and the improved adaptive mesh, with metric advection and gradation, are shown. Errors at a variety of Courant numbers and using different adaptivity settings are presented in Tables B.4–B.5.

The errors calculated using the supermesh are shown in Table 3.12. They demonstrate the improved behaviour of the solution using metric advection and edge length gradation. Most importantly, they show that errors equivalent to the shearing test case on a fixed mesh can be achieved with far fewer nodes and hence fewer computational resources.

3.6 Summary

In this chapter various methods of selecting a face value for finite volume advection were reviewed using a number of different sensor and flux limiting functions. Investigating the application of these to material advection an explicit scheme known as HyperC was shown to be optimal and comparable to standard interface reconstruction techniques at advecting step functions in one dimension. Importantly, HyperC requires no expensive interface reconstruction.

The issues extending one-dimensional flux limiting techniques to multiple dimensions were discussed and a control volume mesh was designed around the parent simplex mesh that overcame these corner coupling problems. On this mesh it was possible to then

Figure 3.56: *See next page.* Results for the reverse advection of Figure 3.55(a–b)(i–vi) in a shearing flow, $\tilde{u} = (-\sin(x_1)\cos(x_2), \cos(x_1)\sin(x_2))$, after $N = 250$ (i), 500 (ii), 1,000 (iii), 2,000 (iv), 3,000 (v) and 4,000 (vi) equivalent timesteps using HyperC. Contours are shown at values of 0.025, 0.5 and 0.975 (a) alongside the absolute difference, $|\alpha - \alpha^e|$, between the numerical solutions and the exact solution (b, see Figure 3.54(a)) on the super mesh (see Figure 3.57(b)). Differences greater than the 0.025 contour are shaded grey.

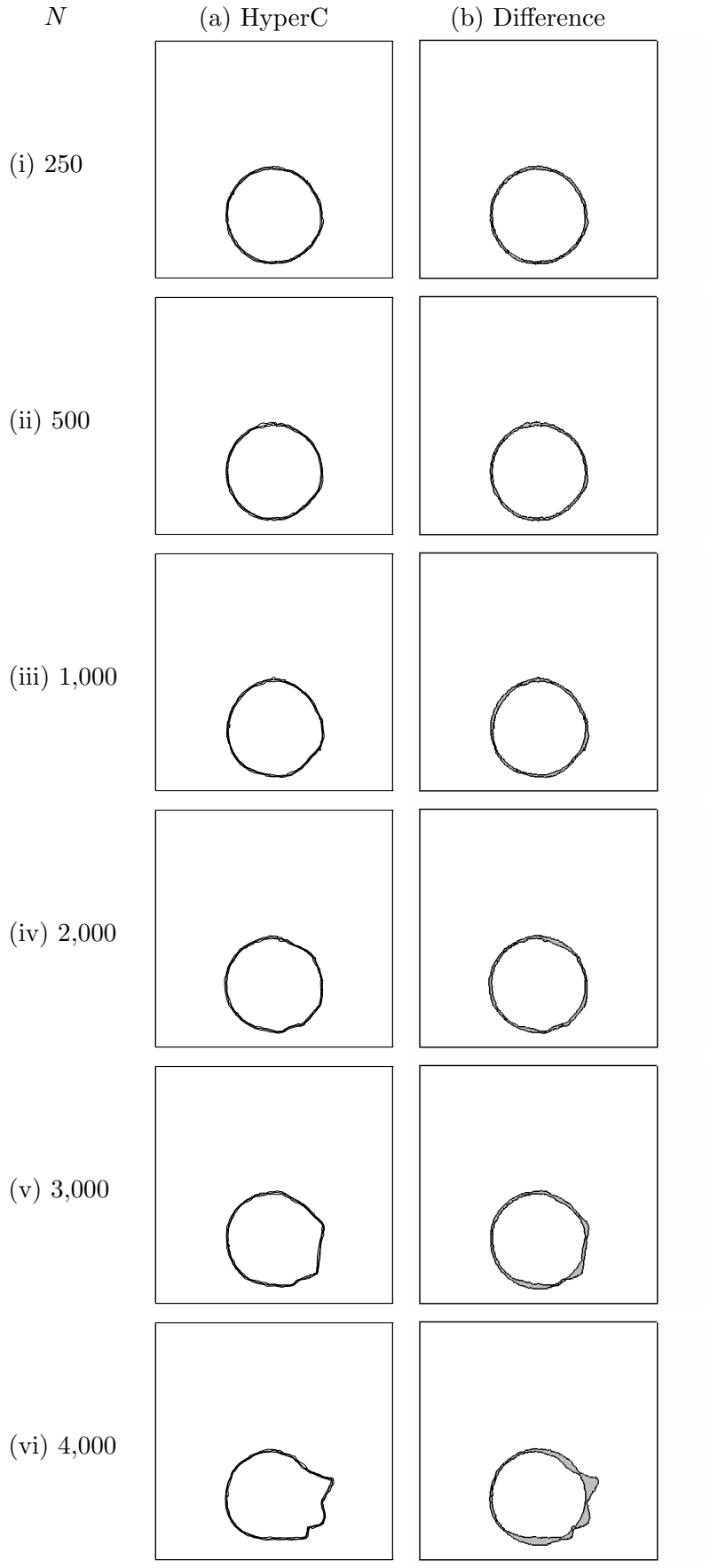


Figure 3.56: *cont.*

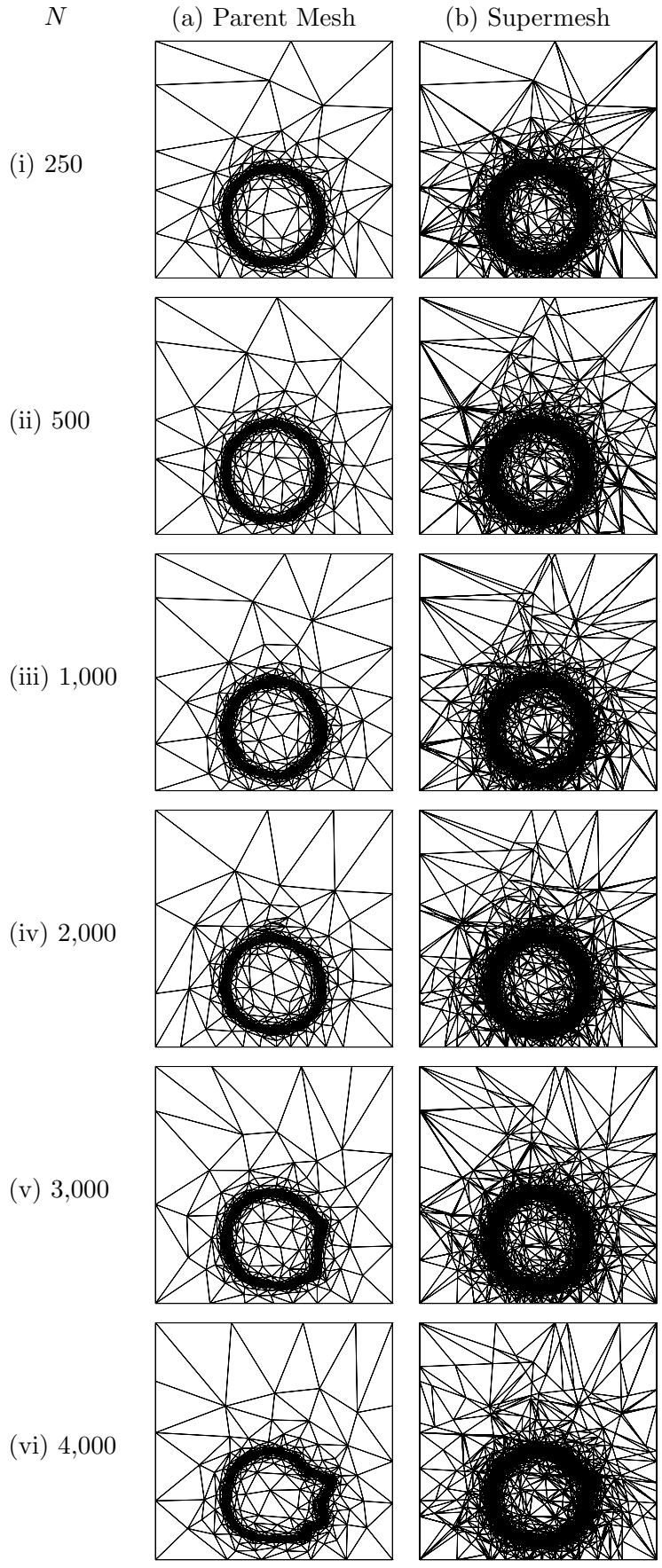


Figure 3.57: *See next page.*

Figure 3.57: *cont.* (a) Adapted parent meshes corresponding to the timeframes shown in Figure 3.56(i–vi), the final timestep in the reverse advection of Figure 3.55(a)(i–vi) in a shearing flow, $\tilde{u} = (-\sin(x_1)\cos(x_2), \cos(x_1)\sin(x_2))$, after $N = 250$ (i), 500 (ii), 1,000 (iii), 2,000 (iv), 3,000 (v) and 4,000 (vi) equivalent timesteps using HyperC. (b) The supermesh of the meshes shown in (a) with the initial mesh from Figure 3.54(b), which was used to calculate the errors.

extend HyperC to advect materials in higher dimensions, using a suitable definition of the Courant number, without the need for operator splitting. This technique was then tested on unstructured meshes and compared against standard structured mesh techniques using interface reconstruction and split advection. HyperC was seen to perform extremely well in comparison.

When the distance between two interfaces approached the mesh resolution, for example during a shearing flow advection test problem, HyperC leads to lower cell values, equivalent to non-physical mixed cells with a volume fraction. Various techniques were discussed to prevent the development of these, including extending the boundedness criterion so that the total variation was bounded rather than guaranteed diminishing. This led to the development of a new flux limiting scheme, UltraC, the performance of which was also evaluated using several advection tests. It was found to have poorer shape preserving qualities than HyperC but was more successful at preventing the development of mixed cells.

Finally an adaptive mesh was used to focus resolution around the interface, thereby preventing the development of isolated patches of the advected field. A number of issues using the adaptive mesh were encountered. Optimal results were found using metric advection and edge length gradation. Additionally, to maintain both conservation and the sharpness of the interface a bounded projection method was required to project the volume fraction field from the old mesh to the new mesh. The suitability of this projection technique was discussed for the control volume mesh and shown to be valid due to its relationship with the linear parent mesh.

All the tests in this chapter have used a prescribed analytical velocity field. Clearly in a multi-material flow, where the field represents a volume fraction of material, the material advection should instead be coupled to the solution of the momentum equations. This is discussed in the following chapter (Chapter 4). Additionally, only the advection of a single field has been discussed here; however, when considering more than two materials multiple volume fractions must be advected. Determining how they interact with each other will be considered in Chapter 5.

Chapter 4

Flow Model

In the previous chapter new methods of advecting material properties, such as volume fractions, were developed and tested assuming a velocity field was known *a priori*. In a multi-material model of geophysical flow a two way coupling exists between these fields and the momentum equations (2.1). The volume fractions specify the locations of regions with different material properties, such as density. Through terms, like the buoyancy, in the momentum equation these drive and influence the velocity field. Completing the cycle the updated velocity field affects the movement of the materials through the advection equation.

A useful fully coupled model must achieve the properties of boundedness, stability and conservation. Boundedness relates primarily to the volume fraction, which must remain between the values of zero and one to be physically meaningful. Stability ensures that a solution can be found and is not contaminated by numerical artefacts. It relates both to the solution of the material advection equation and to the solutions from the momentum equation. Conservation is a property of the coupled system that requires the momentum and the materials' mass be conserved both globally and individually.

A bounded method of advecting volume fractions on unstructured meshes was suggested in Chapter 3. In all the examples of its use this maintained an upper limit on the field of one and a lower limit of zero. Additionally, it is conditionally stable, dependent on the Courant number being less than one.

The momentum equations are themselves a coupled system of velocity and pressure. The stability of this system is dependent on a number of factors. In this chapter various standard discretisations for the momentum equations are discussed and tested. The choice of discretisation influences the conservation properties of the full multi-material system. Modifications to the standard methods that ensure material conservation are investigated and a suitable coupled discretisation suggested. The effectiveness of this discretisation is then demonstrated in comparison to experimental data in two and three dimensions using both fixed and adaptive meshes.

4.1 Projection Method

In Chapter 2 the full system of momentum and mass equations (2.1) was simplified to the case of an isotropic, isothermal, Newtonian, incompressible fluid (Equations 2.7a). Then, various methods of discretising the momentum equation (2.7a(a)) were considered, resulting in a finite element representation of the equation (2.21). However, the discretisation of the conservation of mass equation and its coupling to the solution of the momentum equation was not considered.

For an incompressible fluid the conservation of mass equation effectively becomes a conservation of volume or area requirement, meaning that the velocity field must be divergence free:

$$u_{i,i} = 0. \tag{4.1}$$

In general, the numerical solution to the momentum equation (2.21) will not satisfy this requirement.

Any vector field, such as a velocity, u_i^*, may be split into a divergence free part, u_i, and a curl free part, $\lambda_{,i}$, via a Helmholtz decomposition (Temam, 2001):

$$u_i^* = \underbrace{u_i}_{\text{divergence free}} + \underbrace{\lambda_{,i}}_{\text{curl free}}. \tag{4.2}$$

If Equation 4.2 is rearranged to be in terms of the gradient of λ:

$$-\lambda_{,i} = u_i - u_i^*, \tag{4.3}$$

then by taking the divergence:

$$-\lambda_{,ii} = -u_{i,i}^*, \tag{4.4}$$

it is possible, through Equation 4.1, to eliminate u_i from the equation. Solving Equation 4.4 for λ given u_i^*, the gradient of λ can then be used to find the divergence free field u_i:

$$u_i = u_i^* - \lambda_{,i}. \tag{4.5}$$

As in Chapter 2, Equation 4.3 may be discretised using finite elements. First it is multiplied throughout by a test function, ω_i, and integrated over the domain, Ω:

$$-\int_\Omega \omega_i \lambda_{,i} \mathrm{d}V = \int_\Omega \omega_i u_i \mathrm{d}V - \int_\Omega \omega_i u_i^* \mathrm{d}V. \tag{4.6}$$

Then the continuous variables ω_i, u_i and λ are discretised onto their finite dimensional subspaces: $\omega_i = N_A \omega_{iA}$, $u_i = N_A u_{iA}$, $u_i^* = N_A u_{iA}^*$ and $\lambda = M_A \lambda_A$. Here λ is represented using a potentially different set of basis functions, M_A, to the velocity, N_A. For example, velocity may be piecewise quadratic while λ is piecewise linear, this is often termed a Taylor-Hood element pair (Taylor & Hood, 1973).

Using the discretised variables and discarding the trivial solution, $\omega_{iA} = 0 \,\forall\, A, i$, Equation 4.6 may be written in matrix form as:

$$\mathbf{C}_i \lambda = \mathbf{M_u} u_i - \mathbf{M_u} u_i^*, \tag{4.7}$$

where:

$$\mathbf{c}_{iab} = -\int_{\Omega_e} N_a M_{b,i} \mathrm{d}V \qquad \Rightarrow \mathbf{C}_{iAB} \qquad \text{gradient matrix} \tag{4.8a}$$

$$\mathbf{m}_{\mathbf{u}ab} = \int_{\Omega_e} N_a N_b \mathrm{d}V \qquad \Rightarrow \mathbf{M}_{\mathbf{u}}{}_{AB} \qquad \text{velocity mass matrix,} \tag{4.8b}$$

where, as in Chapter 2, lower case subscripted, ab, indicates the node numbers local to the element and $\Rightarrow$ indicates the assembly and indexing processes that translates these into upper case subscripted, AB, global indices.

Multiplying Equation 4.7 throughout by the inverse of the velocity mass matrix and a discrete divergence operator, $\mathbf{D}$, Equation 4.4 may be discretised consistently giving:

$$\mathbf{D}_i {\mathbf{M}_{\mathbf{u}}}^{-1} \mathbf{C}_i \lambda = \mathbf{D}_i u_i^* - \mathbf{D}_i u_i, \tag{4.9}$$

where:

$$\mathbf{d}_{iab} = \int_{\Omega_e} M_a N_{b,i} \mathrm{d}V \qquad \Rightarrow \mathbf{D}_{iAB} \qquad \text{divergence matrix.} \tag{4.10}$$

If the divergence matrix, $\mathbf{D}_i$, is integrated by parts (assuming the finite element space represented by M_A is continuous):

$$\mathbf{d}_{iab} = -\int_{\Omega_e} M_{a,i} N_b \mathrm{d}V + \int_{\partial\Omega_e \in \partial\Omega_h} M_a N_b n_i \mathrm{d}s \quad \Rightarrow \mathbf{D}_{iAB} \quad \text{divergence matrix,} \tag{4.11a}$$

the Dirichlet boundary conditions on the velocity may be incorporated into a right hand side term, $\mathbf{d}_\partial$:

$$d_{\partial_a} = -\int_{\partial\Omega_e \in \partial\Omega_{g_i}} M_a d_i n_i \mathrm{d}s \qquad \Rightarrow \mathbf{d}_{\partial A} \quad \text{divergence right hand side.} \tag{4.11b}$$

Hence, the continuity equation (4.1) may be represented in discrete form as:

$$\mathbf{D}_i u_i = \mathbf{d}_\partial. \tag{4.12}$$

Ignoring boundary terms, the matrices defined in Equations 4.8a and 4.11a are the transpose of each other, so Equations 4.9 and 4.12 can be rewritten:

$$\mathbf{C}_i^T {\mathbf{M}_{\mathbf{u}}}^{-1} \mathbf{C}_i \lambda = \mathbf{C}_i^T u_i - \mathbf{C}_i^T u_i^*, \tag{4.13a}$$

$$\mathbf{C}_i^T u_i = \mathbf{d}_\partial, \tag{4.13b}$$

which can be combined into:

$$\mathbf{C}_i^T {\mathbf{M}_{\mathbf{u}}}^{-1} \mathbf{C}_i \lambda = \mathbf{d}_\partial - \mathbf{C}_i^T u_i^*. \tag{4.13c}$$

The divergence free field may be regained again by returning to Equation 4.7 and solving:

$$\mathbf{M}_{\mathbf{u}} u_i = \mathbf{M}_{\mathbf{u}} u_i^* + \mathbf{C}_i \lambda \tag{4.14}$$

for u_i.

Hence, given a velocity field, u_i^*, represented on some mesh, it is possible to use Equations 4.13c and 4.14 to project it to a divergence free subspace, u_i. $\mathbf{C}_i^T {\mathbf{M_u}}^{-1} \mathbf{C}_i$ acts as a projection matrix while the new variable λ, is a Lagrange multiplier designed to maintain the incompressibility of the flow. However, λ by itself has no physical meaning. Its nearest physical equivalent is a normal force acting equally in all directions: the pressure, $p = \sigma_{ii}/d$.

The pressure only appears in the momentum equation as a pressure gradient term (from Equation 2.19d):

$$g_{ai} = -\int_{\Omega_e} N_a p_{,i} \mathrm{d}V \qquad \Rightarrow \mathbf{g}_{Ai} \qquad \text{pressure gradient.} \tag{4.15}$$

Its equivalence to λ may be seen by discretising this term further, representing pressure on the same basis functions, $p = M_A p_A$:

$$g_{ai} = -\int_{\Omega_e} N_a M_{b,i} \mathrm{d}V p_b \qquad \Rightarrow \mathbf{C}_i p \qquad \text{pressure gradient.} \tag{4.16}$$

Rewriting Equation 2.21 with the pressure gradient term in the form of Equation 4.16, and recognising that the pressure varies with time, produces:

$$\mathbf{M}\frac{u_i^{n+1} - u_i^n}{\Delta t} + \theta \mathbf{N} u_i^{n+1} + \theta \mathbf{K} u_i^{n+1} = -\left(1-\theta\right)\mathbf{N} u_i^n - \left(1-\theta\right)\mathbf{K} u_i^n + \mathbf{C}_i p^{n+\frac{1}{2}} + \mathbf{f}_i + \mathbf{b}_i. \tag{4.17}$$

It now becomes clear that the right hand side of Equation 4.17 contains an unknown variable, p. Therefore it is only possible to solve for a best guess at the velocity, u_i^*, using the most up to date pressure field available, p^*:

$$\mathbf{M}\frac{u_i^* - u_i^n}{\Delta t} + \theta \mathbf{N} u_i^* + \theta \mathbf{K} u_i^* = -\left(1-\theta\right)\mathbf{N} u_i^n - \left(1-\theta\right)\mathbf{K} u_i^n + \mathbf{C}_i p^* + \mathbf{f}_i + \mathbf{b}_i. \tag{4.18}$$

If the advection and stress terms are treated semi-implicitly then the full momentum equation (4.17) becomes:

$$\mathbf{M}\frac{u_i^{n+1} - u_i^n}{\Delta t} + \theta \mathbf{N} u_i^* + \theta \mathbf{K} u_i^* = -\left(1-\theta\right)\mathbf{N} u_i^n - \left(1-\theta\right)\mathbf{K} u_i^n + \mathbf{C}_i p^{n+\frac{1}{2}} + \mathbf{f}_i + \mathbf{b}_i. \tag{4.19}$$

Here semi-implicit means that the advection and stress terms are treated implicitly according to θ within the solution to the momentum equation. However, they will not be treated implicitly within any correction from u_i^* to u_i^{n+1}.

The difference between the desired equation (4.19) and the currently solvable equation (4.18) is given by:

$$\mathbf{C}_i \left(p^{n+\frac{1}{2}} - p^*\right) \Delta t = \mathbf{M} u_i^{n+1} - \mathbf{M} u_i^*. \tag{4.20}$$

Comparing this with Equation 4.7 it becomes clear that they are the same with $\lambda = \left(p^{n+\frac{1}{2}} - p^*\right)\Delta t$ and a factor of ρ included in the mass matrices (i.e. $\mathbf{M}$ not $\mathbf{M_u}$). Using the same strategy as in Equation 4.13, Equation 4.20 may be transformed into an equation for the pressure correction over the timestep, $\left(p^{n+\frac{1}{2}} - p^*\right)\Delta t$:

$$\mathbf{C}_i^T \mathbf{M}^{-1} \mathbf{C}_i \left(p^{n+\frac{1}{2}} - p^*\right) \Delta t = \mathbf{d}_\partial - \mathbf{C}_i^T u_i^*. \tag{4.21}$$

Thus, the error in the pressure between the best estimate used in Equation 4.18, p^*, and the actual pressure for that timestep, $p^{n+\frac{1}{2}}$, acts as a Lagrange multiplier to project the solution of the momentum equation, u_i^*, into a divergence free and continuity satisfying space, u_i^{n+1}. This solution can be recovered as before by solving:

$$\mathbf{M}u_i^{n+1} = \mathbf{M}u_i^* + \mathbf{C}_i \left(p^{n+\frac{1}{2}} - p^* \right) \Delta t, \tag{4.22}$$

while the equivalent pressure is simply, $p^{n+\frac{1}{2}}$.

In summary, the solution procedure for the combined momentum and continuity equations is as follows (Chorin, 1968):

1. using the best current guess at pressure, p_i^*, solve Equation 4.18 for u_i^*;
2. solve Equation 4.21 for the new pressure, $p^{n+\frac{1}{2}}$;
3. use the correction in the pressure to ensure the new velocity, u_i^{n+1}, satisfies the conservation of mass using Equation 4.22.

This clearly parallels a Helmholtz decomposition, projecting the initial guess at velocity into a divergence free space. In this sense the pressure acts as a Lagrange multiplier to ensure that continuity is satisfied. However, at the same time the gradient of the pressure is a physically meaningful quantity, influencing the flow around the domain.

The semi-implicit procedure ensures that (depending upon the value of θ chosen) high Courant number flows may be considered (Gresho *et al.*, 1998). Furthermore, if steps 1–3 above are repeated twice per timestep, with $\theta = 1/2$ and $\theta_i = 1/2$ and updating the nonlinear terms in the momentum equation after the first iteration, then it is second-order accurate in time (Heywood & Rannacher, 1982, 1990; Gresho, 1990; E & Liu, 1996; Shen, 1996).

One obstacle to using the method described above is the projection matrix itself, $\mathbf{C}_i^T \mathbf{M}^{-1} \mathbf{C}_i$. For constant $\rho = 1$ this is identical to $\mathbf{C}_i^T {\mathbf{M_u}}^{-1} \mathbf{C}_i$ and is the discrete equivalent of the Laplace operator acting on λ in Equation 4.4. The pressure is therefore found by solving an elliptic equation (4.21), which requires appropriate boundary conditions to be well-posed. However, the velocity and pressure form a coupled system and their boundary conditions are similarly linked.

Dirichlet boundary conditions have already been imposed on the velocity through the construction of the divergence operator, $\mathbf{C}^T = \mathbf{D}$, in Equation 4.11. These boundary terms were ignored when comparing the discrete gradient, $\mathbf{C}$, and divergence, $\mathbf{C}^T$, operators. However, it is interesting to consider their effect within $\mathbf{C}_i^T \mathbf{M}^{-1} \mathbf{C}_i$.

Assuming that $\mathbf{M}^{-1}\mathbf{C}_i p$ acts as a suitable gradient operator, giving a discrete approximation of $p_{,i}$, then pre-multiplying by $\mathbf{C}_i^T$ should have the effect of taking the divergence of that gradient, $p_{,ii}$ (as in a Laplace operator). However, if any Dirichlet velocity boundary conditions have been imposed, $\partial\Omega_{g_i} \neq \emptyset$, then $\mathbf{C}^T$ is not a complete divergence operator as it is missing some boundary integrals that have been incorporated into the right hand side term of Equation 4.12, $\mathbf{d}_\partial$ (Equation 4.11b). Using $\mathbf{C}^T$ on the discrete approximation of $p_{,i}$ these missing boundary integrals act as weakly enforced homogeneous Neumann boundary conditions, enforcing $p_{,i} n_i = 0$ on $\partial\Omega_{g_i}$. Hence, wherever

a Dirichlet velocity boundary condition is enforced a homogeneous Neumann pressure gradient boundary condition is also assumed.

In the case where Dirichlet boundary conditions for velocity are specified on all boundaries, $\bigcup_i \partial\Omega_{g_i} = \partial\Omega$, then none of the boundary integrals are left within $\mathbf{C}^T$. Instead they all occur within the right hand side term, $\mathbf{d}_\partial$. In this case homogeneous Neumann pressure gradient boundary conditions are enforced on all boundaries. The solution to Equation 4.21 is then only unique up to some additive constant because only gradients are specified at the boundaries. In other words, $\mathbf{C}_i^T\mathbf{M}^{-1}\mathbf{C}_i$ has a one-dimensional nullspace containing constant functions or $\mathbf{C}_i^T\mathbf{M}^{-1}\mathbf{C}_i c = 0$, where c is the set of constant vectors. In order to numerically solve Equation 4.21 it is necessary to remove this nullspace by 'lifting' all the entries in the projection matrix related to a reference node. The information regarding this node is then moved to the right hand side of Equation 4.21 and its value pinned to some fixed value. As only the gradient, and not the absolute value of the pressure, is physically meaningful in an incompressible simulation, the choice of node is arbitrary as is the value, though it is always taken as zero in this thesis.

In the case where a boundary is left open, with no specified Dirichlet velocity boundary conditions, then a boundary integral remains within $\mathbf{C}^T$. The effect of this can be seen by examining the gradient operator, $\mathbf{C}$ (from Equation 4.8a):

$$\mathbf{c}_{iab} = -\int_{\Omega_e} N_a M_{b,i} \mathrm{d}V. \tag{4.23}$$

In exactly the same manner as in the divergence operator (Equation 4.11), by integrating $\mathbf{c}_{iab}$ by parts:

$$\mathbf{c}_{iab} = \int_{\Omega_e} N_{a,i} M_b \mathrm{d}V - \int_{\partial\Omega_e \in \partial\Omega_{g_i}} N_a M_b n_i \mathrm{d}s \quad \Rightarrow \mathbf{C}_{iAB} \quad \text{gradient matrix,} \tag{4.24a}$$

Dirichlet boundary conditions may be imposed on the pressure where Dirichlet velocity boundary conditions are not being applied. Letting $p = s$ on $\partial\Omega \backslash \partial\Omega_{g_i} = \partial\Omega_h$, this condition can be incorporated into a new right hand side term, $\mathbf{c}_\partial$:

$$c_{\partial_a} = \int_{\partial\Omega_e \in \partial\Omega_h} N_a s n_i \mathrm{d}s \qquad \Rightarrow \mathbf{c}_{\partial A} \quad \text{gradient right hand side.} \tag{4.24b}$$

Hence the gradient of the pressure is now defined as:

$$\mathbf{C}_i p = \mathbf{c}_\partial \tag{4.25}$$

The 'missing' boundary integral from Equation 4.24a may be added and subtracted back into the equation:

$$\mathbf{c}_{iab} = \int_{\Omega_e} N_{a,i} M_b \mathrm{d}V - \int_{\partial\Omega_e \in \partial\Omega} N_a M_b n_i \mathrm{d}s + \int_{\partial\Omega_e \in \partial\Omega_h} N_a M_b n_i \mathrm{d}s, \tag{4.26}$$

which allows the first terms to be integrated by parts back to their original form, giving:

$$\mathbf{c}_{iab} = -\int_{\Omega_e} N_a M_{b,i} \mathrm{d}V + \int_{\partial\Omega_e \in \partial\Omega_h} N_a M_b n_i \mathrm{d}s \quad \Rightarrow \mathbf{C}_{iAB} \quad \text{gradient matrix,} \tag{4.27}$$

which is the transpose of the divergence operator, $\mathbf{C}^T$:

$$\mathbf{d}_{iab} = \mathbf{c}^T_{iab} = -\int_{\Omega_e} M_{a,i} N_b \mathrm{d}V + \int_{\partial\Omega_e \in \partial\Omega_h} M_a N_b n_i \mathrm{d}s \quad \Rightarrow \mathbf{C}^T_{iAB} \quad \text{divergence matrix.} \tag{4.28}$$

Hence, the boundary terms, which were previously ignored when comparing $\mathbf{C}$ and $\mathbf{C}^T$, do not invalidate the comparison. In fact they allow Dirichlet boundary conditions to be imposed on the pressure at open boundaries. As only the gradient of the pressure is physically relevant, s is taken to be zero throughout this project. Hence, when a boundary is left open the pressure is set to zero at that boundary. This means that $\mathbf{C}^T_i \mathbf{M}^{-1} \mathbf{C}_i$ should have a zero-dimensional nullspace and Equation 4.21 may be solved without the imposition of a pinned reference node.

In the above discussion the projection matrix, $\mathbf{C}^T_i \mathbf{M}^{-1} \mathbf{C}_i$, was described as the discrete equivalent of a Laplace operator. However, except for special cases, it is not the same as the direct finite element discretisation of the Laplace operator (e.g. like the stress term for continuous Galerkin with a constant viscosity of unity). Instead the gradient is first computed using $\mathbf{M}^{-1}\mathbf{C}_i$ before the divergence of this result is found by pre-multiplying it with $\mathbf{C}^T_i$. The first of these steps requires the inversion of a mass matrix.

Using a finite element discretisation, the limited overlapping support of the basis functions results in matrices, such as $\mathbf{M}$, being sparse, with most of the entries equal to zero. Numerical models exploit this sparsity by only storing the non-zero entries of the matrix, significantly reducing the memory requirements (Tewarson, 1973). Linear solvers, which repeatedly perform matrix vector multiplications, exploit the sparse storage to only perform those operations (multiplications and additions) where required (Trefethen & Bau, 1997). In other words, they do not multiply out the zero entries of the matrix, which reduces the computational cost of solving a linear equation.

In general, the inverse of a sparse matrix is not itself sparse. Hence because $\mathbf{C}^T_i \mathbf{M}^{-1} \mathbf{C}_i$ contains the inverse of a mass matrix, for an arbitrary discretisation, it is not a sparse matrix. This means it cannot be easily stored and Equation 4.21 cannot be solved directly using a standard linear solver.

For a generic system:

$$\mathbf{A}y = b, \tag{4.29}$$

most linear solvers find the residual:

$$r = b - \mathbf{A}y, \tag{4.30}$$

and perform operations to try to minimise it such that, after k iterations, $y_k \approx \mathbf{A}^{-1}b$. For the more complex system:

$$\mathbf{C}^T_i \mathbf{M}^{-1} \mathbf{C}_i y = b, \tag{4.31}$$

the residual consists of:

$$r = b - \mathbf{C}^T_i \mathbf{M}^{-1} \mathbf{C}_i y. \tag{4.32}$$

Rather than requiring $\mathbf{C}_i^T\mathbf{M}^{-1}\mathbf{C}_i$ to exist as a sparse matrix with which to compute the residual, its effect can instead by approximated by first solving:

$$\mathbf{M}y^* = \mathbf{C}_i y, \tag{4.33}$$

for y^* using a linear solver. Then the residual of the original system may be found as:

$$r = b - \mathbf{C}_i^T y^*. \tag{4.34}$$

Hence, two linear solvers are required: an outer one for the global system, which aims to solve Equation 4.31, and an inner one, which at every iteration of the outer solve, solves Equation 4.33. While this approach allows Equation 4.21 to be solved by circumventing the lack of sparsity in the projection matrix, its use of inner and outer linear solvers is expensive.

An alternative approach involves the careful choice of velocity discretisation so that the inverse of the mass matrix can be found directly and is either a sparse matrix itself or can be approximated by a sparse matrix. In essence, this involves choosing the basis functions of velocity, N_A, carefully. This choice cannot be made in isolation as the stability of Equation 4.21 depends on the coupled discretisation of both velocity and pressure. In other words, the ability to find a solution to Equation 4.21 depends on the suitable choice of an element pair involving both N_A and M_A.

4.2 Stability

Stability is an essential property of any numerical discretisation. Instability can lead to spurious numerical artefacts contaminating the solution or a solution not being found at all. The analysis of the stability properties of the projection matrix, $\mathbf{C}_i^T\mathbf{M}^{-1}\mathbf{C}_i$, was performed by Ladyzhenskaya (1963), Babuška (1971) and Brezzi (1974) resulting in the concept of LBB stability through the so-called inf-sup condition. Given the finite element representations of any functions, υ^δ and ω_i^δ, from the pressure and velocity trial spaces respectively, the discrete inf-sup condition may be written as:

$$\min_{\upsilon^\delta \neq \text{constant}} \max_{\omega_i^\delta \neq 0} \frac{|\int_\Omega \upsilon_{,i}^\delta \omega_i^\delta \mathrm{d}V|}{||\omega_i^\delta||_{1,\Omega}||\upsilon^\delta||_{0,\Omega}} \geq C_{h_{\max}} > 0 \tag{4.35}$$

where $||\omega_i^\delta||_{1,\Omega} = \sqrt{\int_\Omega \omega_i^\delta\omega_i^\delta + \omega_{i,j}^\delta\omega_{i,j}^\delta \mathrm{d}V}$ is the H^1 norm of the velocity space and $||\upsilon^\delta||_{0,\Omega} = ||\upsilon^\delta - (1/|\Omega|)\int_\Omega \upsilon^\delta \mathrm{d}V||$ is the quotient space norm of the pressure space (Elman *et al.*, 2005). If there exists a positive constant, $C_{h_{\max}}$, which is bounded away from zero as the maximum edge length, $h_{\max}$, tends to zero, then an element pair is said to be LBB stable. In essence the inf-sup condition tries to ascertain whether the finite element representation of the gradient, evaluated in the velocity trial space, ω_i^δ, is bounded away from zero for all non-constant functions in the pressure trial space, υ^δ. This is the same as asking whether there are any pressure fields, other than a constant pressure, for which the gradient evaluated in the velocity space is zero. If there are then the pressure gradient may be incorrectly evaluated and the system will be unstable.

An LBB stable element pair has several important properties. As the edge lengths are decreased, Equation 4.21 will converge on the correct solution with an optimal order of accuracy: $\min(n+1, m+1)$, where n and m are the polynomial orders of the velocity, N_A, and pressure, M_A, basis functions respectively. Furthermore, the projection matrix, using an LBB stable element pair, will be free from eigenvalues which tend to zero as the edge length decreases. The corollary to this is that discretising $\mathbf{C}_i^T\mathbf{M}^{-1}\mathbf{C}_i$ using an LBB unstable element pair will in general result in functions, υ^δ, for which it is not possible to bound the gradient away from zero. These functions correspond to eigenvectors of the matrix associated with eigenvalues that tend to zero as $h_{\max} \to 0$. These eigenvectors inflate the nullspace of the projection matrix and make Equation 4.21 difficult, if not impossible, to solve. In essence, for an unstable discretisation the solution to Equation 4.21 is not unique and there exist multiple pressures which project the velocity into a divergence free space. These alternate pressure solutions are known as pressure modes.

The simplest choice of element pair, an equal-order interpolant for velocity and pressure, involves letting $M_A = N_A$. For linear continuous shape functions on a simplex mesh (Figure 2.1(a, c)) this is referred to as P1P1, where the first P1 refers to the discretisation of velocity and the second refers to the discretisation of pressure. P is used to represent a simplex mesh, while cube meshes, which are not considered here, use Q (Elman *et al.*, 2005).

A P1P1 discretisation has the advantage of being simple to implement. However, while the P1 mass matrix is sparse, its inverse is not. A non-sparse matrix cannot easily be stored therefore, it is not possible to explicitly assemble $\mathbf{C}_i^T\mathbf{M}^{-1}\mathbf{C}_i$. As discussed previously the lumped P1 mass matrix, $\mathbf{M}^L$:

$$\mathbf{M}^L_{ii} = \sum_j \mathbf{M}_{ij}, \tag{4.36a}$$

$$\mathbf{M}^L_{ij} = 0, \forall i \neq j, \tag{4.36b}$$

shares the conservation properties of the full P1 mass matrix. Additionally, it is diagonal so its inverse is also diagonal and can be found directly. Hence, the projection matrix in Equation 4.21 may be approximated by $\mathbf{C}_i^T\mathbf{M}^{L^{-1}}\mathbf{C}_i$, which can be assembled explicitly. For this to be consistent with its derivation (Equation 4.20) the mass must also be lumped in the momentum equation:

$$\mathbf{M}^L\frac{u_i^* - u_i^n}{\Delta t} + \theta\mathbf{N}u_i^* + \theta\mathbf{K}u_i^* = -(1-\theta)\,\mathbf{N}u_i^n - (1-\theta)\,\mathbf{K}u_i^n + \mathbf{C}_i p^* + \mathbf{f}_i + \mathbf{b}_i, \tag{4.37}$$

and in the velocity correction:

$$\mathbf{M}^L u_i^{n+1} = \mathbf{M}^L u_i^* + \mathbf{C}_i\left(p^{n+\frac{1}{2}} - p^*\right)\Delta t. \tag{4.38}$$

This leads to the $\mathrm{P1_L P1}$ element pair, where L indicates the mass matrix has been lumped.

Despite its ease of implementation and, through lumping, the ability to explicitly form the projection matrix, equal order element pairs are always LBB unstable. In fact a necessary, though not sufficient, condition for stability is that the velocity has more

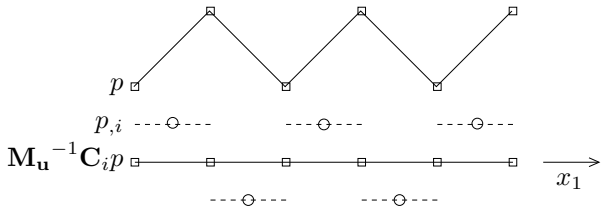


Figure 4.1: One-dimensional example of the pressure gradient operator nullspace. The pressure, p, is piecewise linear and stored at the nodes (squares). The pressure gradient, $p_{,i}$ (dashed lines), is piecewise constant and represented exactly at the cell centres (circles). The discrete pressure gradient operator, $\mathbf{M_u}^{-1}\mathbf{C}_i p$, using P1P1, averages the pressure gradients back to the nodes (squares), resulting in the evaluation of the gradient as zero.

degrees of freedom than the pressure (Gresho *et al.*, 1998; Elman *et al.*, 2005; Cotter *et al.*, 2009a). This can be seen intuitively by considering again the successive operations in the projection matrix: first taking the gradient then taking the divergence of that gradient. Consider the case where pressure is a piecewise linear continuous function, P1. Then its gradient is piecewise constant in each element, P0. The projection matrix evaluates the gradient of the pressure using the same test space as the velocity. The resulting discrete gradient, $\mathbf{M}^{-1}\mathbf{C}p$, lies within the velocity space. Using the same velocity and pressure space the operator $\mathbf{M}^{-1}\mathbf{C}$ in some sense averages the element based gradient information back onto the nodes. This process is illustrated on a one-dimensional structured mesh in Figure 4.1. A piecewise linear pressure, p, is represented at the element vertices (squares). Its gradient, $p_{,i}$, is exactly evaluated as a piecewise constant function at the element centres (circles). However, averaging the cell centred data back onto the nodes results in the discrete gradient, $\mathbf{M}^{-1}\mathbf{C}p$, being evaluated as zero everywhere. As the pressure was not constant the inf-sup condition is clearly violated. Considering the reverse problem (which is what the pressure equation has to solve) there are multiple such zig-zag solutions which could produce a discrete gradient of zero. The pressure equation therefore cannot be solved as a unique solution does not exist.

In multiple dimensions and on any mesh, the instability using equal order interpolants can also be thought of as a loss of information. Again considering a piecewise linear pressure its gradient lies at the element centres. As there are more elements than vertices in a simplex mesh (above a certain small size) averaging the gradient back onto the nodes throws away some data. The discrete divergence operator, $\mathbf{C}^T$, then projects the discrete gradient back into the same space as the pressure. If information has been lost during the gradient evaluation, as with equal order interpolants, it cannot be regained when the divergence is taken. Hence the projection matrix does not provide a good approximation of the Laplace operator and is ill-conditioned.

Given the requirement that the velocity space has more degrees of freedom to accommodate the pressure gradient, a standard method of achieving LBB stability is to increase the number of degrees of freedom used for velocity (Elman *et al.*, 2005). In

standard structured finite difference and finite volume numerical models this has led to the use of staggered grids (Arakawa & Lamb, 1977) where the velocity and pressure are offset relative to one another and represented at different points on the mesh so that there are more degrees of freedom for velocity than pressure. For instance, many structured multi-material models store pressure at the cell centres, in the same location as the volume fractions, and velocities either at the vertices of the cells (a so-called B grid, Arakawa & Lamb, 1977) or with different components on each of the orthogonal cell faces (a so-called C grid, Arakawa & Lamb, 1977).

Using finite elements extra velocity nodes can be introduced by changing the interpolating function. For example, in combination with a P1 pressure, a P1 velocity field could either have an extra degrees of freedom added to each element, leading to a so-called 'bubble' function (Crouzeix & Raviart, 1973; Arnold *et al.*, 1984), or the order of the velocity basis functions could simply be raised to quadratic (P2), introducing a new node on every edge. In this thesis only changing the order and continuity of the basis functions was considered. This was done for simplicity of implementation as the resulting interpolants are always Lagrange elements (only differentiable once, C^1, at the vertices in the continuous case), which generalise easily between different dimensions.

The simplest extension to a P1P1 element pair is P2P1, where velocity is represented on piecewise quadratic continuous basis functions (P2, Figure 2.1(b, d)) and pressure is interpolated using piecewise linear continuous basis functions (P1, Figure 2.1(a, c)). P2P1 is an LBB stable element pair (Hughes, 2000) that has been widely used in finite element models of fluid dynamics (Taylor & Hood, 1973; Gresho *et al.*, 1998). As the velocity is piecewise quadratic, the basis functions, N_A, are quadratic interpolants and the resulting mass matrix, $\mathbf{M}$, does not have a sparse inverse. Therefore, as with P1P1, it is not possible to form the projection matrix explicitly.

Unlike with P1P1, it is not possible to lump the mass matrix using Equation 4.36 in a P2P1 discretisation as the velocity basis functions are not strictly positive. The resulting lumped matrix contains zero entries along the diagonal (Zienkiewicz, 1977), which make the matrix impossible to invert. Several alternative methods have been suggested to lump mass matrices (Zienkiewicz, 1977) including scaling the diagonal entries appropriately to ensure conservation and numerical quadrature schemes that only sample at the nodes. By any method, the lumped mass matrix, $\mathbf{M}^L$, must preserve conservation, such that:

$$\mathbf{M}^L u_i = \mathbf{M} u_i. \tag{4.39}$$

It is suggested here that the mass matrix for the control volume mesh dual to the piecewise quadratic continuous finite element mesh (Figure 3.16(b, d)) provides a suitable diagonal mass matrix, $\mathbf{M}^L$, for use with P2P1 velocity-pressure discretisations. For two-dimensional elements this mass matrix may be formed by considering the decomposition of the quadratic triangles into four linear triangles, on which the standard lumped mass (by Equation 4.36) is formed. This is the same as each node in the quadratic element being assigned the area of its surrounding control volume contained within that element: 1/4 of the element area for the three edge based nodes, 1/12 of the element area for the three vertex based nodes. As these areas do not overlap the result is a diagonal matrix

Triangles		36	79	151	1,586	15,574
Vertices		24	48	87	820	7,890
P1P1/P1$_{\mathrm{L}}$P1	u_i	24	48	87	820	7,890
	p	24	48	87	820	7,890
P2P1/P2$_{\mathrm{L}}$P1	u_i	85	176	326	2,414	31,354
	p	24	48	87	820	7,890
P0P1	u_i	36	79	151	1,586	15,574
	p	24	48	87	820	7,890
P1$_{\mathrm{DG}}$P2	u_i	108	237	453	4,758	46,722
	p	85	176	326	2,414	31,354

Table 4.1: Number of degrees of freedom (or solution nodes) for velocity and pressure on various two-dimensional simplex meshes. Mesh data taken from Cotter *et al.* (2009a).

and as they sum to the area of the element they are conservative. Solving Equation 4.37 and approximating the projection matrix by $\mathbf{C}_i^T \mathbf{M}^{L^{-1}} \mathbf{C}_i$ gives rise to a P2$_{\mathrm{L}}$P1 element pair.

Another family of LBB stable element pairs is given by Pn$_{\mathrm{DG}}$Pn+1, where $n \in \mathbb{N}$ and the subscripted DG indicates that the velocity is discontinuous (Cotter *et al.*, 2009a,b). These achieve stability because the gradient of the pressure space, Pn+1, actually lies within the velocity space, Pn$_{\mathrm{DG}}$. In other words, if the pressure is piecewise linear continuous, P1, its gradient will be tested by and evaluated in the lowest order discontinuous space, P0 (where the subscripted DG is normally dropped for simplicity in the piecewise constant case). Therefore, the operator $\mathbf{M}^{-1}\mathbf{C}$ applies no averaging and no information is lost. This results in $\mathbf{C}_i^T {\mathbf{M}_{\mathbf{u}}}^{-1} \mathbf{C}_i \lambda$ being exactly equivalent to the standard finite element discretisation of the Laplace operator, $-\lambda_{,ii}$ (Cotter *et al.*, 2009a).

The Pn$_{\mathrm{DG}}$Pn+1 family of element pairs has the additional advantage that the mass matrix from the velocity space, Pn$_{\mathrm{DG}}$, is always directly invertible and sparse. This can be seen most easily with the P0P1 discretisation where the velocity basis functions, N_A (Figure 2.3), do not overlap and hence the mass matrix is diagonal. For higher orders, $n > 0$, the velocity basis functions only overlap within elements (Figure 2.2) and hence the mass matrix is block diagonal. Each block is simply the elemental mass matrix, m_{ab}. For P1$_{\mathrm{DG}}$P2 this is a 3×3 matrix in two dimensions and a 4×4 matrix in three dimensions. These can be individually inverted and $\mathbf{C}_i^T \mathbf{M}^{-1} \mathbf{C}_i$ can be assembled explicitly.

For simplicity only the P0P1 and P1$_{\mathrm{DG}}$P2 are considered further here. The P1$_{\mathrm{DG}}$P2 element pair is used for the modelling of geophysical flows due to its balance properties: for instance between the gradient of pressure and the buoyancy and Coriolis terms (Cotter *et al.*, 2009b). The P0P1 discretisation was originally suggested by Williams & Zienkiewicz (1981) and has been used extensively to model shallow water flows: for example in lagoon environments (Umgiesser *et al.*, 2004).

Tables 4.1 and 4.2 demonstrate that, unlike the P1P1 and P1$_{\mathrm{L}}$P1 element pairs, for all but the smallest numbers of elements the P2P1, P2$_{\mathrm{L}}$P1, P0P1 and P1$_{\mathrm{DG}}$P2 element

Tetrahedra		44	215	398	2,003	19,140
Vertices		26	80	130	488	3,690
Edges		93	227	633	2,792	24,165
P1P1/P1$_{\mathrm{L}}$P1	u_i	26	80	130	488	3,690
	p	26	80	130	488	3,690
P2P1/P2$_{\mathrm{L}}$P1	u_i	119	307	763	3,280	27,855
	p	26	80	130	488	3,690
P0P1	u_i	44	215	398	2,003	19,140
	p	26	80	130	488	3,690
P1$_{\mathrm{DG}}$P2	u_i	176	860	1,592	8,012	77,640
	p	119	307	763	3,280	27,855

Table 4.2: Number of degrees of freedom (or solution nodes) for velocity and pressure on various three-dimensional simplex meshes. Mesh data taken from Cotter *et al.* (2009a).

Mesh	h, Mesh Spacing
A	$\frac{\pi}{5}$
B	$\frac{\pi}{10}$
C	$\frac{\pi}{20}$
D	$\frac{\pi}{40}$
E	$\frac{\pi}{80}$

Table 4.3: Grid point spacings for meshes used in convergence analysis.

pairs satisfy the requirement that there are more degrees of freedom for velocity than for pressure. Due to their stability these discretisations were implemented[1] and tested. This testing is reviewed in the next section. Their suitability for multi-material flows is then discussed in Section 4.3.

4.2.1 Convergence Analysis

To test the implementation of these methods in Fluidity a convergence analysis was performed using the method of manufactured solutions (MMS, as in Section 3.3.1). In these test cases a periodic square domain, Ω, was defined as $x_1 \in [-\pi/2, 3\pi/2]$, $x_2 \in [-\pi/2, 3\pi/2]$. This was discretised at a series of successively finer mesh sizes, h, given in Table 4.3 (also see Figure 4.2(iii)).

A periodic, smooth and divergence free desired solution field, u_i^e, was created:

$$u_1^e = \sin(x_1)\cos(x_2), \tag{4.40a}$$

$$u_2^e = -\cos(x_1)\sin(x_2). \tag{4.40b}$$

[1] A pre-existing implementation of the P1$_{\mathrm{DG}}$P2 discretisation was used.

This was then inserted into a modified momentum equation (2.7a):

$$\bar{S}_i = \rho u^e_{i,t} + \rho \left(u^e_i u^e_j\right)_{,j} - \rho u^e_i u^e_{j,j} - \mu u^e_{i,jj} - \rho g_i \tag{4.41}$$

to find a modified source term, $\bar{S}_i$, which, using Equation 4.40, becomes (found using the symbolic maths package SAGE, Stein *et al.*, 2009):

$$\bar{S}_1 = \rho \cos(x_1) \sin(x_1) + 2\mu \sin(x_1) \cos(x_2) - \rho g_1 \tag{4.42}$$

$$\bar{S}_2 = \rho \cos(x_2) \sin(x_2) - 2\mu \cos(x_1) \sin(x_2) - \rho g_2 \tag{4.43}$$

In Equation 4.41 the gradient of the desired pressure, $p^e_{,i}$, and standard momentum source, S_i, have been incorporated into the modified source such that $\bar{S}_i = S_i - p^e_{,i}$. $\bar{S}_i$ therefore incorporates both the solenoidal, S_i, and irrotational, $p^e_{,i}$, contributions to the momentum forcing term. These can be separated analytically using Fourier series and a Helmholtz decomposition as in Equation 4.4 to give the desired pressure solution, p^e:

$$p^e = \frac{\rho \cos(2x_1)}{4} + \frac{\rho \cos(2x_2)}{4}, \tag{4.44}$$

and the true momentum source, S_i:

$$S_1 = \rho \left(\cos(x_1) \sin(x_1) - \frac{\sin(2x_1)}{2} \right) + 2\mu \sin(x_1) \cos(x_2) - \rho g_1, \tag{4.45a}$$

$$S_2 = \rho \left(\cos(x_2) \sin(x_2) - \frac{\sin(2x_2)}{2} \right) - 2\mu \cos(x_1) \sin(x_2) - \rho g_2. \tag{4.45b}$$

Using a density of one ($\rho = 1$), zero gravity ($g_i = 0$), a viscosity of 0.7 ($\mu = 0.7$) and the source from Equation 4.45, Equations 4.18, 4.21 and 4.22 may be solved using the element pairs described above. As in Section 3.3.1 the desired solutions, Equations 4.40 and 4.44, are temporally invariant. However, the initial velocity condition, u_{0i}, was set to zero everywhere along with the initial pressure guess, p^*_0. Therefore the numerical solutions will vary over time until a steady state is reached. The tolerance for the steady state was set as $\max\left(\max_i\left(||u^{n+1}_i - u^n_i||_\infty\right), ||p^{n+\frac{1}{2}} - p^{n-\frac{1}{2}}||_\infty\right) = 10^{-10}$. The temporal discretisation used $\theta = 1/2$ and $\theta_i = 1/2$ while the timestep, Δt, was set on each mesh to give a maximum Courant number of 0.1.

Once a steady state has been obtained on all meshes (Table 4.3) the convergence analysis may be performed. Given the error on two meshes, in the pressure field on meshes A and B for example, the order of convergence, c_p, is given by:

$$c_p \approx \log_r \left(\frac{E^p_A}{E^p_B} \right), \tag{4.46}$$

where r is the refinement ratio ($r = 2$ in this case). An LBB stable discretisation should give $c_p = \min(n+1, m+1)$, where n and m are the polynomial orders of the velocity, N_A, and pressure, M_A, basis functions respectively. The errors were evaluated in the L_2 norm, such that for any given mesh:

$$E^p = ||p - p^e||_2 = \sqrt{\int_\Omega |p - p^e|^2 dV}, \tag{4.47}$$

$$E^{u_i} = ||u_i - u^e_i||_2 = \sqrt{\int_\Omega |u_i - u^e_i|^2 dV}, \tag{4.48}$$

where p and u_i are the numerical solutions to the momentum and continuity equations on that mesh once a steady state has been attained, and p^e and u_i^e are representations of the exact solution on the same mesh.

The implementations of the LBB stable P0P1, $\mathrm{P1_{DG}P2}$, P2P1 and $\mathrm{P2}_L\mathrm{P1}$ element pairs were tested using this method. First however, Figure 4.2 demonstrates the unstable pressure solution using the $\mathrm{P1_LP1}$ element pair. Pressure modes can be seen in the numerical solution, p, (Figure 4.2(i)) on all the meshes shown (Figure 4.2(iii)). While the solution still bears some resemblance to the desired solution, p^e, (Figure 4.2(ii)) it is contaminated by oscillations which saturate the scale bar. On meshes A, B and C these modes cause the linear solver to fail to converge and a steady state was not reached.

No pressure modes are apparent in the steady state numerical pressure solutions, p, for the P0P1 (Figure 4.3(i)), $\mathrm{P1_{DG}P2}$ (Figure 4.5(i)), P2P1 (Figure 4.7(i)) and $\mathrm{P2_LP1}$ (Figure 4.9(i)) element pairs. Instead the gradient of the solution is seen to converge to the gradient of the desired solution, p^e, with no contaminating oscillations seen. The absolute values of the solutions are offset because of the arbitrary choice of the zero reference pressure node in the lower left corner of the domain.

Similarly, the steady state numerical velocity solutions, u_i, for the P0P1, $\mathrm{P1_{DG}P2}$, P2P1 and $\mathrm{P2_LP1}$ element pairs are shown in Figures 4.4(i, iii), 4.6(i, iii), 4.8(i, iii) and 4.10(i, iii) respectively. These solutions converge to the desired solutions, u_i^e, both in their patterns and their absolute values.

Tables 4.4 and 4.5 give the order of convergence for each of the LBB stable element pairs for pressure and velocity respectively. Similarly, Figure 4.11 demonstrates the rates in comparison to the expected orders of convergence graphically. All the element pairs tend towards a convergence rate with the expected order: P0P1 achieves first-order accuracy in both pressure and velocity, $\mathrm{P1_{DG}P2}$ is second-order accurate, and P2P1 and $\mathrm{P2_LP1}$ both initially achieve very high convergence in velocity before tending towards second-order, as in the pressure. This implies that the methods are both stable and implemented correctly. As with the advection convergence analysis in Section 3.3.1 these velocity and pressure convergence checks were submitted as regression tests. This ensures that subsequent changes to the code base are tested to ensure these results are unaffected.

Interestingly, the P2P1 and $\mathrm{P2_LP1}$ element pairs give identical convergence results to the number of decimal places shown. This indicates that the lumped mass matrix,

Meshes	P0P1	$\mathrm{P1_{DG}P2}$	P2P1	$\mathrm{P2_LP1}$
A→B	0.85	1.84	1.96	1.96
B→C	0.97	1.96	1.97	1.97
C→D	0.96	1.99	1.99	1.99
D→E	0.98	1.99	2.00	2.00

Table 4.4: Order of convergence, c_p, for pressure using the method of manufactured solutions on the momentum and continuity equations and various element pairs.

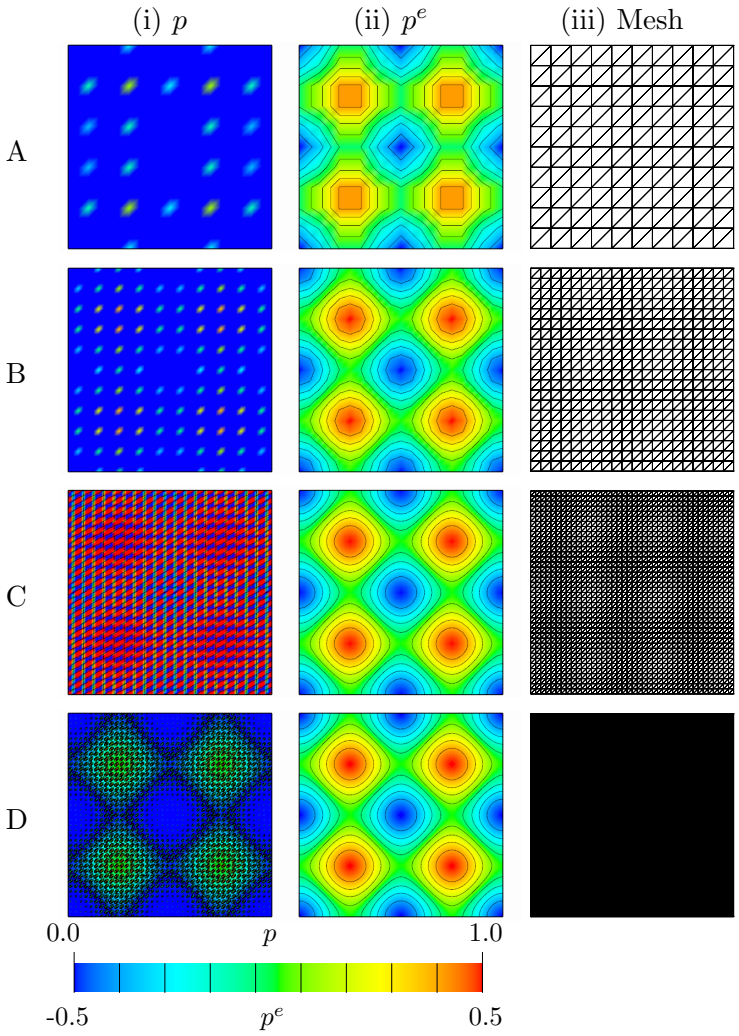


Figure 4.2: Numerical pressure solution, p, (i) to Equation 4.21 using the $\mathrm{P1_L P1}$ element pair and the method of manufactured solutions described in the text. The development of pressure modes is clearly visible when compared with the desired solution, p^e, (ii) on all the meshes shown (iii). Note that on meshes A, B and C the linear solver eventually failed to converge due to the pressure modes and the requested steady state tolerance was not reached. In these cases the final solution reached is shown here. Additionally the scale for the numerical solution is saturated. The discrepancy between the ranges of the numerical solution, p, and the desired solution, p^e, arises due to the arbitrary choice of the reference pressure node in the lower left corner of the mesh at $(-\pi/2, -\pi/2)$.

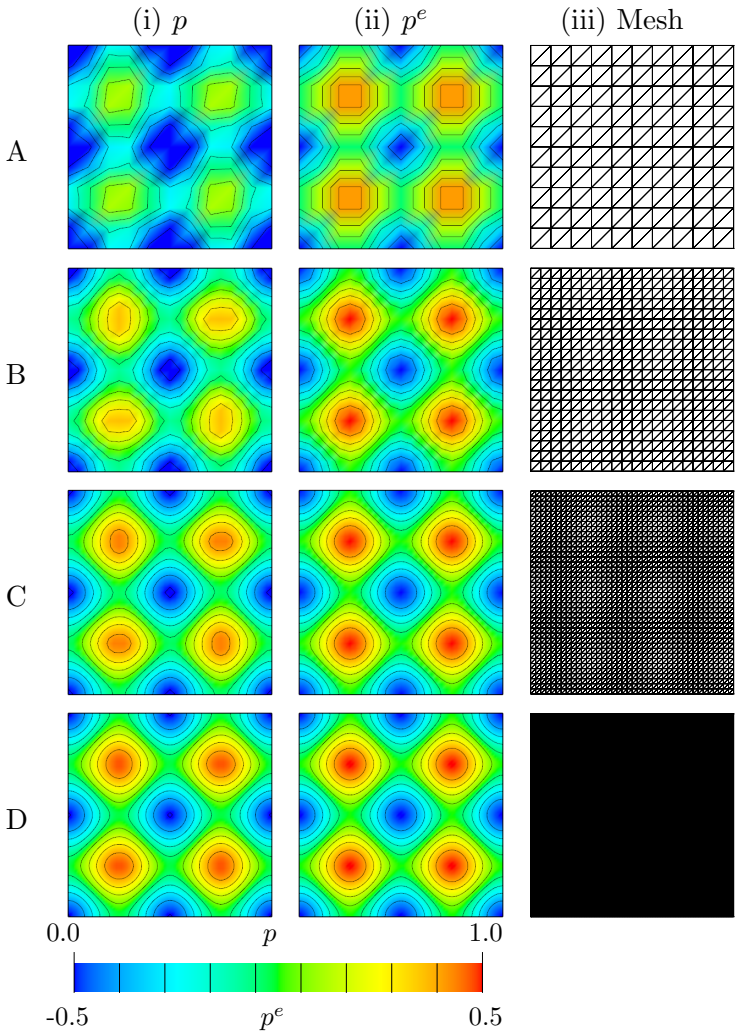


Figure 4.3: Numerical pressure solution, p, (i) to Equation 4.21 using the P0P1 element pair and the method of manufactured solutions described in the text. The gradients of the numerical solution are seen to converge on the gradients of the desired solution, p^e, (ii) with increasing refinement of the mesh (iii). The discrepancy between the ranges of the numerical solution, p, and the desired solution, p^e, arises due to the arbitrary choice of the reference pressure node in the lower left corner of the mesh at $(-\pi/2, -\pi/2)$.

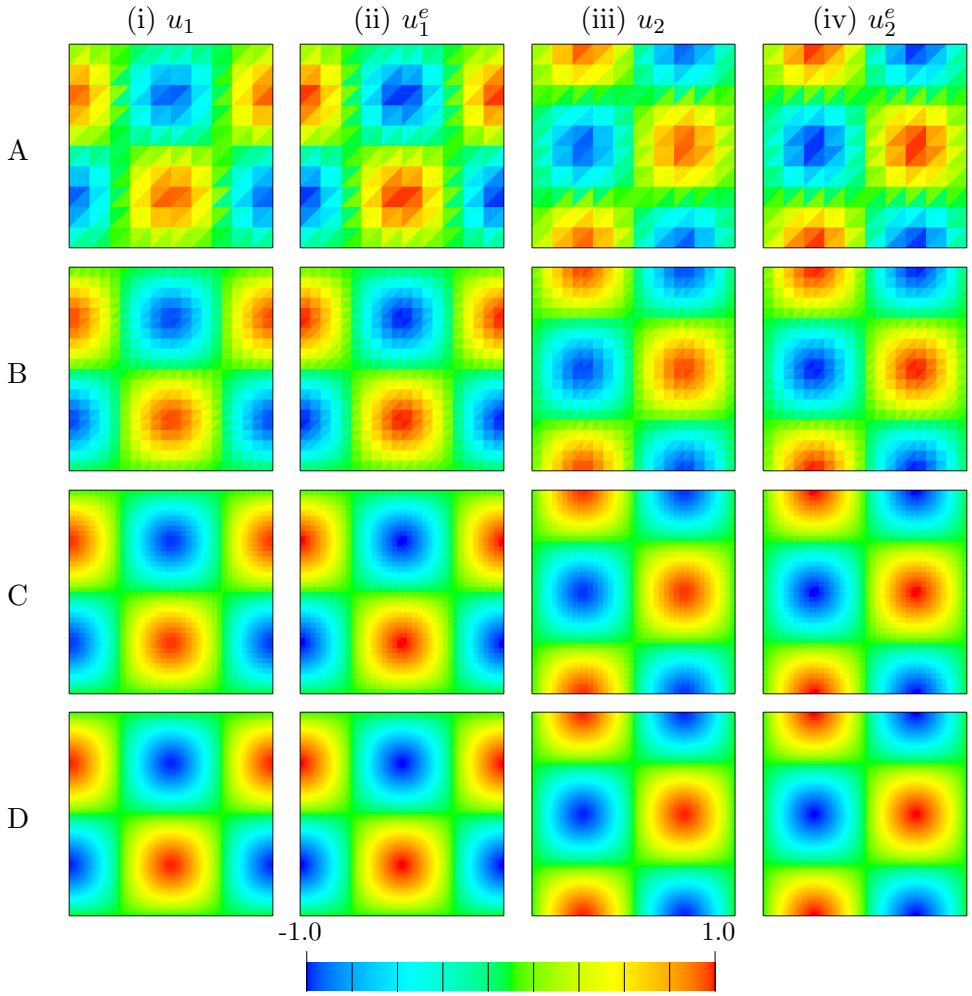


Figure 4.4: Numerical velocity solution, u_i, (i, iii) to Equations 4.18 and 4.22 using the P0P1 element pair and the method of manufactured solutions described in the text. The numerical solution is seen to converge on the desired solution, u_i^e, (ii, iv) with increasing refinement of the mesh (Figure 4.3(iii)).

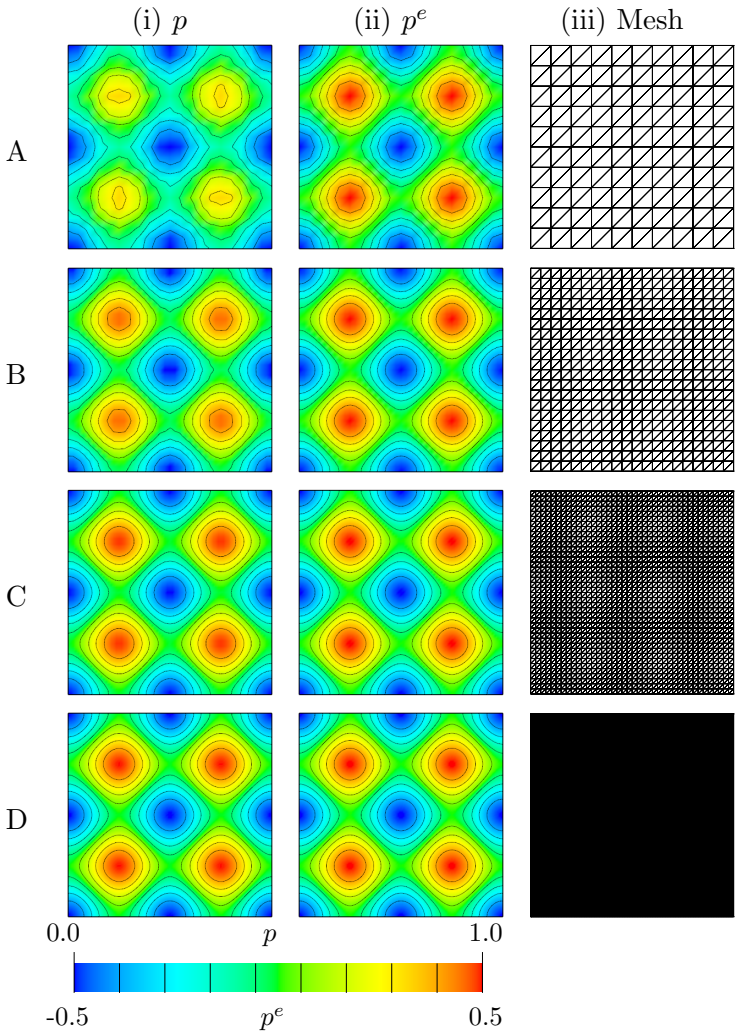


Figure 4.5: Numerical pressure solution, p, (i) to Equation 4.21 using the $\mathrm{P1_{DG}P2}$ element pair and the method of manufactured solutions described in the text. The gradients of the numerical solution are seen to converge on the gradients of the desired solution, p^e, (ii) with increasing refinement of the mesh (iii). The discrepancy between the ranges of the numerical solution, p, and the desired solution, p^e, arises due to the arbitrary choice of the reference pressure node in the lower left corner of the mesh at $(-\pi/2, -\pi/2)$.

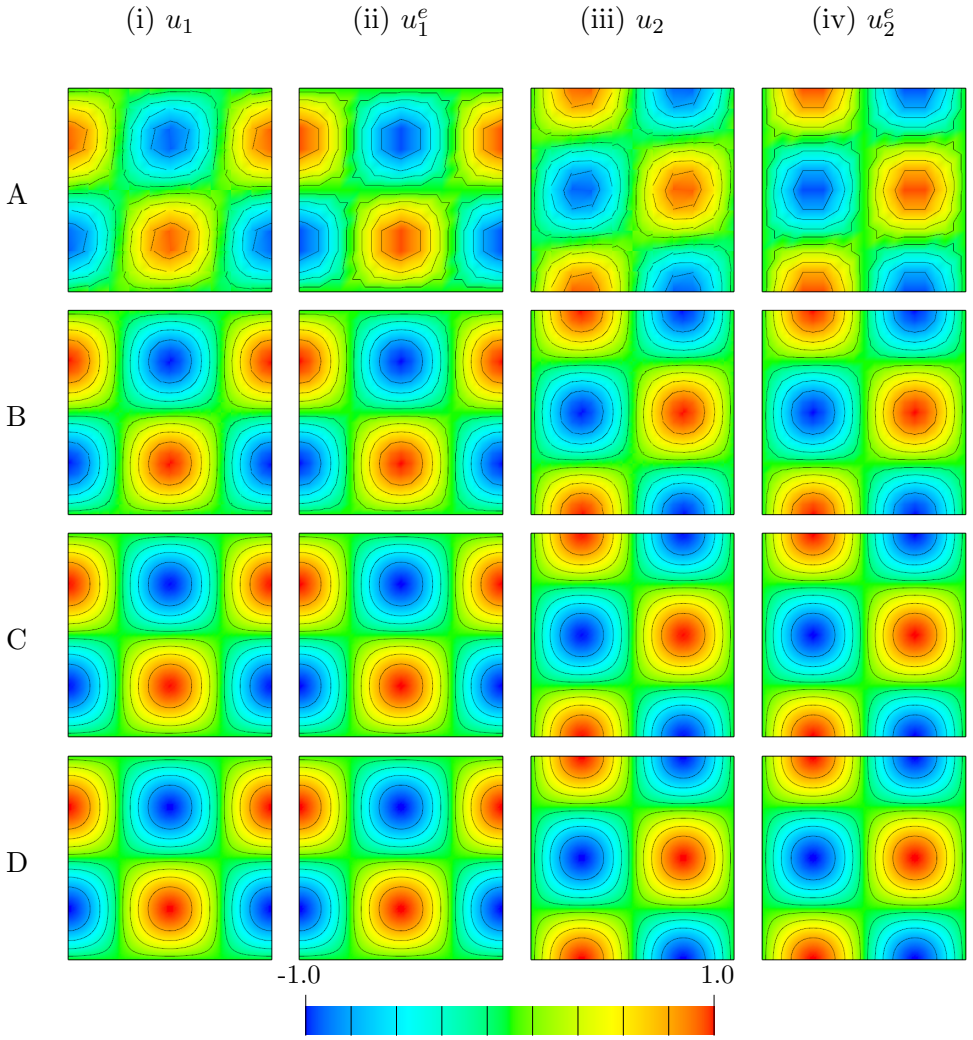


Figure 4.6: Numerical velocity solution, u_i, (i, iii) to Equations 4.18 and 4.22 using the $P1_{DG}P2$ element pair and the method of manufactured solutions described in the text. The numerical solution is seen to converge on the desired solution, u_i^e, (ii, iv) with increasing refinement of the mesh (Figure 4.5(iii)).

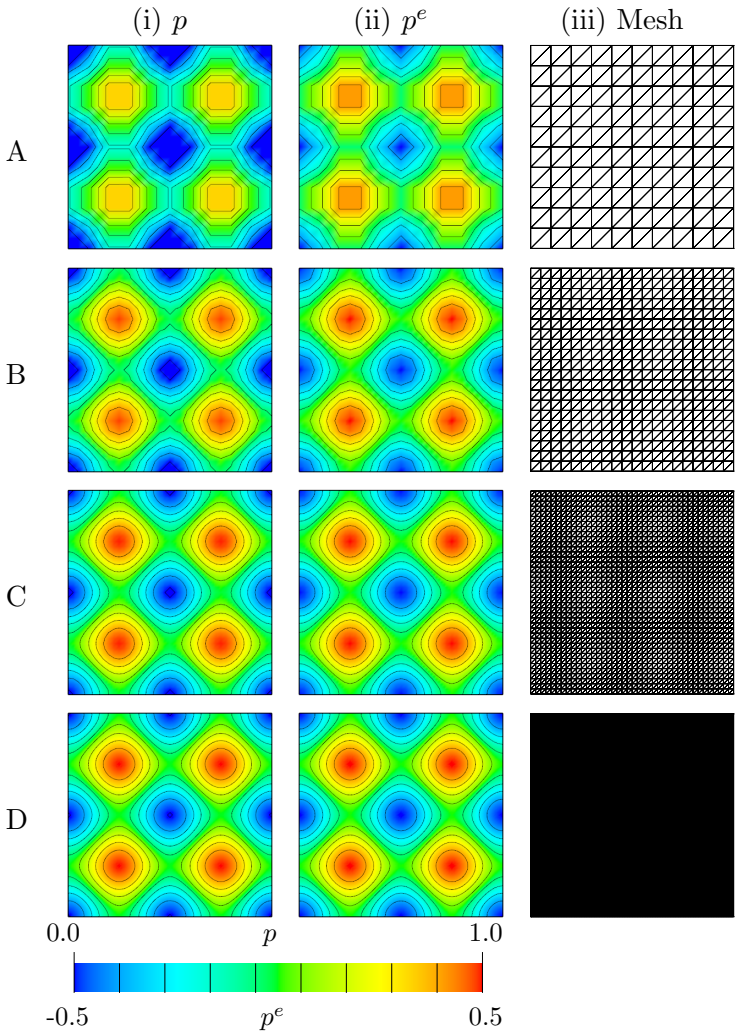


Figure 4.7: Numerical pressure solution, p, (i) to Equation 4.21 using the P2P1 element pair and the method of manufactured solutions described in the text. The gradients of the numerical solution are seen to converge on the gradients of the desired solution, p^e, (ii) with increasing refinement of the mesh (iii). The discrepancy between the ranges of the numerical solution, p, and the desired solution, p^e, arises due to the arbitrary choice of the reference pressure node in the lower left corner of the mesh at $(-\pi/2, -\pi/2)$.

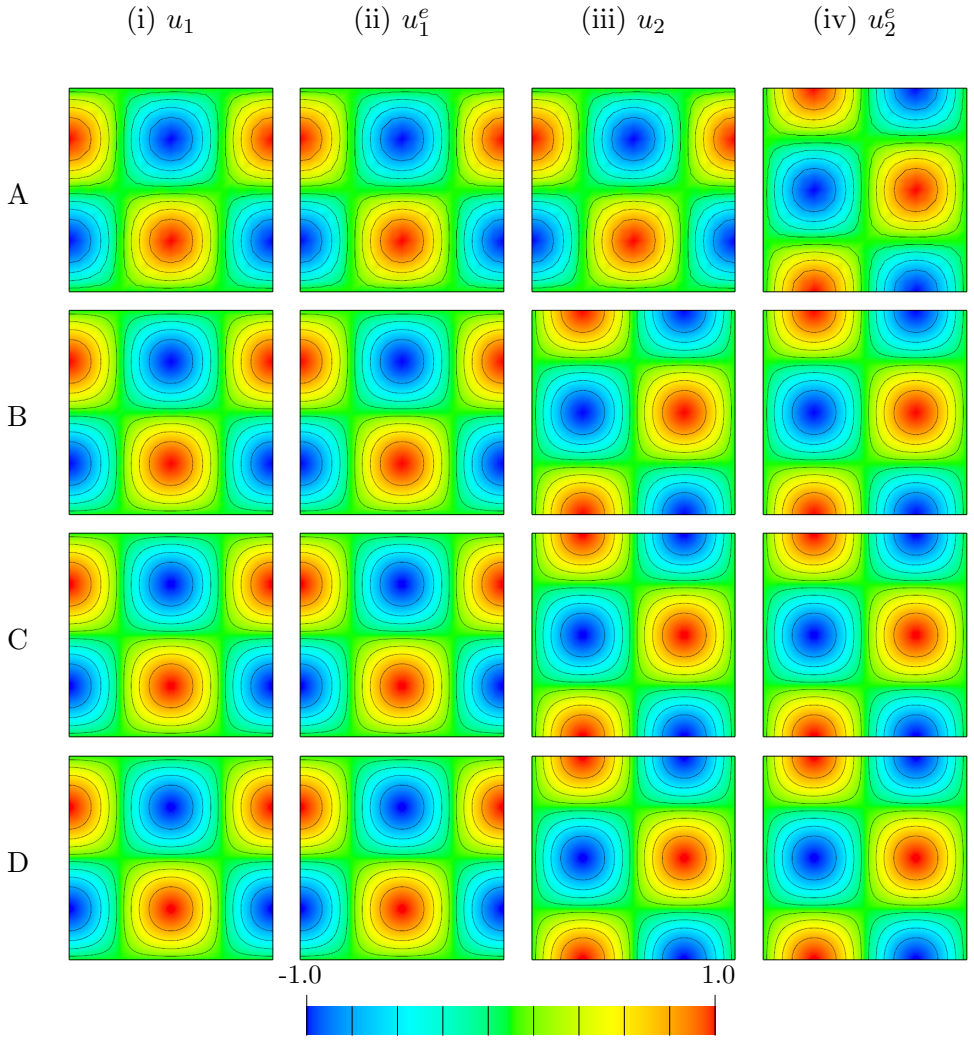


Figure 4.8: Numerical velocity solution, u_i, (i, iii) to Equations 4.18 and 4.22 using the P2P1 element pair and the method of manufactured solutions described in the text. The numerical solution is seen to converge on the desired solution, u_i^e, (ii, iv) with increasing refinement of the mesh (Figure 4.7(iii)).

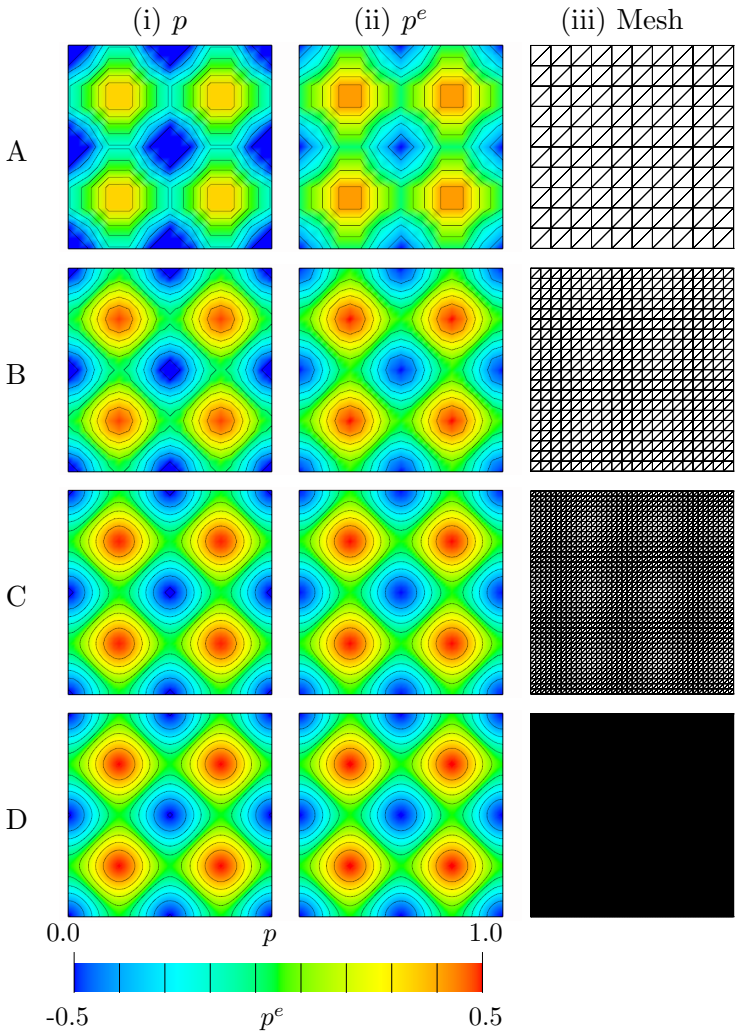


Figure 4.9: Numerical pressure solution, p, (i) to Equation 4.21 using the $\mathrm{P2_L P1}$ element pair and the method of manufactured solutions described in the text. The gradients of the numerical solution are seen to converge on the gradients of the desired solution, p^e, (ii) with increasing refinement of the mesh (iii). The discrepancy between the ranges of the numerical solution, p, and the desired solution, p^e, arises due to the arbitrary choice of the reference pressure node in the lower left corner of the mesh at $(-\pi/2, -\pi/2)$.

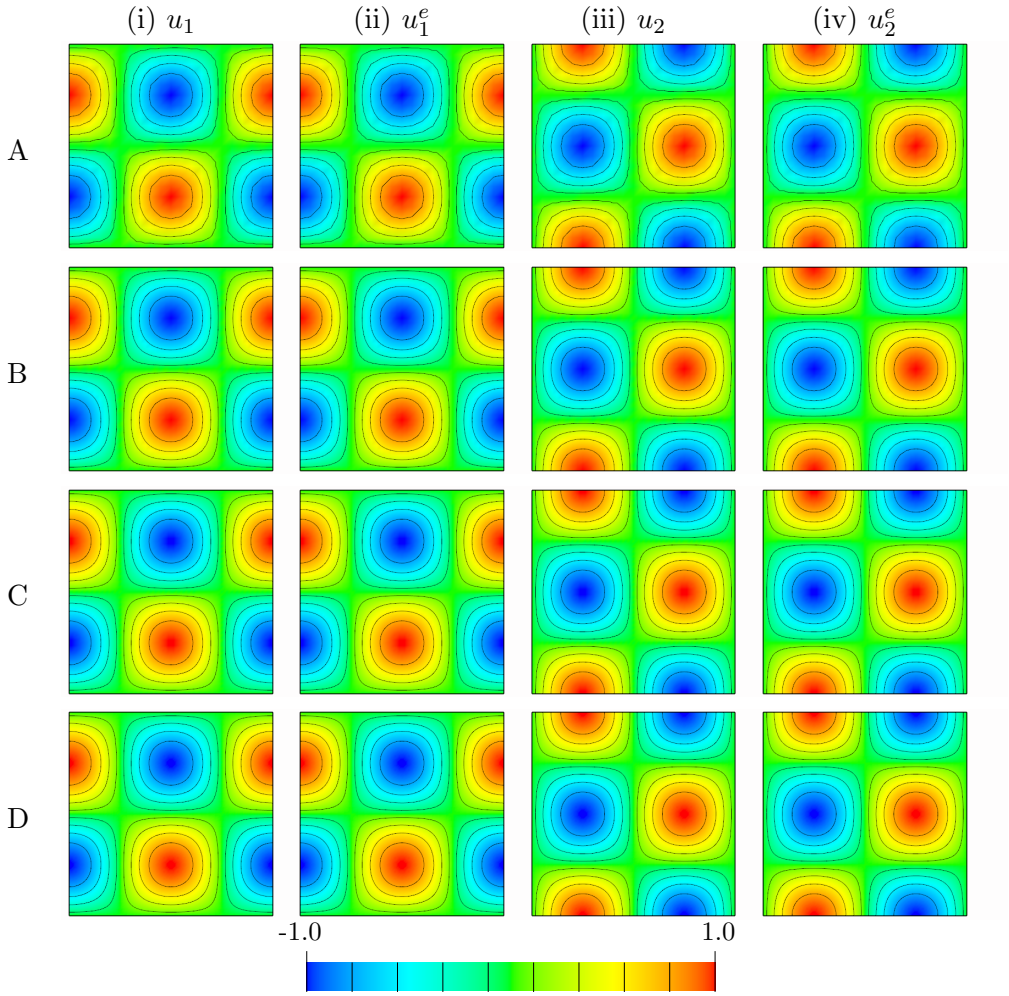
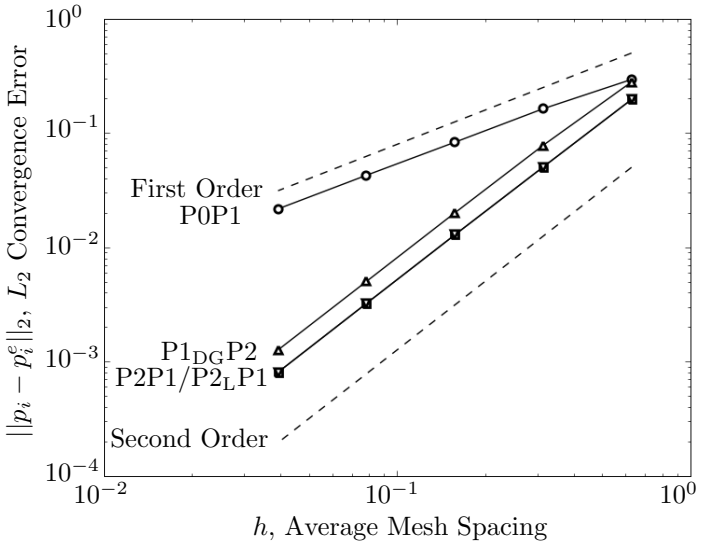


Figure 4.10: Numerical velocity solution, u_i, (i, iii) to the lumped Equations 4.18 and 4.22 using the $P2_LP1$ element pair and the method of manufactured solutions described in the text. The numerical solution is seen to converge on the desired solution, u_i^e, (ii, iv) with increasing refinement of the mesh (Figure 4.9(iii)).

(a) Pressure



(b) Velocity

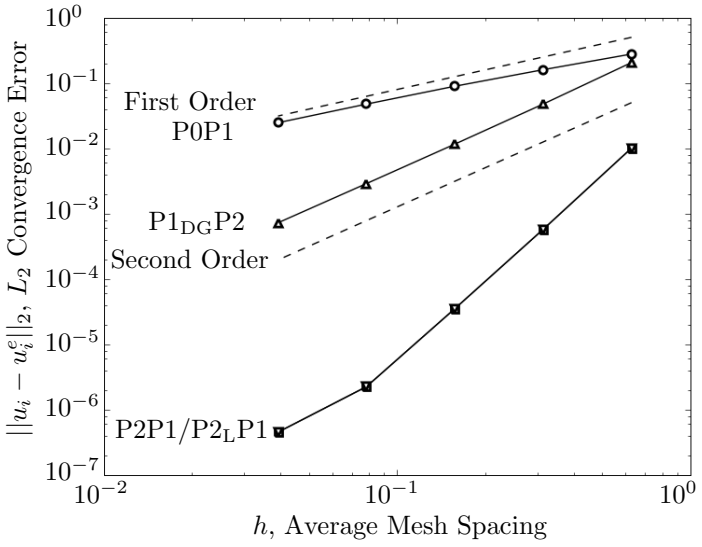


Figure 4.11: Plot of the error in the pressure, p, and velocity, u_i, solutions against the mesh spacing, h, for the momentum and continuity convergence tests.

	P0P1		$P1_{DG}P2$		P2P1		$P2_{L}P1$	
Meshes	u_1	u_2	u_1	u_2	u_1	u_2	u_1	u_2
A→B	0.80	0.80	2.11	2.11	4.13	4.13	4.13	4.13
B→C	0.84	0.84	2.04	2.04	4.04	4.04	4.04	4.04
C→D	0.91	0.91	2.01	2.01	3.97	3.97	3.97	3.97
D→E	0.95	0.95	1.99	1.99	2.30	2.30	2.31	2.31

Table 4.5: Order of convergence, c_p, for velocity using the method of manufactured solutions on the momentum and mass equations and various element pairs.

$\mathbf{M}^L$, taken from the dual control volume mesh is a suitable substitute for the full mass, $\mathbf{M}$, at least in terms of spatial convergence. Similarly, the convergence rates for each component of velocity are identical for all element pairs. This arises because the simplex meshes used were structured.

4.3 Material Conservation

Having demonstrated the successful implementation of several stable discretisations for the coupled velocity-pressure system it is now necessary to investigate which would be suitable for use in a multi-material model. Here, suitable is taken to mean that the velocity field produced by the discretisation can be used to advect a material volume fraction around the domain. The resulting coupled model must be stable, be able to maintain the bounds on the volume fraction, and conserve the mass of all the materials represented by the volume fraction. Stability is achieved through the suitable discretisation of the velocity-pressure system and by the selection of the timestep. Boundedness is a property of the choice of advection algorithm. Conservation however is a coupled problem between the velocity field and the volume fraction advection equation.

The advection equation has two forms, a conservative form:

$$\alpha_{,t} + (\tilde{u}_i \alpha)_{,i} = 0, \tag{4.49}$$

and a non-conservative form:

$$\alpha_{,t} + (\tilde{u}_i \alpha)_{,i} - \alpha \tilde{u}_{i,i} = 0. \tag{4.50}$$

The non-conservative or linear advection form results in bounded solutions (LeVeque, 2002). Because of the non-physicality of unbounded volume fractions this form was selected in Chapter 3. Furthermore, careful consideration was given to maintaining its bounded properties during its discretisation. However, for a generic non-divergence free velocity field, $\tilde{u}$, the volume fraction will not be conserved even though its bounds will always be maintained.

The only difference between Equations 4.49 and 4.50 is the velocity divergence term, $\alpha \tilde{u}_{i,i}$. Hence if the divergence of the velocity is zero then the field, α, will be advected

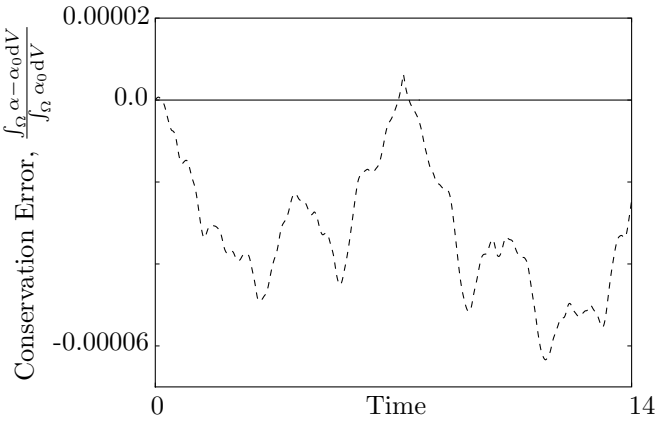


Figure 4.12: Conservation error using a discretely (in the control volume dual space) divergence free velocity (solid line, discrete divergence range = $[-8 \times 10^{-13}, 1 \times 10^{-12}]$) and an analytically divergence free velocity (dashed line, discrete divergence range = $[-0.01008572, 0.00827337]$) in the shearing flow test case (Figure 3.30).

preserving both the bounds and its integral. Physically this arises because divergence free flows cannot accumulate material locally. Hence, sufficient material is always transported away from a point to prevent the build up of material in its vicinity and the bounds are preserved.

All of the element pairs implemented for the velocity-pressure system achieve divergence free velocity fields. In fact the desired velocity field is identical to that used in the shearing test case (Section 3.3.2). Examining the integral of the field, $\int_\Omega \alpha \mathrm{d}V$, during the shearing flow (Figure 3.30) however, conservation errors can be seen (see dashed line in Figure 4.12).

The conservation errors seen in Figure 4.12 arise because the prescribed velocity field is analytically divergence free, $\tilde{u}_{i,i} = 0$, but not discretely divergence free. The velocity fields from the convergence analysis are discretely divergence free up to the preconditioned tolerance set on the linear solver (10^{-10}) used to find the pressure (Equation 4.21). This means that $\mathbf{M_p}^{-1}\mathbf{C}_i^T\tilde{u}_i \approx 0$, where $\mathbf{M_p}$ is the pressure mass matrix such that:

$$\mathbf{m_p}_{ab} = \int_{\Omega_e} M_a M_b \mathrm{d}V \qquad \Rightarrow \mathbf{M_p}_{AB} \qquad \text{pressure mass matrix.} \tag{4.51}$$

Therefore, in much the same way as the gradient of the pressure was tested and evaluated in the discrete velocity space using $\mathbf{M_u}^{-1}\mathbf{C}_i p$, the divergence of the velocity is tested and evaluated in the discrete pressure space using $\mathbf{M_p}^{-1}\mathbf{C}_i^T\tilde{u}_i$. This means that the definition of discrete divergence varies for each element pair and just as the analytically divergence free field was not conservative a general discretely divergence free field will not necessarily be either.

The correct definition of divergence will depend on the discretisation used for the linear advection equation (4.50). In Chapter 3 an explicit control volume discretisation was selected for the volume fraction and the discretised equation can be seen in Equation

3.53. The explicit velocity divergence term was given as:

$$\alpha\tilde{u}_{i,i} \approx \int_{\partial\Omega_{v_i}} \tilde{u}_i n_i \mathrm{d}s\, \alpha_A^n. \tag{4.52}$$

Hence, in order for the discrete equation (3.53) to be both conservative and bounded, the integral of the normal component of velocity around a control volume must be zero. This is equivalent to requiring that the pressure be represented on the control volume dual mesh such that the divergence matrix and right hand side are defined as:

$$\mathbf{c}_{iab}^T = \int_{\partial\Omega_v \ni \partial\Omega_{g_i}} M_a N_b n_i \mathrm{d}s \qquad \Rightarrow \mathbf{C}_{iAB}^T \qquad \text{divergence matrix,} \tag{4.53a}$$

$$d_{\partial_a} = -\int_{\partial\Omega_v \in \partial\Omega_{g_i}} M_a d_i n_i \mathrm{d}s \qquad \Rightarrow \mathbf{d}_{\partial A} \qquad \text{divergence right hand side.} \tag{4.53b}$$

As M_A is now a piecewise constant shape function within the control volumes (Figure 3.16) the body integral term is lost because $M_{A,i} = 0$. Furthermore because the pressure space is now discontinuous across the mesh the surface integrals must be evaluated on all control volume faces, $\partial\Omega_v$, not just those on the domain boundary, $(\bigcup_i \partial\Omega_{g_i}) \bigcup \partial\Omega_h = \partial\Omega$. The integrals described in Equations 4.53a and 4.53b may be evaluated on the control volume faces using the same elemental strategy discussed in Section 3.3.1. Using the definition of divergence evaluated on the dual control volume mesh in Equation 4.53, Equations 4.13c and 4.14 may be used to project the velocity field in the shearing flow test case to be discretely divergence free. This results in bounded advection with zero conservation error (see solid line in Figure 4.12).

None of the stable element pairs tested in the previous section (4.2.1) have a pressure space which lies in or spans the control volume mesh. In other words, the pressure basis functions, M_A, are not control volume shape functions (Figure 3.16) in any of the discretisations. However, as the control volume mesh is dual to the finite element mesh it is suggested here that the element pairs may be modified so that the pressure is discretised on the control volumes surrounding the original finite element nodes. This gives rise to the $\mathrm{P2P1_{CV}}$, $\mathrm{P2_LP1_{CV}}$, $\mathrm{P0P1_{CV}}$, $\mathrm{P1_{DG}P2_{CV}}$, $\mathrm{P1P1_{CV}}$ and $\mathrm{P1_LP1_{CV}}$ element pairs, where a CV subscript indicates the dual control volume mesh is being used for pressure.

The stability properties of the derived control volume pressure element pairs do not necessarily follow from their 'parent' discretisations. However, they all retain the same number of velocity and pressure degrees of freedom (see Tables 4.1 and 4.2). Hence, the $\mathrm{P1P1_{CV}}$ and $\mathrm{P1_LP1_{CV}}$ element pairs are unstable while the remaining discretisations satisfy one requirement for stability. However, having more velocity nodes than pressure nodes is only a necessary condition for LBB stability, which is concerned with preventing the development of pressure modes. Pressure modes occur when the pressure gradient is under-constrained by the space in which it is tested and evaluated, the velocity space. In other words there are multiple pressure solutions which achieve a divergence free velocity field. A similar instability is also possible in the velocity: velocity modes.

Velocity modes occur when the velocity divergence is under-constrained by the space in which it is tested and evaluated, the pressure space. In other words, there are multiple configurations of the velocity that give a discrete divergence of zero. Therefore, finding

a 'perfect' element pair is a balancing act between ensuring there are sufficient velocity degrees of freedom to prevent pressure modes but not so many that velocity modes occur. However, velocity modes are computationally less problematic than pressure modes. They do not affect the nullspace of the projection matrix and therefore do not prevent a solution to Equation 4.21 from being found. Therefore, ensuring LBB stability is generally considered more important in the design of momentum discretisations. Although computationally less problematic, under certain circumstances velocity modes do produce non-physical behaviour, which even LBB stable element pairs are not immune from. For example, in shallow water models with a Coriolis term, the P0P1 element pair has been shown to produce velocity modes (LeRoux *et al.*, 2007).

Modifying the element pairs from Section 4.2 to include a control volume based pressure space in effect lowers the constraints on the velocity divergence. For example, during the conversion of $\mathrm{P1_{DG}P2}$ to $\mathrm{P1_{DG}P2_{CV}}$, the element pair goes from having a piecewise quadratic continuous pressure space to having a piecewise constant discontinuous pressure space, although it has the same number of nodes in both. Hence, a simple test was performed on each of the modified element pairs to see whether velocity modes were excited even in highly simplified circumstances. A two-dimensional column of inviscid, $\mu = 0$, fluid with unit density, $\rho = 1$, was considered in the domain Ω: $x_1 \in [0, 0.5]$, $x_2 \in [0, 1]$. The bottom, $x_2 = 0$, left, $x_1 = 0$, and right, $x_1 = 0.5$, boundaries of the domain are closed with free slip boundary conditions, $d_i n_i = 0$, while the top is left open, $p = 0$. Initially at rest, $u_{0i} = 0$, with an initial pressure guess of zero, $p^* = 0$, it is subjected to gravity, $g = [0, -10]$, at the beginning of the simulation. At the first timestep this should result in a pressure field ranging from zero at the top of the domain to ten at the bottom. The gradient of this pressure should therefore balance the buoyancy force resulting in the fluid remaining at rest, $u_i = 0$. After the first timestep a steady state should be reached so the simulation was only run for two timesteps. The timestep size was selected arbitrarily (as the fluid is at rest) as $\Delta t = 0.005$.

The results after two timesteps of this simulation are shown in Figures 4.13–4.15 using the P0P1, $\mathrm{P0P1_{CV}}$, $\mathrm{P1_{DG}P2}$, $\mathrm{P1_{DG}P2_{CV}}$, P2P1, $\mathrm{P2P1_{CV}}$, $\mathrm{P2_LP1}$ $\mathrm{P2_LP1_{CV}}$, P1P1, $\mathrm{P1P1_{CV}}$, $\mathrm{P1_LP1}$ and $\mathrm{P1_LP1_{CV}}$ element pairs. All of the original element pairs, including the LBB unstable P1P1 and $\mathrm{P1_LP1}$ discretisations, produce the correct answer at the first timestep and reach a steady state by the second timestep.

Most of the modified element pairs fail to adequately represent the analytical solution. $\mathrm{P1_{DG}P2_{CV}}$ (Figure 4.13(d)) comes close to getting the correct pressure range but produces non-physical currents in the velocity field. Both $\mathrm{P2P1_{CV}}$ (Figure 4.14(b)) and $\mathrm{P2_LP1_{CV}}$ (Figure 4.14(d)) fail to get the pressure range correct and produce clear modes in both velocity components. Furthermore, these three element pairs fail to reach a steady state with the discrepancy between the analytical and numerical solutions increasing between the first and second timesteps.

$\mathrm{P1P1_{CV}}$ (Figure 4.15(b)) and $\mathrm{P1_LP1_{CV}}$ (Figure 4.15(d)) both approach the analytical solution with approximately the same pressure range and achieve a steady state. The velocities, while being almost zero, fail to remain stationary to the preconditioned tolerance used in the linear solvers (10^{-10}). While $\mathrm{P1P1_{CV}}$ and $\mathrm{P1_LP1_{CV}}$ are unstable,

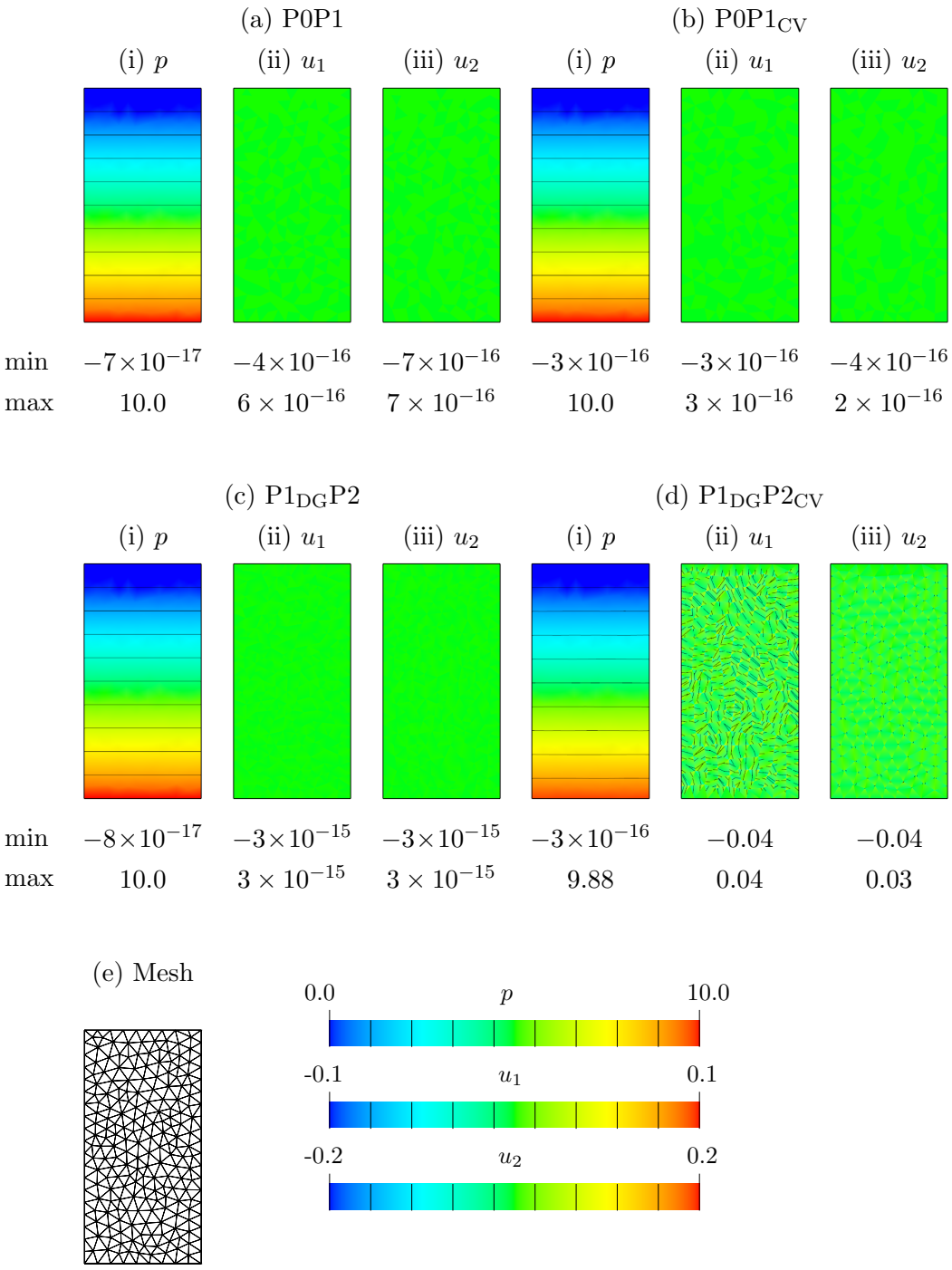


Figure 4.13: Column of unit length (Ω: $x_1 \in [0, 0.5]$, $x_2 \in [0, 1]$) containing an inviscid ($\mu = 0$), unit density ($\rho = 1$) fluid subjected to a gravity, $g = [0, -10]$, after two timesteps. The domain is tessellated using an unstructured mesh (e) and free slip boundary conditions, $d_i n_i = 0$, are imposed on the bottom, $x_2 = 0$, and sides, $x_1 = 0, 0.5$, of the domain. The top boundary is left open, $p = 0$. The analytical solution results in a pressure range of $[0, 10]$ and a velocity of 0. The numerical solutions and pressure, p, and velocity, u_i, ranges are shown using the (a) P0P1, (b) P0P1$_{\text{CV}}$, (c) P1$_{\text{DG}}$P2 and (d) P1$_{\text{DG}}$P2$_{\text{CV}}$ element pairs.

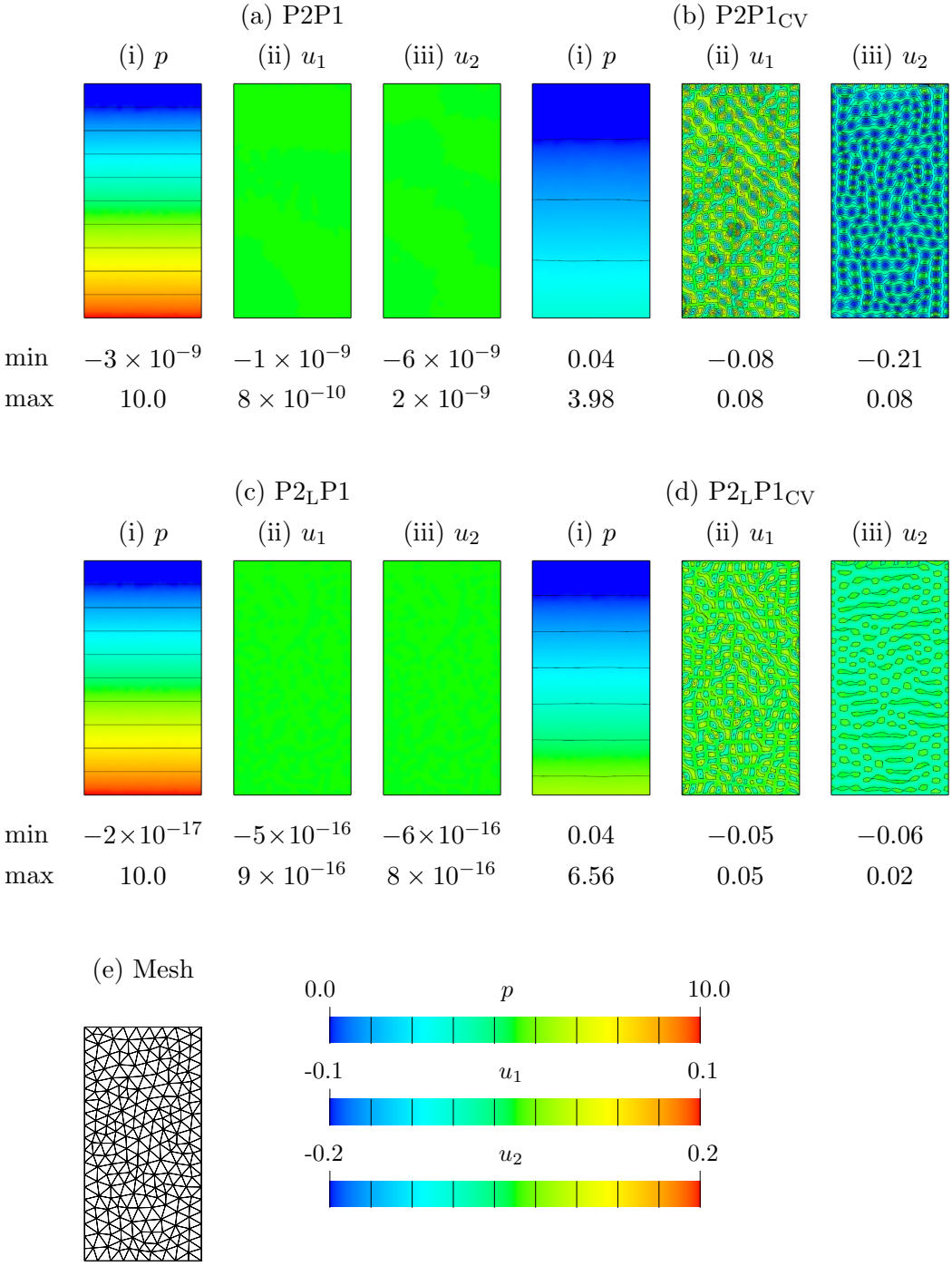


Figure 4.14: Column of unit length (Ω: $x_1 \in [0, 0.5]$, $x_2 \in [0, 1]$) containing an inviscid ($\mu = 0$), unit density ($\rho = 1$) fluid subjected to a gravity, $g = [0, -10]$, after two timesteps. The domain is tessellated using an unstructured mesh (e) and free slip boundary conditions, $d_i n_i = 0$, are imposed on the bottom, $x_2 = 0$, and sides, $x_1 = 0, 0.5$, of the domain. The top boundary is left open, $p = 0$. The analytical solution results in a pressure range of $[0, 10]$ and a velocity of 0. The numerical solutions and pressure, p, and velocity, u_i, ranges are shown using the (a) P2P1, (b) P2P1$_{\text{CV}}$, (c) P2$_{\text{L}}$P1 and (d) P2$_{\text{L}}$P1$_{\text{CV}}$ element pairs.

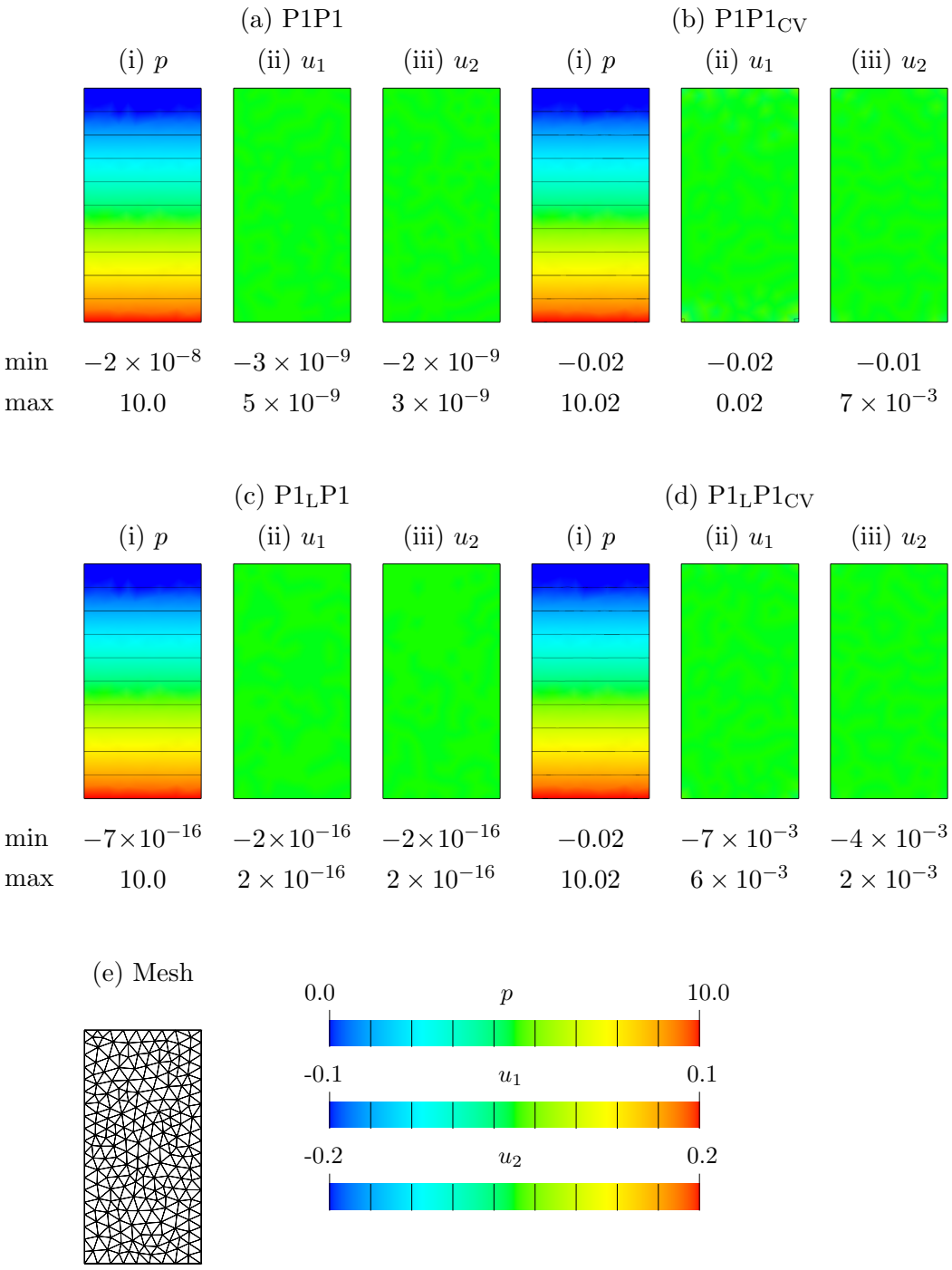


Figure 4.15: Column of unit length (Ω: $x_1 \in [0, 0.5]$, $x_2 \in [0, 1]$) containing an inviscid ($\mu = 0$), unit density ($\rho = 1$) fluid subjected to a gravity, $g = [0, -10]$, after two timesteps. The domain is tessellated using an unstructured mesh (e) and free slip boundary conditions, $d_i n_i = 0$, are imposed on the bottom, $x_2 = 0$, and sides, $x_1 = 0, 0.5$, of the domain. The top boundary is left open, $p = 0$. The analytical solution results in a pressure range of $[0, 10]$ and a velocity of 0. The numerical solutions and pressure, p, and velocity, u_i, ranges are shown using the (a) P1P1, (b) P1P1$_{\mathrm{CV}}$, (c) P1$_{\mathrm{L}}$P1 and (d) P1$_{\mathrm{L}}$P1$_{\mathrm{CV}}$ element pairs.

owing to their relative velocity and pressure node counts, Figure 4.15 shows that the instability does not always prevent a solution from being found (as in Figure 4.2). This is especially true on unstructured meshes where the non-alignment of the nodes disrupts the formation of 'checker-board' pressure modes. However, unstructured or not, as the mesh is refined the eigenvalues of the projection matrix tend towards zero, often leading to solver failure. This can be compensated for by stabilising the projection matrix. Several schemes exist to do this (Elman *et al.*, 2005) but they generally take the form of a stabilisation matrix, $\mathbf{K}$, added to Equation 4.21 such that (Piggott *et al.*, 2008):

$$\left(\mathbf{C}_i^T\mathbf{M}^{-1}\mathbf{C}_i + \mathbf{K}\right)\left(p^{n+\frac{1}{2}} - p^*\right)\Delta t = \mathbf{d}_\partial - \mathbf{C}_i^T u_i^* - \mathbf{K}p^*\Delta t. \tag{4.54}$$

The form of $\mathbf{K}$ varies depending on the scheme but is generally a smoothing operator, such as a diffusion matrix with a diffusivity dependent on the local edge lengths (Piggott *et al.*, 2008; Gresho *et al.*, 1998). This improves the condition number of the stabilised projection matrix and smooths any oscillations in the pressure solution. However, it also modifies the definition of discrete divergence such that Equation 4.12 becomes (Piggott *et al.*, 2008):

$$\mathbf{C}_i^T u_i + \Delta t\mathbf{K}p^{n+\frac{1}{2}} = \mathbf{d}_\partial. \tag{4.55}$$

While this does not affect global conservation, Equation 4.55 means that a stabilised $\mathrm{P1P1_{CV}}$ or $\mathrm{P1_LP1_{CV}}$ discretisation is unsuitable for material advection because the local divergence is no longer zero leading to conservation errors in the materials.

While all the other element pairs tested in the simple column test proved to be unsuitable for a multi-material model $\mathrm{P0P1_{CV}}$ (Figure 4.13(b)) successfully reproduced both the analytical pressure and velocity ranges and a steady state between the two timesteps. The result is also discretely divergence free using the control volume based definition of divergence in Equation 4.53. Proving the LBB stability of this element pair is non-trivial (Donéa & Huerta, 2003; Hughes, 2000) and beyond the scope of this project. Instead the convergence analysis from Section 4.2.1 is repeated to establish that first-order convergence is achieved as would be expected from a stable element pair.

Figures 4.16 and 4.17 present the results of the convergence analysis for pressure, p, and velocity, u_i, respectively. In both cases the numerical solutions can be seen to converge towards the analytical solution as the mesh is refined. As can be seen in Table 4.6 and graphically in Figure 4.18, this convergence tends toward the expected order, $c_p = 1$, with increasing refinement of the mesh for both pressure and velocity. This is taken to imply that the element pair is LBB stable. Furthermore, in the demonstration of the conservation error in the shearing flow test case this element pair was used to project the prescribed velocity to be discretely divergence free (see solid line in Figure 4.12), demonstrating that it also conserves the area of each material. Similarly, during the adaptive shearing flow test case (Figure 3.55) the prescribed velocity field was also projected to be divergence free after every step of mesh optimisation. This resulted in a discrete divergence in the range $[-8 \times 10^{-13}, 1 \times 10^{-12}]$ which, in combination with the conservative bounded interpolation discussed in Section 3.5, led to a conservation error of 10^{-13}.

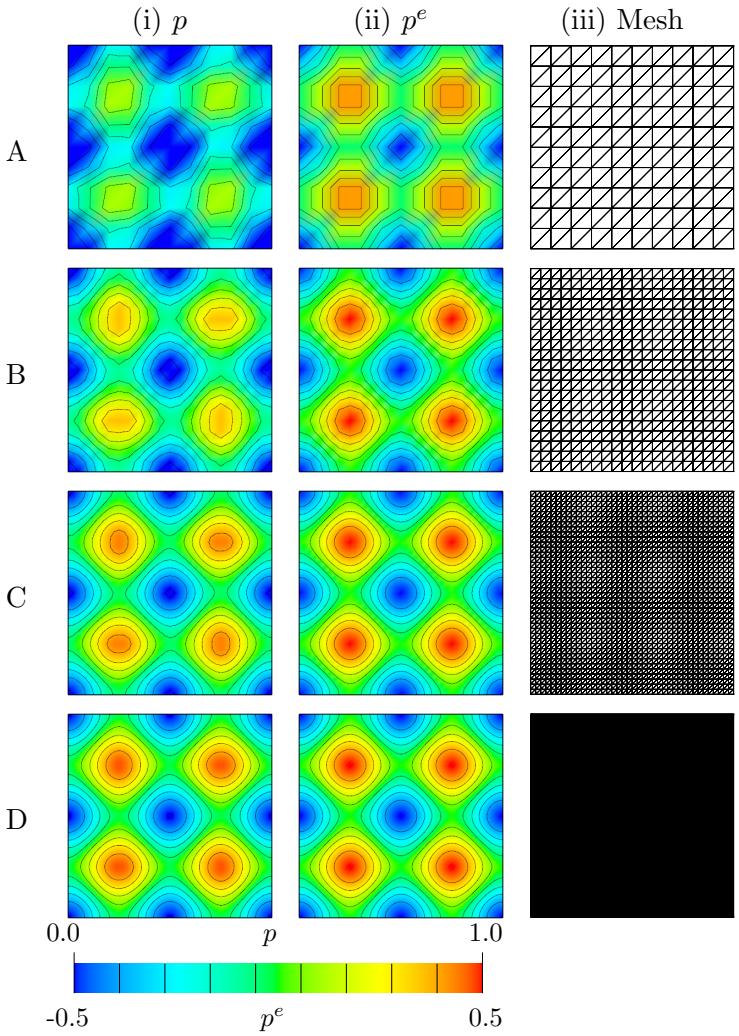


Figure 4.16: Numerical pressure solution, p, (i) to Equation 4.21 using the P0P1$_{\mathrm{CV}}$ element pair and the method of manufactured solutions described in Section 4.2.1. The gradients of the numerical solution are seen to converge on the gradients of the desired solution, p^e, (ii) with increasing refinement of the mesh (iii). The discrepancy between the ranges of the numerical solution, p, and the desired solution, p^e, arises due to the arbitrary choice of the reference pressure node in the lower left corner of the mesh at $(-\pi/2, -\pi/2)$.

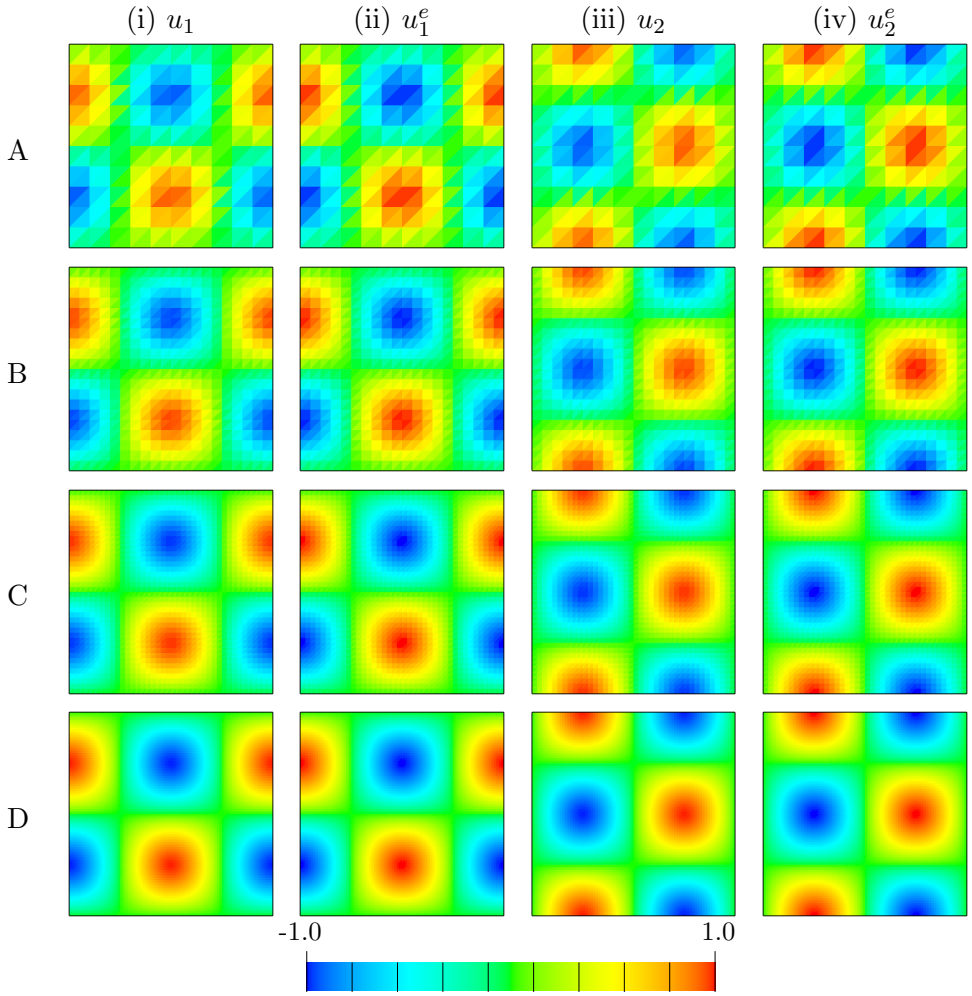
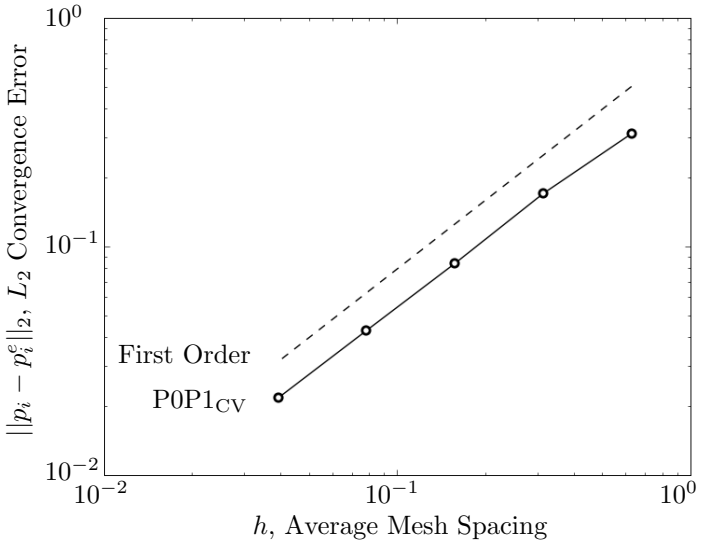


Figure 4.17: Numerical velocity solution, u_i, (i, iii) to Equations 4.18 and 4.22 using the P0P1$_{\mathrm{CV}}$ element pair and the method of manufactured solutions described in Section 4.2.1. The numerical solution is seen to converge on the desired solution, u_i^e, (ii, iv) with increasing refinement of the mesh (Figure 4.16(iii)).

(a) Pressure



(b) Velocity

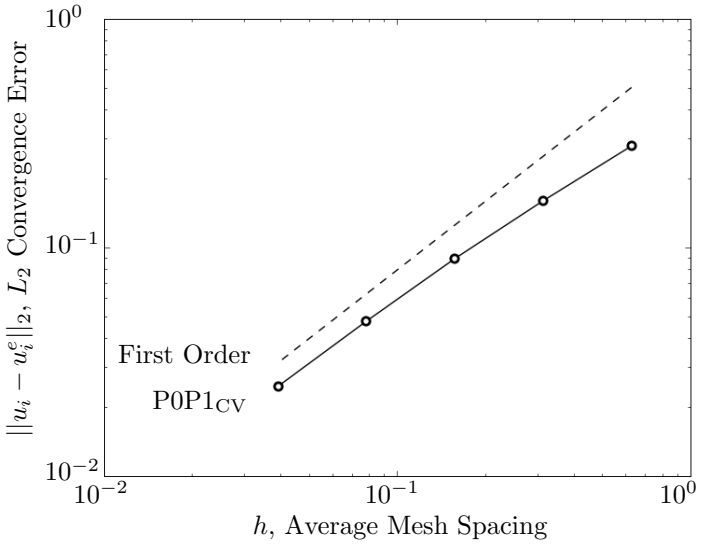


Figure 4.18: Plot of the error in the pressure, p, and velocity, u_i, solutions against the mesh spacing, h, for the momentum and mass convergence tests using the P0P1$_{\mathrm{CV}}$ element pair.

	P0P1$_{\mathrm{CV}}$		
Meshes	p	u_1	u_2
A→B	0.87	0.80	0.80
B→C	1.02	0.84	0.84
C→D	0.97	0.91	0.91
D→E	0.98	0.95	0.95

Table 4.6: Order of convergence, c_p, for pressure, p, and velocity, u_i, using the method of manufactured solutions on the momentum and mass equations and the P0P1$_{\mathrm{CV}}$ element pair.

The P0P1$_{\mathrm{CV}}$ discretisation is similar to a structured finite volume staggered B grid (Arakawa & Lamb, 1977). Generally, on structured staggered cube meshes the velocity is placed at the tessellation vertices while the pressure is considered to be cell or element centred. On an unstructured simplex mesh however, this is reversed due to the relative vertex to element count. Hence, with P0P1$_{\mathrm{CV}}$ the velocity is essentially a finite volume, cell centred variable while the pressure is a vertex based, control volume variable. The finite element parent element to P0P1$_{\mathrm{CV}}$, P0P1, was suggested as a basis for a multi-material model by Wan *et al.* (2009). However, instead of using the projection matrix used here, the Helmholtz projection (Equation 4.4) was discretised directly and the volume fractions were discretised using an element centred advection routine. These factors meant that the divergence was not evaluated in the correct discrete space and conservation of the materials was not achieved. Instead here, a conservative, bounded and stable coupled two material model has been designed through the careful discretisation of both the volume fraction and the velocity-pressure system.

In the fully coupled model, the advection schemes designed in Chapter 3 are used to advect a volume fraction around the domain according to the velocity field found from the solution to the momentum equation. This volume fraction is then used to update the global bulk density and viscosity according to Equations 3.1–3.2. These are then inserted into the momentum equation to update the velocity and pressure using the projection method. Typically, two nonlinear iterations are performed of this process updating the velocity in the advection equation and in the nonlinear terms of the momentum equation on the second iteration (see Figure 4.19). In the next section several examples will be given of the application of this model to experimental validation problems.

Figure 4.19: *See next page.* Simplified flow diagram showing the advection, momentum and mesh optimisation stages of a timestep and the corresponding nonlinear iterations and subcycled subtimesteps.

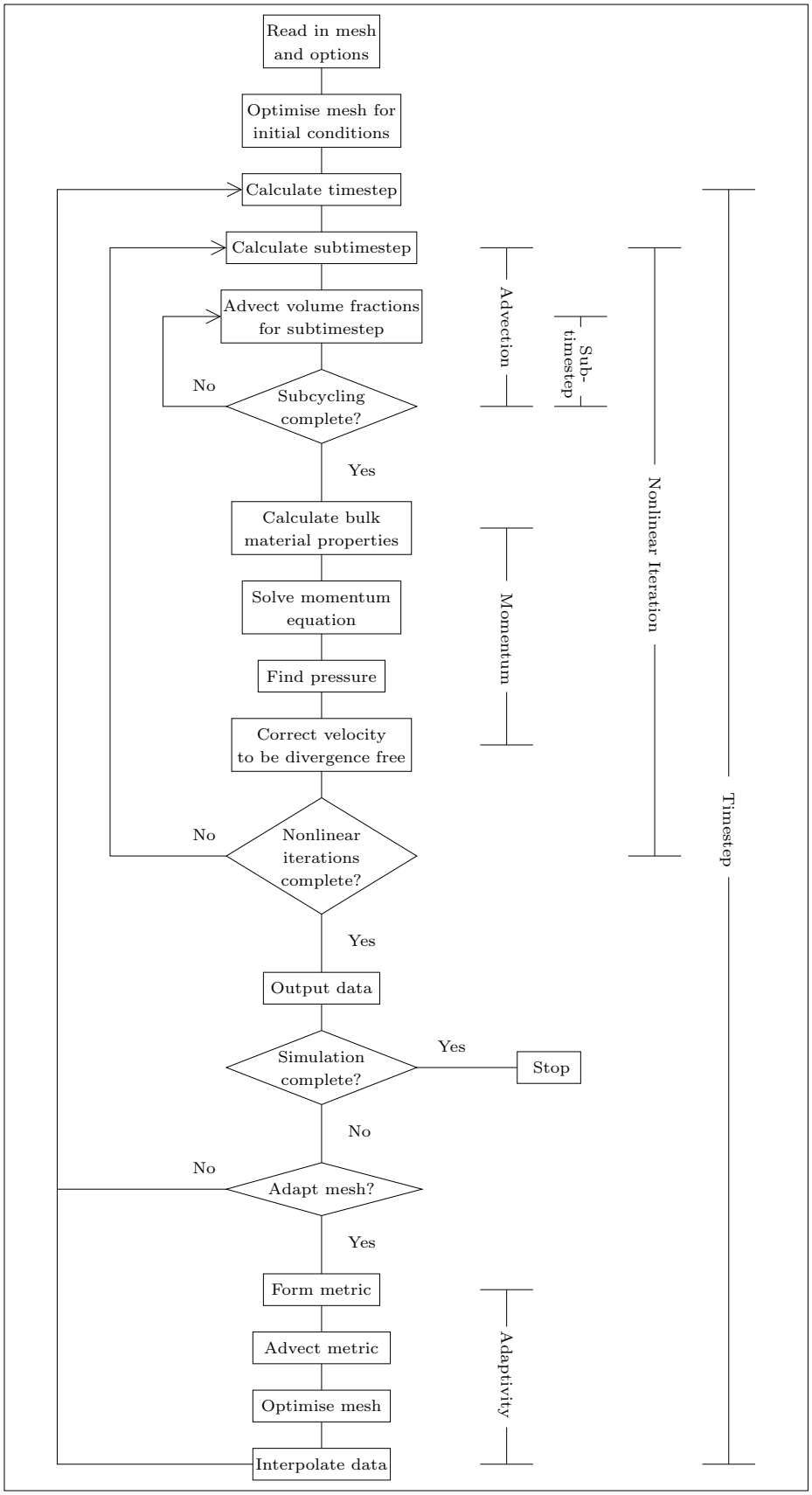


Figure 4.19: *cont.*

4.4 Experimental Validation

A commonly used validation experiment for multi-material models is that of a collapsing column of liquid, normally water, within an atmosphere or vacuum (Lakehal *et al.*, 2002), also known as the dam break problem. In the experimental set-up a reservoir of water is held behind an impermeable barrier separating it from the rest of the tank. The barrier is then quickly removed, allowing the water column to collapse and flood the remaining sections of the tank. To increase the complexity of the flow, fixed obstacles may be introduced to impede the flood. In the numerical analogue the initial condition is generally taken as the trapped water column, still behind the dam. At the start of the simulation the barrier is imagined to have been removed instantaneously and switching on gravity, $|g| = 9.81$, causes the column to collapse. Several experimental set-ups have been published and used as comparison and validation tools for numerical models (Martin & Moyce, 1952; Greaves, 2006). Those with water depth gauges distributed throughout the tank are particularly useful, allowing the direct comparison of data. Furthermore, pressure gauges located on the tank walls or on any obstacles within the tank provide another useful validation tool. Two experimental set-ups were selected that provide both depth and pressure gauge data for comparison to the multi-material model developed in the previous sections.

4.4.1 Two-Dimensional Water Column Collapse

Zhou *et al.* (1999) modelled a simple dam break problem experimentally in a $3.22 \times 2 \times 1m$ (length $\times$ height $\times$ depth) tank. A reservoir of water $1.2 \times 0.6 \times 1m$ (length $\times$ height $\times$ depth) was held behind a barrier at one end of the tank. Water depth gauges were placed at two points, marked H1 and H2 in Figure 4.20(a), at $x_1 = 2.725m$ and $2.228m$ respectively. Additionally, a pressure gauge was located at the point marked P2 in Figure 4.20(a), at $x_2 = 0.16m$ on the wall facing the initial water column.

As no variations were introduced in the third dimension, the experiment is reproduced here numerically in two dimensions within the domain Ω: $x_1 \in [0, 3.22]$, $x_2 \in [0, 2]$ (Lee *et al.*, 2002; Colagrossi & Landrini, 2003; Park *et al.*, 2009). The initial condition of the water volume fraction is shown in Figure 4.20(a). The presence of water, $\alpha^1 = 1$, is indicated as a grey region and the interface to air, $\alpha^1 = 0$, is delineated by contours at $\alpha^1 = 0.025$, 0.5 and 0.975. The density of the water is taken as $\rho^1 = 1,000kg\,m^{-2}$ while the density of air is given by $\rho^2 = 1kg\,m^{-2}$. Both fluids are treated inviscidly. No information was given about the surface drag inside the tank during the experiment so instead, as the simulation is inviscid, free slip boundary conditions, $d_i n_i = 0$, are imposed on the tank bottom, $x_2 = 0$, and sides, $x_1 = 0$, 3.22. The top of the tank, $x_2 = 2$, is left open, $p = 0$.

Similar two-dimensional numerical set-ups have been used by several other authors. Zhou *et al.* (1999) and Lee *et al.* (2002) both used a shallow water model, modelling the interface as a free surface and only considering hydrostatic variations in pressure. Colagrossi & Landrini (2003) used a smooth particle hydrodynamics (SPH) model, considering the continuum as a set of isolated particles which interact with their neighbours

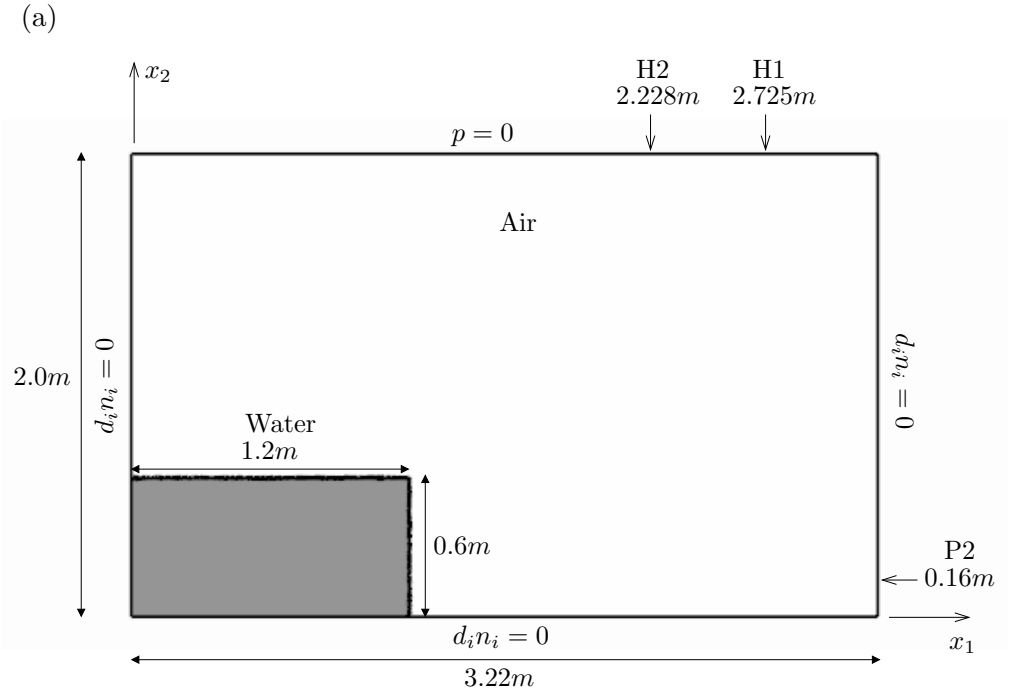


Figure 4.20: (a) Initial set-up of the water volume fraction, α^1, and the velocity and pressure boundary conditions for the two-dimensional water column collapse validation problem (Zhou *et al.*, 1999). The presence of water, $\alpha^1 = 1$, is indicated as a grey region and the interface to air, $\alpha^1 = 0$, is delineated by contours at $\alpha^1 = 0.025$, 0.5 and 0.975. The locations of the pressure (P2) and water depth gauges (H1, H2) are also indicated. (b) The adapted mesh used to represent the initial conditions.

within a specified smoothing length-scale through so-called smoothing functions. The SPH model was run using both a free surface interface (treating the air as a vacuum) and a 1,000:1 density ratio. Park *et al.* (2009) used a coupled level set-VOF material tracking scheme on a structured orthogonal mesh to model the evolution of the interface between the water and air. Three meshes were used in Park *et al.* (2009): a coarse, 168×112 node mesh, a medium, 238×158 node mesh, and a fine, 336×224 node mesh. In each case a high resolution zone, containing 30, 43 and 60 grid points respectively, was place within $0.1m$ of the base and right wall of the domain in an attempt to more accurately capture the dynamics at the boundaries.

Here, the water volume fraction, α^1, is advected using HyperC (see Chapter 3) on the control volume mesh dual to the piecewise linear finite element parent mesh while the velocity and pressure are discretised using the $\mathrm{P0P1_{CV}}$ element pair with $\theta = 1/2$ and $\theta_i = 1/2$. The domain is discretised using the same resolution as the medium case of Park *et al.* (2009), with 238×158 grid points distributed evenly throughout the domain except in a high resolution zone within $0.1m$ of the lower and right boundaries where 43 points are concentrated. The timestep is selected to achieve a Courant number of 2.5 while the advection equation uses approximately 10 subcycles so the volume fraction is advected at a Courant number of 0.25. Several timesteps of this simulation can be seen in Figure 4.21(i) where the interface is represented by contours of the water volume fraction, α^1, at 0.025, 0.5 and 0.975. The effect of the high resolution zone can be seen most clearly in Figure 4.21(i)(a) where the contours become more tightly packed towards the bottom of the domain. This effect diminishes as the water flood impacts into the opposing wall (Figure 4.21(i)(b)) becoming an overturning wave (Figure 4.21(i)(c)) and leaving the boundary layer. As the simulation continues the overturning wave travels back into the domain (Figure 4.21(i)(d)) and the dynamics become increasingly complex, trapping air bubbles and generating new waves (Figure 4.21(i)(e–g)). While HyperC performs very well tracking the location of the water, by the end of the simulation several regions of mixed cells can be seen (Figure 4.21(i)(h)).

While the strategy of increasing the resolution around the boundaries works well during the initial stages of the simulation ($t < 1.5$) once the principal dynamics have left this zone the resolution is in the wrong place to adequately capture the interface. Therefore an adaptive simulation was also performed with the minimum edge length constrained to $2mm$, equivalent to the high resolution zone in the fixed mesh simulation. The upper bound on the edge lengths was specified as half the domain length and height in each dimension. The pressure and water volume fraction were directly adapted to using interpolation error bounds, $\hat{\epsilon}$, of $1,000$ and 0.05 respectively. As the velocity is element centred it was first projected to the vertices before being adapted to with an interpolation error bound of 1 for each component. Given the ranges of these fields seen in the fixed mesh runs these correspond to desired errors of less than 5% for each field. As in Section 3.5 the volume fraction is transferred between successive meshes using a minimally diffusive bounded projection algorithm. The velocity is transferred using a straightforward projection while the pressure is consistently interpolated using the linear basis functions from its parent mesh. All the remaining adaptivity settings are the same

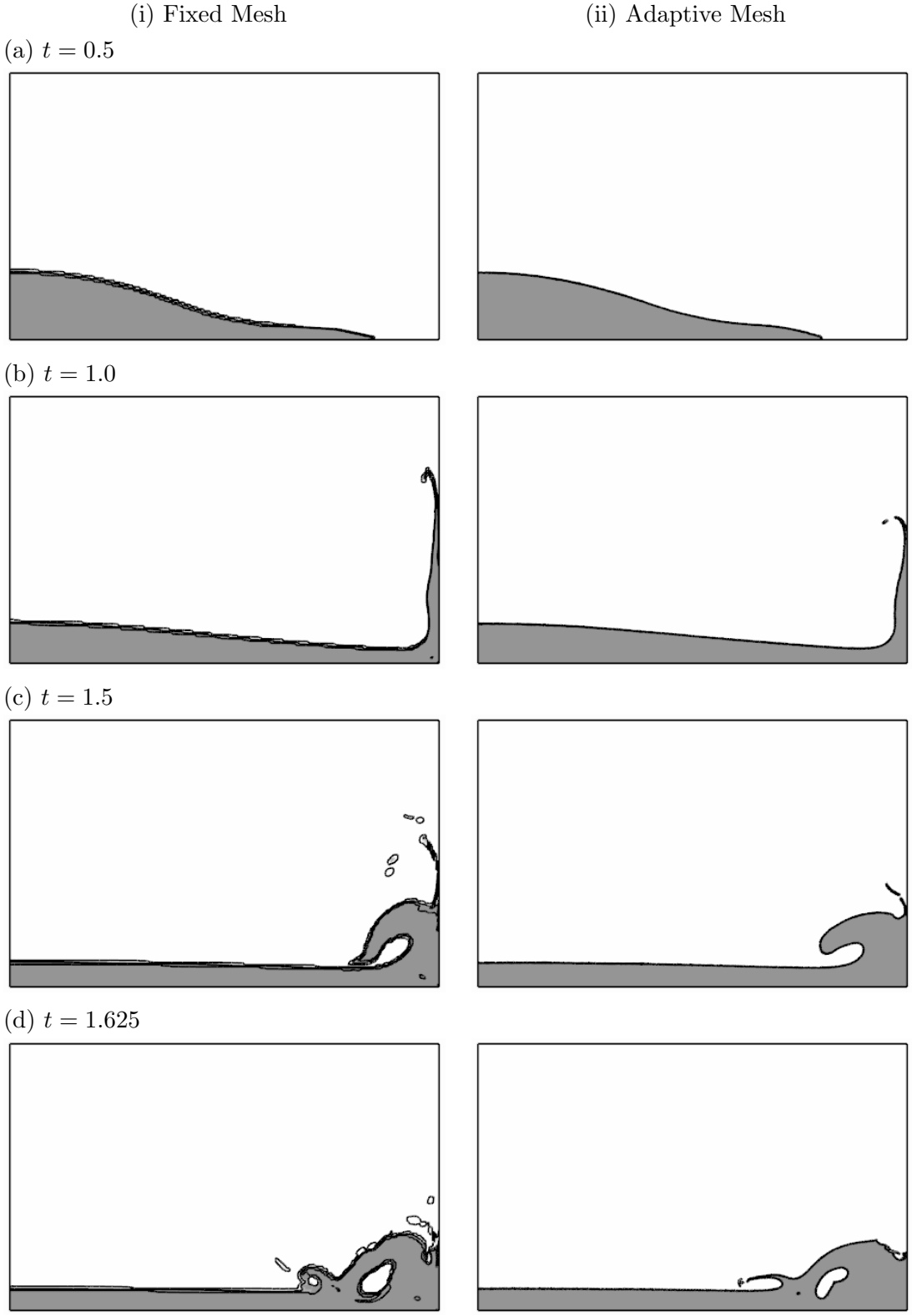


Figure 4.21: The evolution of the water volume fraction, α^1, over several timesteps on a fixed 238×158 mesh (i) and an adaptive mesh with the same minimum edge length (ii). The presence of water, $\alpha^1 = 1$, is indicated as a grey region and the interface to air, $\alpha^1 = 0$, is delineated by contours at $\alpha^1 = 0.025$, 0.5 and 0.975.

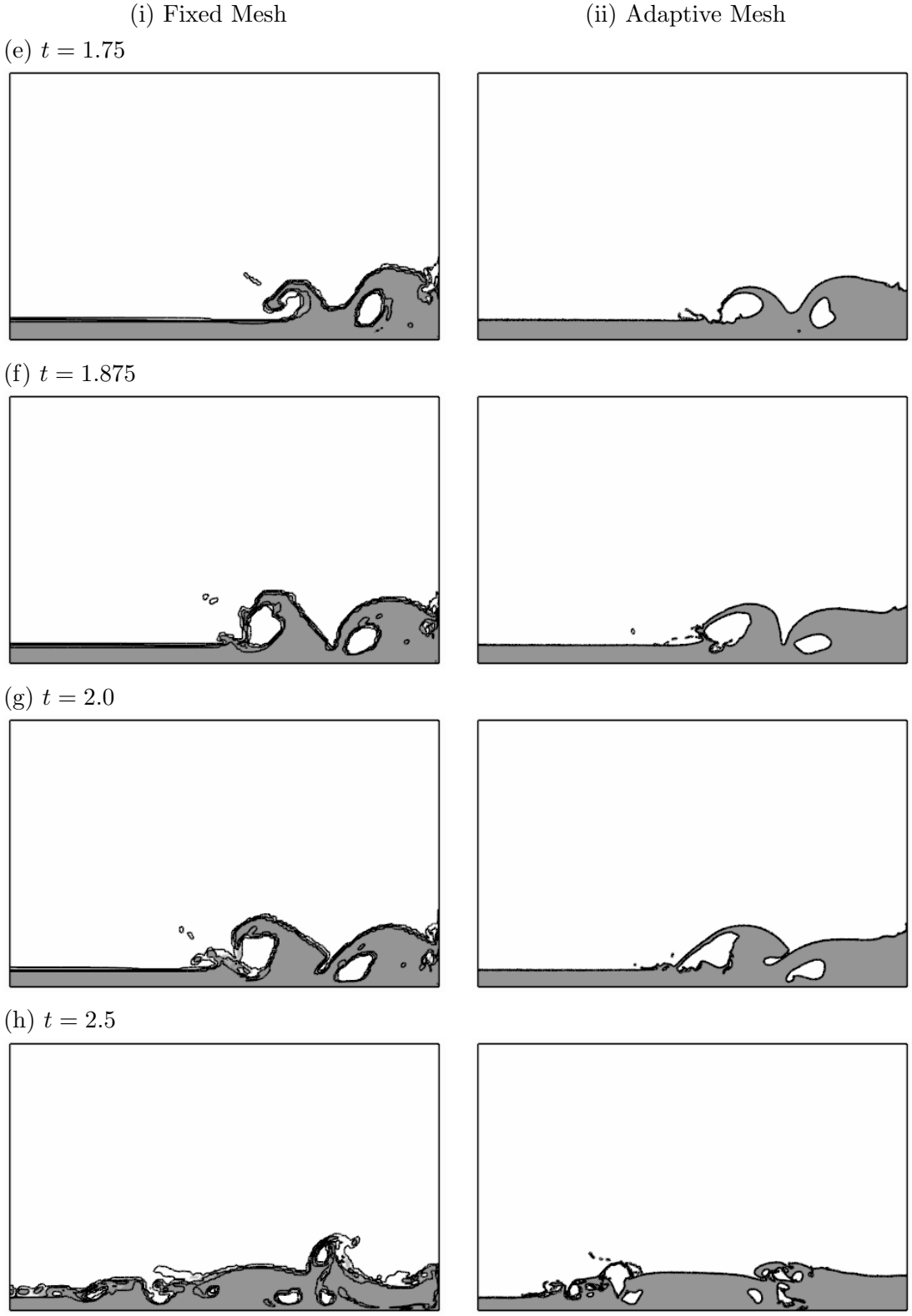


Figure 4.21: *cont.* The evolution of the water volume fraction, α^1, over several timesteps on a fixed 238×158 mesh (i) and an adaptive mesh with the same minimum edge length (ii). The presence of water, $\alpha^1 = 1$, is indicated as a grey region and the interface to air, $\alpha^1 = 0$, is delineated by contours at $\alpha^1 = 0.025$, 0.5 and 0.975.

as used in Figure 3.55 with metric advection, edge length gradation and the solenoidal projection of the velocity interpolant following mesh optimisation. The initial mesh using these settings is shown in Figure 4.20(b).

Results using the adaptive mesh are presented alongside the fixed mesh results in Figure 4.21(ii) while the corresponding adapted meshes are shown in Figure 4.22. The two sets of results are very similar showing the column collapse (Figure 4.21(a)), run-up against the opposing wall (Figure 4.21(b)) and subsequent overturning wave (Figure 4.21(c, d)) and entrainment of air bubbles (Figure 4.21(e–h)). However, discrepancies between the solutions do occur. The adaptive mesh displays closely packed contours along the entire length of the interface throughout the simulation, not just towards the base in the early timesteps as in the fixed mesh run. Furthermore, very few mixed cells are visible in the adaptive mesh run, even towards the end when the dynamics become increasingly complicated. This is a consequence of finer resolution around the entire interface throughout the adaptive mesh simulation.

While the broad dynamics are similar between the two simulations the adaptive mesh run is seen to have a lower wave run-up on the opposing wall (Figure 4.21(b)), resulting in a slightly different overturning wave (Figure 4.21(c)). It is unclear if this is a result of having less mixing in the interface zone, resulting in a better defined, less buoyant water region, or whether some interaction between the adapting mesh and the advection is taking place, as was seen in Figure 3.51. Wave run-up using adapting meshes is investigated further in Chapter 5. After about $t = 1.5s$ the overturning wave impacts with the basal water layer. At this point the dynamics become increasingly complex and discrepancies between the adaptive and fixed mesh simulations become more apparent. However the adaptive mesh still maintains a much sharper interface and displays broadly the same behaviour as the fixed mesh run.

The minor discrepancies between the different runs of the simulation can also be seen from the water depth gauge data displayed in Figure 4.23 alongside the experimental data. Following Colagrossi & Landrini (2003) and Park *et al.* (2009) the numerical results show the total thickness of water at the points H1 and H2, discounting any air bubbles that cross the gauges. Both simulations produce very similar results during the initial collapse phase ($t < 1.5s$). These also show a close similarity to the experimental results with the exception of a small lip of water when the initial water head passes the gauge. It is unclear what causes this structure, though it may be related to the initial withdrawal of the barrier in the experiment or drag effects from the bottom of the tank. To the author's knowledge, all previous published attempts to model the experiment also fail to reproduce this initial lip (Zhou *et al.*, 1999; Lee *et al.*, 2002; Colagrossi & Landrini, 2003; Park *et al.*, 2009).

After $t = 1.5s$ the overturning wave starts to pass the water gauges and the match between the experimental results and each of the numerical simulations deteriorates. The differences between the simulations can be explained by the discrepancies in the positions and shapes of the trapped air bubbles which have a large impact on the depth gauge data for $t > 1.5s$. Presumably this is also the case for the experimental data, although no snapshots of the simulation were published. As would be expected from such complex

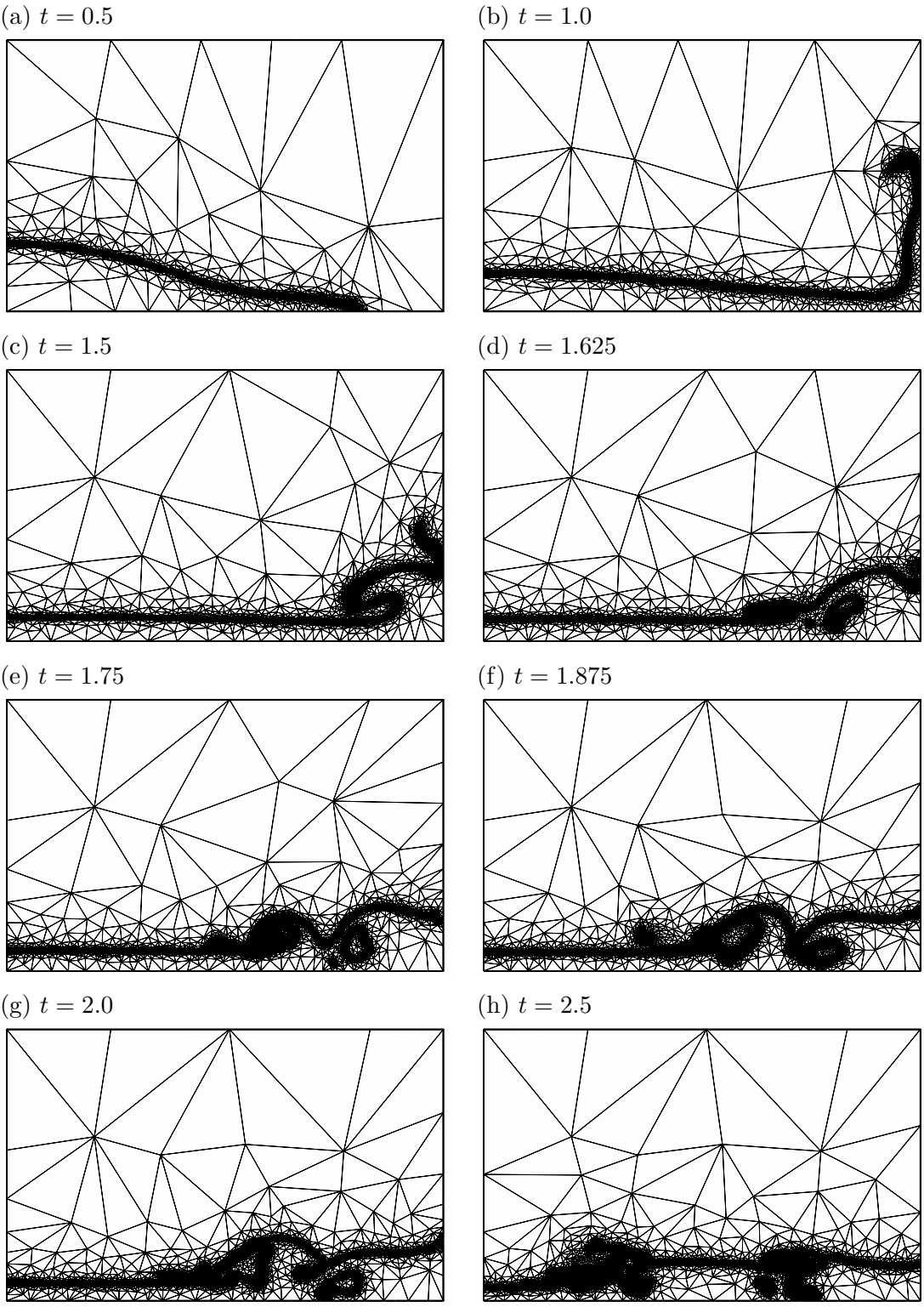


Figure 4.22: The evolution of the adaptive mesh over the same timesteps displayed in Figure 4.21(ii). The mesh can be seen to closely follow the interface between the water and air.

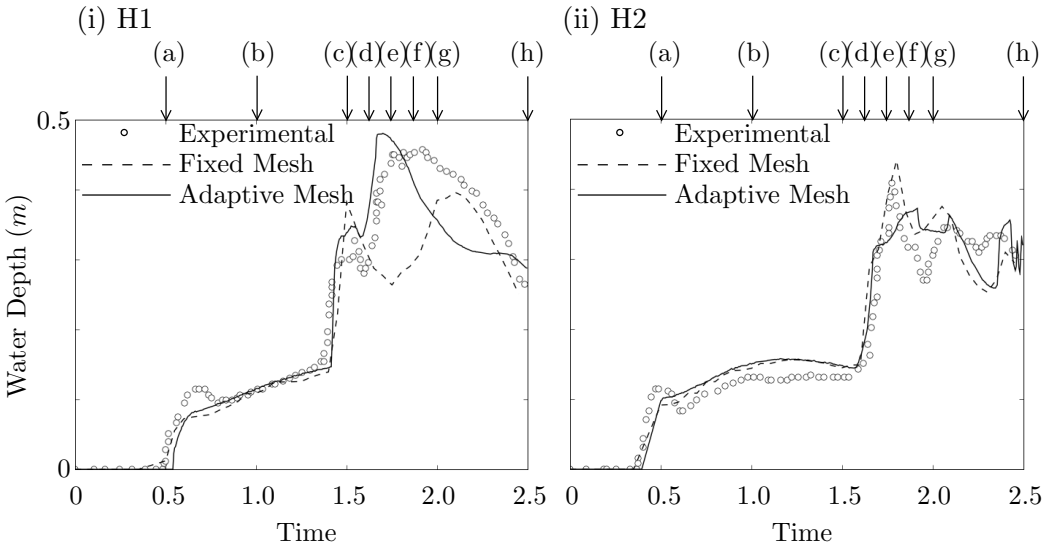


Figure 4.23: Comparison between the experimental (circles) and numerical water gauge data at H1 (i) and H2 (ii), $x_1 = 2.725m$ and $2.228m$ respectively. Numerical data are provided for the fixed (dashed line) and adaptive (solid line) experiments. The letters along the top of the graphs indicate the times corresponding to Figure 4.21(a–h). Experimental data taken from Zhou *et al.* (1999) through Park *et al.* (2009).

behaviour, to the author's knowledge, all previous published attempts have also failed to reproduce the experimental depth gauge data after this point. However, the broad pattern and average depth after $t = 1.5s$ can be seen in Figure 4.23 to match the experiment reasonably well for both the simulations.

Experimental pressure gauge data are also available at the point P2, $(3.22, 0.16)m$, on the right wall of the tank. This is compared to the numerical pressure results in Figure 4.24. After the initial noise in the experimental data, a sudden step in pressure is seen as the water run-up reaches the pressure gauge at about $t = 0.6s$. This is also seen in the numerical simulations however it is slightly delayed, occurring at $t = 0.7s$. As upwinding is being used in the discretisation of the velocity field, the delay may be due to numerical viscosity slowing the advancing water front. However, as the delay was not as extreme at the depth gauges H1 and H2 other factors may also play a role. For instance, if the lip seen in the experimental water gauge data is a head on the water front, that has not been reproduced numerically, it may reach the height of the pressure gauge faster than a front with no head.

Once the pressure jump occurs the experimental and numerical data are in broad agreement until the overturning wave impacts with the water layer at approximately $t = 1.5s$ (Figure 4.21(c)). At the point of contact a pressure pulse is transmitted to the pressure gauge resulting in a modest pressure spike in the experimental data. This is matched by slightly delayed pressure pulses in all the numerical simulations. However, the pulses seen in the numerical data are of a much larger magnitude than the experimental data. A series of experiments were conducted to try to ascertain the cause of

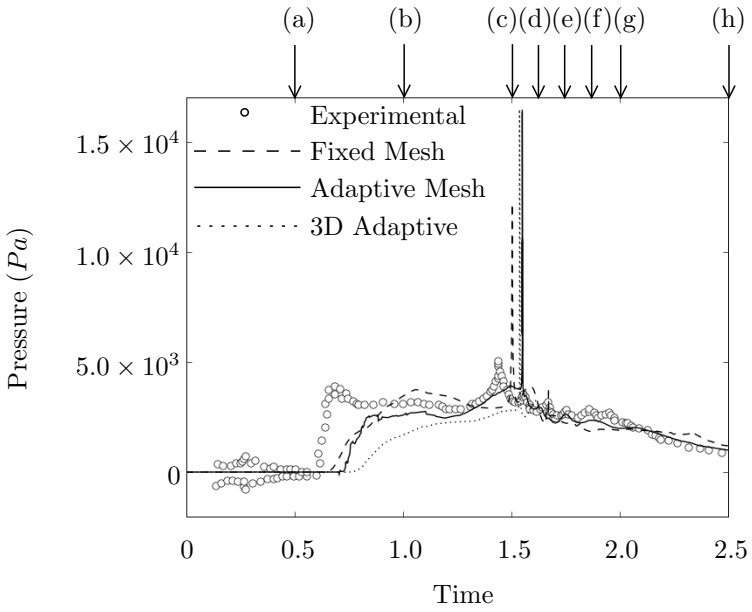


Figure 4.24: Comparison between the experimental (circles) and numerical pressure gauge data at P2, $x_1 = 3.22m$, $x_2 = 0.16m$. Numerical data are provided for the fixed (dashed line), two-dimensional adaptive (solid line) and three-dimensional adaptive (dotted line) experiments. The letters along the top of the graphs indicate the times corresponding to Figure 4.21(a–h). Experimental data taken from Zhou *et al.* (1999) through Park *et al.* (2009).

this discrepancy. Increasing the mesh resolution in both the fixed and adaptive mesh simulations resulted in the magnitude of the pulse converging on a higher rather than lower value. Similarly, a lower resolution (minimum edge length of $3mm$ with a maximum desired error of about 10%) but three-dimensional adaptive mesh simulation of the entire tank produced the same large magnitude pulse (see dotted line in Figure 4.24).

The shallow water models used by Zhou *et al.* (1999) and Lee *et al.* (2002) were unable to model the overturning wave. Similarly the free surface SPH model of Colagrossi & Landrini (2003) failed to adequately reproduce any of the pressure behaviour after $t = 1.5$. Their two material SPH simulation did produce a pressure pulse, which was similarly overestimated however, unlike here, the subsequent pressure data was also contaminated by large oscillations. The pressure gauge results of Park *et al.* (2009) look most similar to those seen here with a delayed, large magnitude pulse which increases with increasing resolution. Without further information about the experimental conditions, the cause of the discrepancy between the pressure pulse from the overturning wave is therefore unclear. However, the magnitude of the subsequent experimental pressure oscillations was reproduced well despite the initial error around $t = 1.5s$.

Having established that the multi-material model using HyperC and a $\mathrm{P0P1_{CV}}$ velocity-pressure discretisation reproduces the experimental water column collapse of Zhou *et al.* (1999) as well as other numerical methods, it is interesting to look at some of the technical

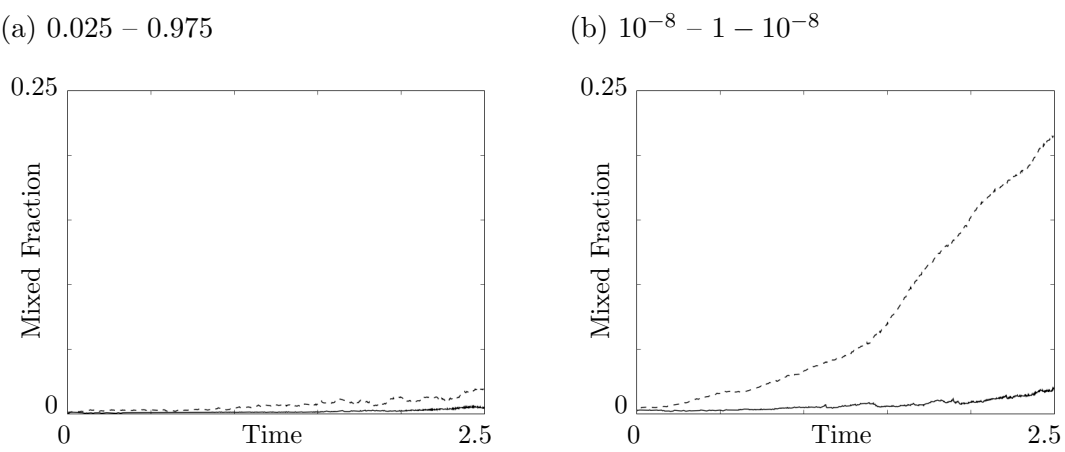


Figure 4.25: Fractional area of the domain which consists of mixed cells over time for the two-dimensional water column collapse. Two ranges of mixing are shown: 0.025 – 0.975 (a) and 10^{-8} – $1 - 10^{-8}$ (b) for the simulation on the fixed (dashed) and adaptive (solid) meshes.

differences between the fixed and adaptive mesh simulations. It was already noted visually that the adaptive simulation maintained resolution around the interface throughout the simulation, which led to fewer mixed cells being generated. This can also be seen in Figure 4.25 where the fraction of the domain where the volume fraction lies between 0.025 and 0.975 (Figure 4.25(a)) and between 10^{-8} and $1 - 10^{-8}$ (Figure 4.25(b)) can be seen for each run. In both measures the adaptive simulation minimises the mixing, particularly in Figure 4.25(b) where nearly a quarter of the domain may be defined as mixed in the fixed mesh run, ten times as much as the adaptive case.

Improved mixing statistics in the adaptive simulation are achieved by increasing the resolution. However, as this resolution is targeted in a limited zone around the interface the total number of nodes remains lower than in the fixed mesh run (see Figure 4.26). The number of vertices in the adapting mesh remains at approximately a quarter of the fixed number until about $t = 1.5s$. At this point the dynamics become more complicated. With an increasing length of interface the adaptive mesh responds by increasing the number of nodes until, at the end of the simulation, the number of vertices is about three quarters that of the fixed mesh equivalent. In addition to improving the solution by reducing the amount of mixing, the adaptive mesh therefore is also more efficient, only ever using the minimum number of nodes required. On an unstructured mesh this increased efficiency is essential to overcome the additional costs of arbitrary nodal indexing, adapting the mesh and interpolating the field data. In three dimensions the number of nodes in structured meshes scales poorly and unstructured, adaptive meshes potential for improved efficiency increases.

4.4.2 Three-Dimensional Water Column Collapse

Kleefsman (2005) performed a slightly more complicated water column collapse experi-

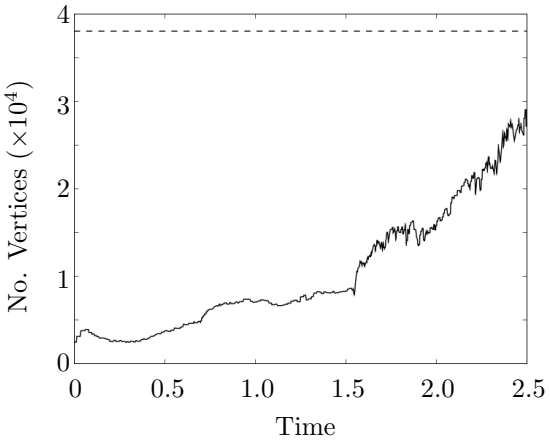


Figure 4.26: Vertex count over time for the two-dimensional water column collapse, comparing the fixed (dashed) and adaptive (solid) numerical experiments.

ment in a $3.22 \times 1 \times 1m$ (length $\times$ height $\times$ depth) domain. This was divided between a reservoir section, which occupied $1.22m$ of one end of the tank and was filled to a height of $0.55m$ with water, and an initially dry tank region. Inside the tank a fixed obstacle measuring $0.16 \times 0.16 \times 0.4m$ (length $\times$ height $\times$ depth) was placed in the centre of the tank $0.67m$ from the end facing the water column. Water depth gauges were placed inside the initial water column at $x_1 = 2.66$ (H4) and between the water column and the obstacle inside the tank region at $x_1 = 1m$ (H2). A pressure gauge, P1, was placed slightly off centre on the obstacle facing the water column at $(0.831, 0.026, 0.025)m$.

The obstacle in the experimental set-up means it cannot be reproduced numerically in two dimensions. Therefore a simulation was performed in three dimensions. Only half the original domain needs to be discretised however, as a plane of symmetry exists through the centre of the tank. The initial set-up in this region, Ω: $x_1 \in [0, 3.22]$, $x_2 \in [0, 0.5]$, $x_3 \in [0, 1]$, as well as the boundary conditions and locations of the gauges is shown in Figure 4.27. The presence of water, $\alpha^1 = 1$, is indicated as a grey block. The density of the water is taken as $\rho^1 = 1,000 kg\, m^{-3}$ while the density of air is given by $\rho^2 = 1 kg\, m^{-3}$. Both fluids are treated inviscidly. No information was given about the surface drag inside the tank during the experiment so instead free slip boundary conditions, $d_i n_i = 0$, are imposed on the tank bottom, $x_3 = 0$, and sides, $x_1 = 0$, 3.22 and $x_2 = 0$, 0.5. The top of the tank, $x_3 = 1$, is left open, $p = 0$.

As being in three dimensions increases the computational cost of numerical simulations considerably, only an adaptive mesh model is considered. This uses a slightly lower resolution than in two dimensions, constraining the minimum edge length to $3mm$. The maximum edge lengths are limited to the domain size in each dimension. The interpolation error bounds on all the fields are the same as in two dimensions, as are the methods used to transfer the field data between the meshes following each step of mesh optimisation. The remainder of the adaptivity settings, such as metric advection and edge length gradation, are also identical to the two-dimensional case. The initial mesh using these

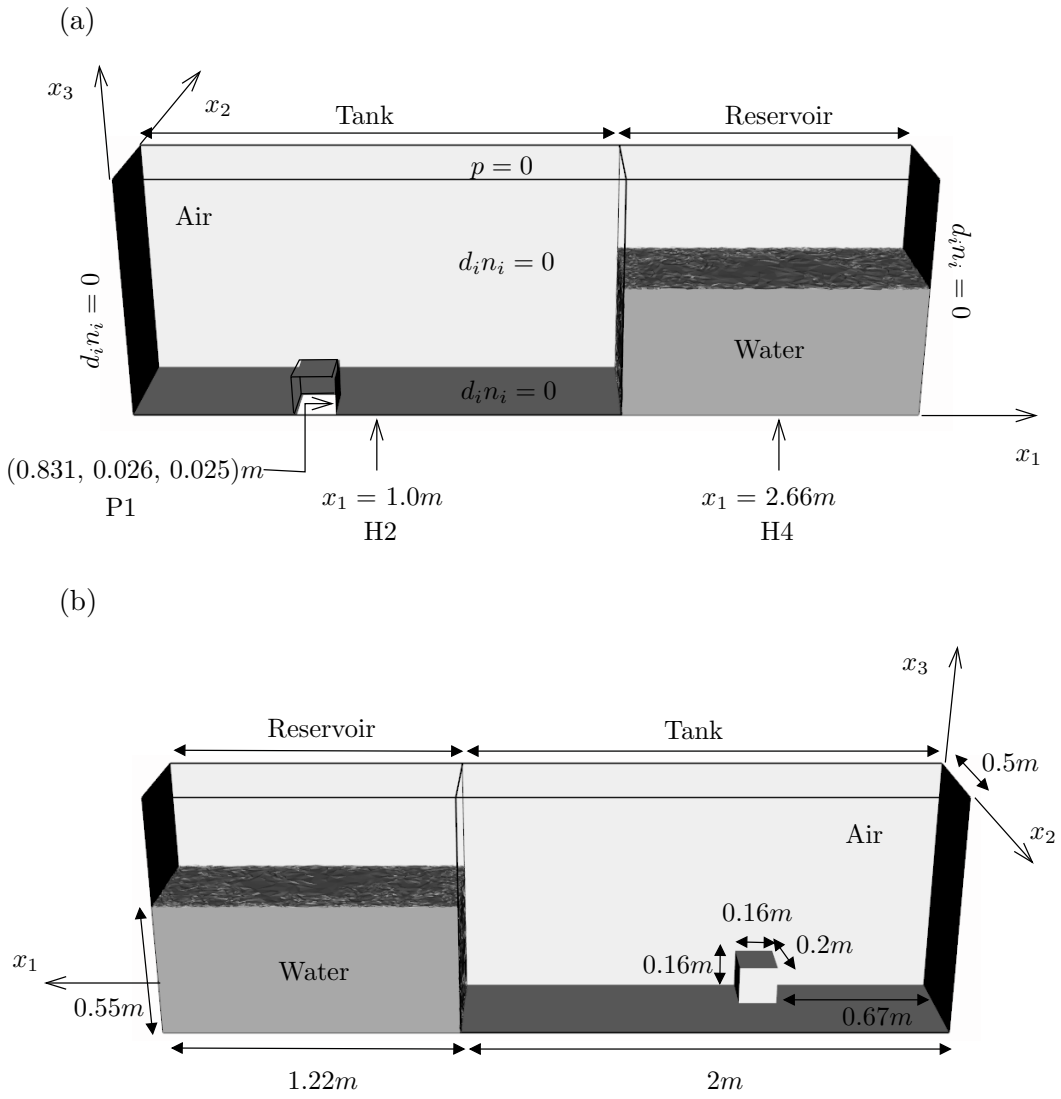
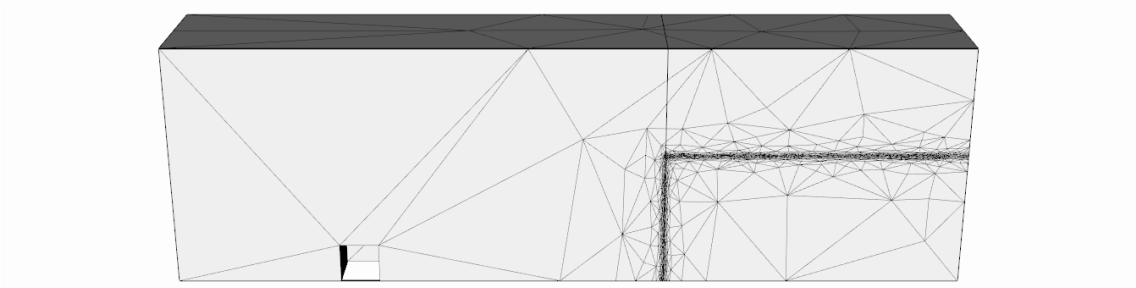


Figure 4.27: (a) Initial set-up of the water volume fraction, α^1, and the velocity and pressure boundary conditions for the three-dimensional water column collapse validation problem (Kleefsman, 2005). The domain is divided into a tank and a reservoir region, where the water, $\alpha^1 = 1$, is located initially (grey region). The locations of the pressure (P1) and water depth gauges (H2, H4) are also indicated. (b) Reverse view of (a) showing the dimensions of the domain and the location and size of the obstacle in the tank region.

(a)



(b)

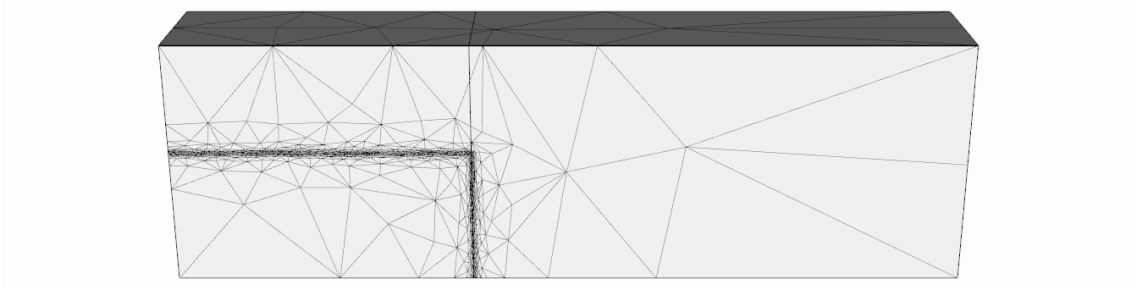


Figure 4.28: (a) Initial mesh used in the initial set-up of the three-dimensional water column collapse validation problem displayed on the surface of the domain. (b) Reverse view of (a).

settings is shown in Figure 4.28 displayed on the surface of the domain only.

UltraC-S is selected to discretise the advection equation, thus preventing any numerical diffusion that might be introduced by the decreased mesh resolution. The water volume fraction is then advected at a maximum Courant number of 0.25, subcycling approximately 10 times per timestep, which is set to give a maximum Courant number of 2.5 with $\theta = 1/2$ and $\theta_i = 1/2$ in the momentum equation.

Close-ups of the tank section of the domain from several timesteps during the collapse and impact of the water column are shown in Figures 4.29–4.30 viewed from two directions, alongside the corresponding adapted meshes. The water column initially inundates the domain (Figures 4.29(i)(a) and 4.30(i)(a)) just like in the two-dimensional example. However, after the obstacle is struck by the water front (Figures 4.29(i)(b) and 4.30(i)(b)) the flow becomes three-dimensional. Close to the plane of symmetry a jet of water travels upwards before beginning to collapse forward (Figures 4.29(i)(c–e) and 4.30(i)(c–e)). On the other side of the tank the water carries on past the obstacle (Figures 4.29(i)(b) and 4.30(i)(b)), flooding the region directly behind it (Figures 4.29(i)(c) and 4.30(i)(c)) before impacting with and rushing up the tank wall (Figures 4.29(i)(d–e) and 4.30(i)(d–e)).

The comparison between the results from Figures 4.29–4.30 and the original experiment at the water depth gauges H2 and H4 is shown in Figure 4.31. The height of the water column within the reservoir region drops in the numerical simulation at almost exactly the same rate as in the experiment (Figure 4.31(b)). However, the rise in water height within the tank region appears to be delayed slightly (by about $0.05s$, see Figure 4.31(a)). Once the water head has passed the gauge the magnitudes quickly come into alignment. A delay is also seen in the initial pressure pulse, when the water head impacts into the obstacle, at the pressure gauge P1 (see Figure 4.32).

The discrepancies between the timings of the experimental and numerical solutions are most likely caused by numerical viscosity introduced by upwinding in the momentum equation. When running the same numerical validation test Park *et al.* (2009) also noted the relative deceleration of the flow speeds compared to the experiment and attributed it to increased numerical damping effects. Kleefsman (2005) also saw an offset of a similar magnitude in the impact pressure pulse but found that the delay was approximately halved by increasing the resolution. Here the number of pressure nodes is approximately $100,000$ at the time of the impact, although the simulation started with about $2,000$ which increased steadily to approximately $120,000$ by its end. This resolution is more focussed than in the structured meshes used by Kleefsman (2005) but still an order of magnitude lower than their finest mesh, which contained over a million nodes. In all cases, as here, the pressure returned to the correct magnitude after the delay.

Kleefsman (2005) provided two snapshots of the simulation for qualitative comparison. These are reproduced next to the corresponding numerical timeframes in Figure 4.33. Two timeframes are shown per snapshot illustrating the numerical delay of about $0.05s$. Despite the delay the shape of the collapse and subsequent impact with the obstacle look very similar. This suggests that with further investigation of the time discrepancies, for instance by increasing the resolution or adjusting the momentum advection scheme,

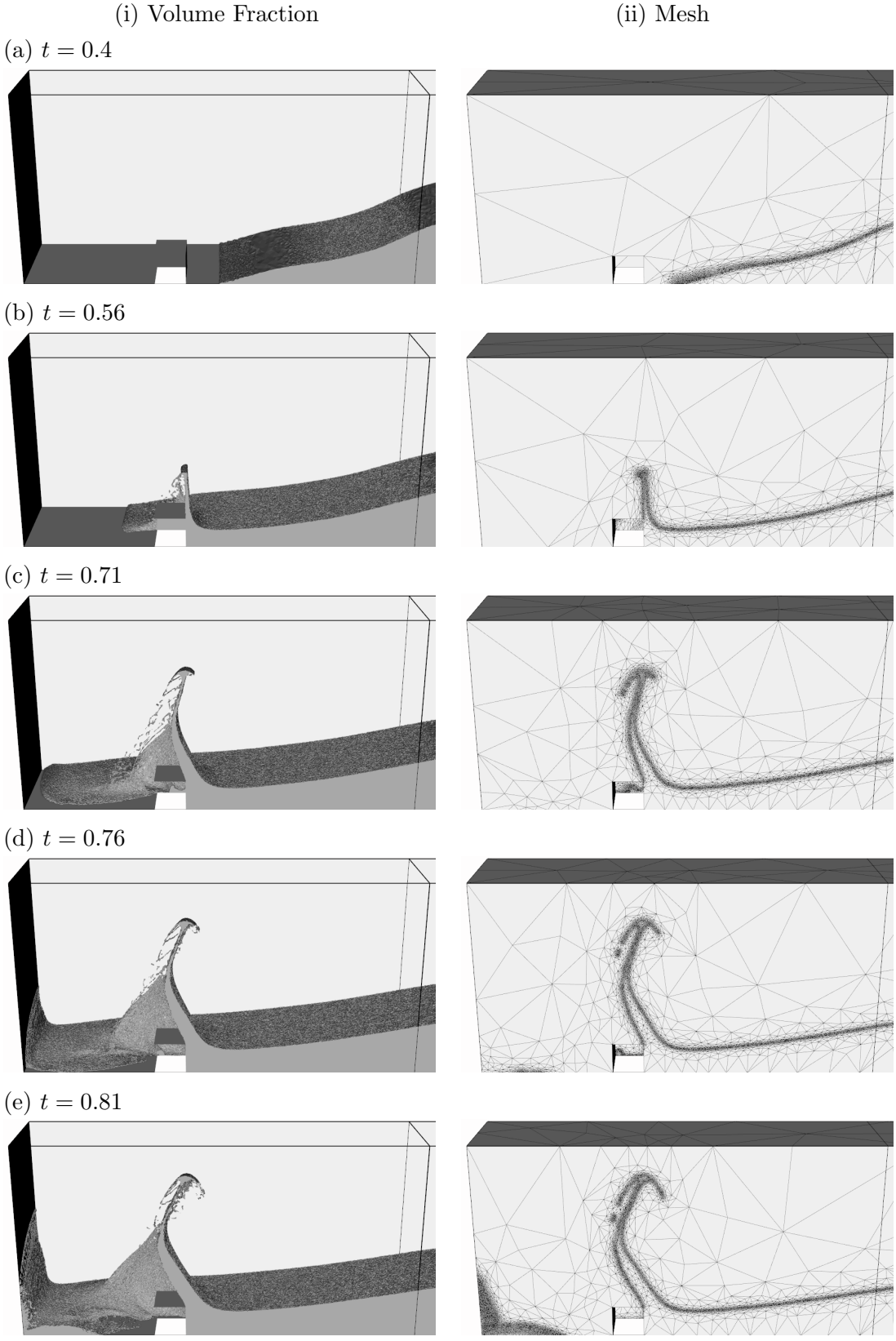


Figure 4.29: Close-ups of the tank region of the three-dimensional water column collapse validation problem showing the evolution of the water volume fraction, α^1, over several timesteps (i) and the adaptive mesh displayed on the domain surface (ii). The presence of water, $\alpha^1 = 1$, is indicated as a grey region.

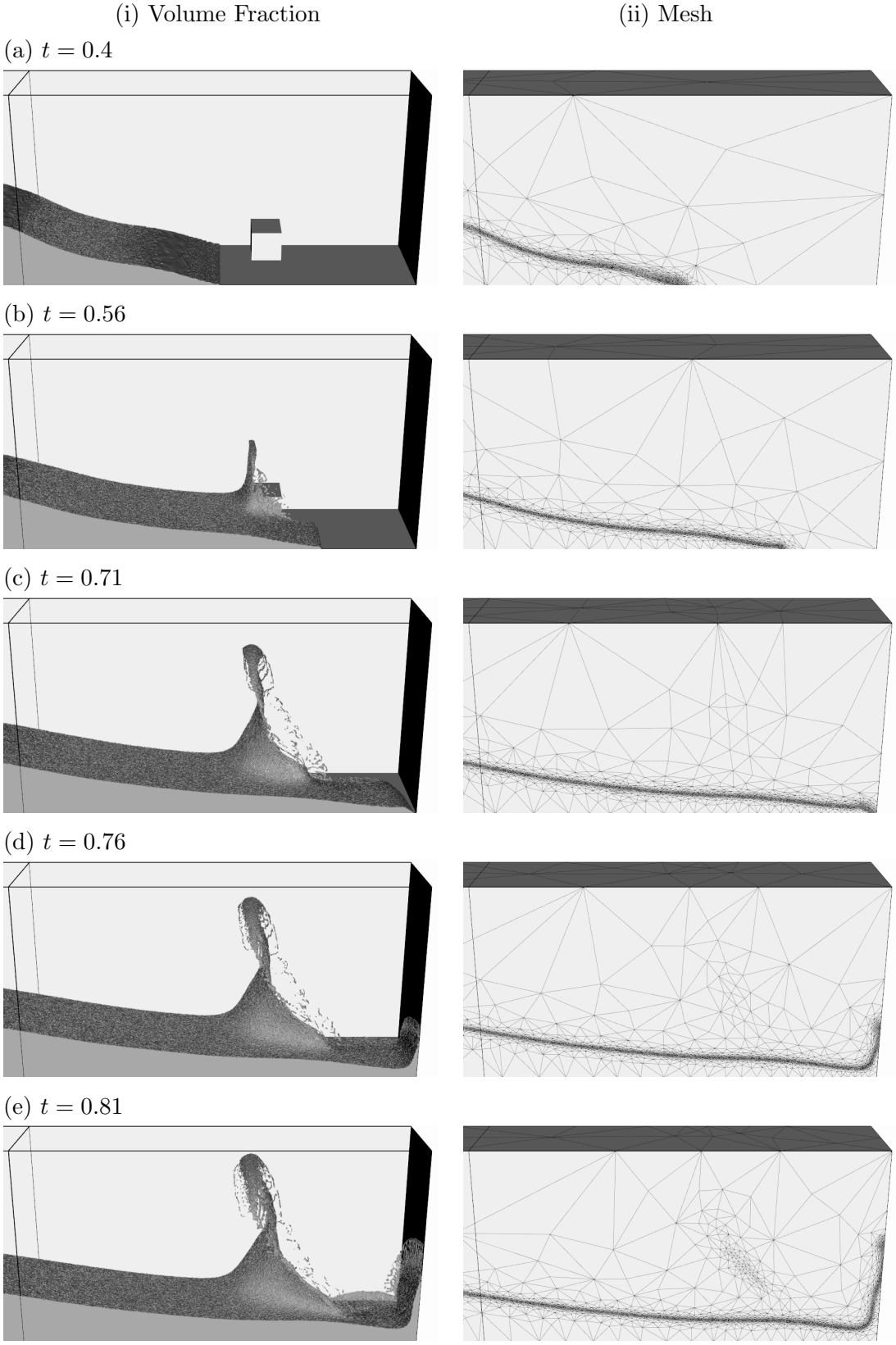


Figure 4.30: Reverse view of Figure 4.29. Close-ups of the tank region of the three-dimensional water column collapse validation problem showing the evolution of the water volume fraction, α^1, over several timesteps (i) and the adaptive mesh displayed on the domain surface (ii). The presence of water, $\alpha^1 = 1$, is indicated as a grey region.

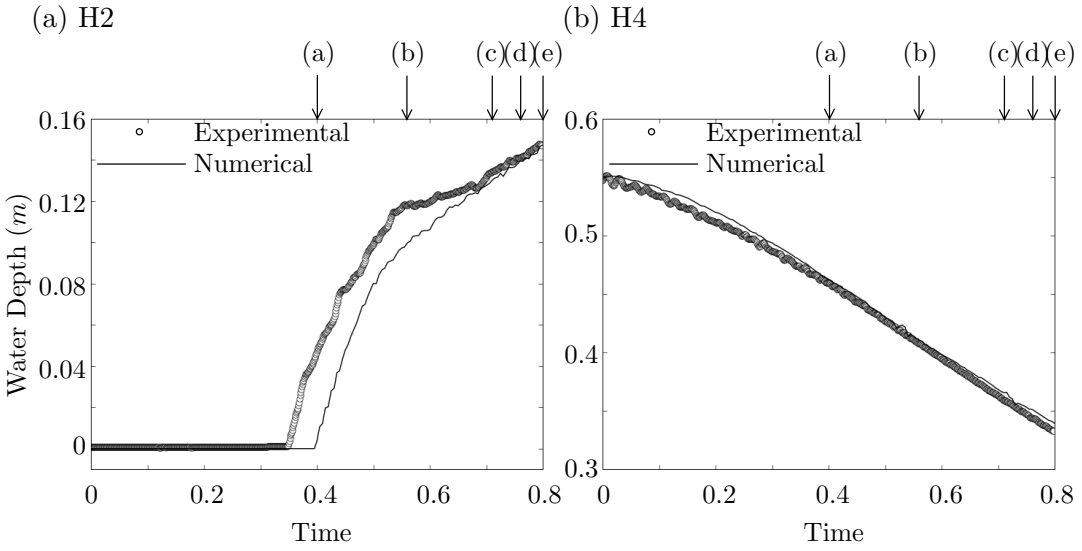


Figure 4.31: Comparison between the experimental (circles) and numerical (solid line) water gauge data at H2 (a) and H4 (b), $x_1 = 1m$ and $2.66m$ respectively. The letters along the top of the graphs indicate the times corresponding to Figures 4.29(a–e) and 4.30(a–e). Experimental data taken from Kleefsman (2005).

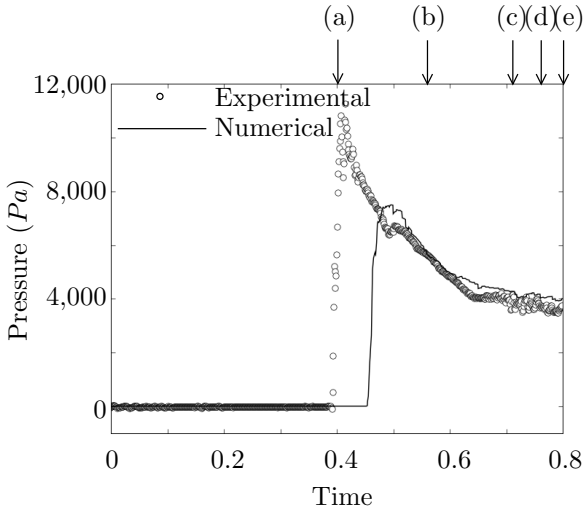
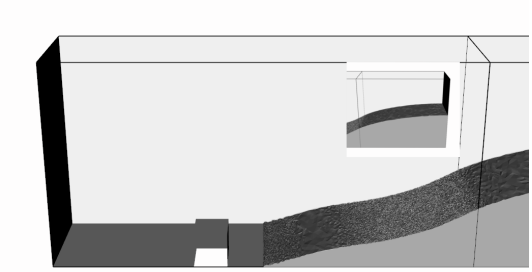


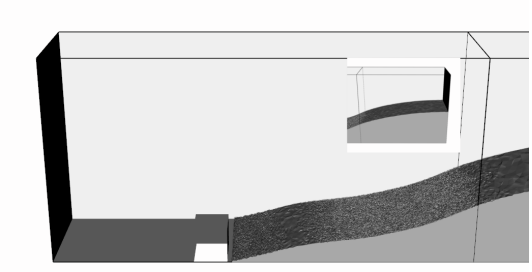
Figure 4.32: Comparison between the experimental (circles) and numerical (solid line) pressure gauge data at P1, $x = (0.831, 0.026, 0.025)m$. The letters along the top of the graph indicate the times corresponding to Figures 4.29(a–e) and 4.30(a–e). Experimental data taken from Kleefsman (2005).

(a)

(i) $t = 0.4$



(ii) $t = 0.45$

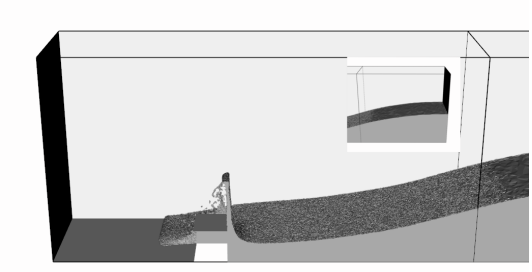


(iii) $t = 0.4$

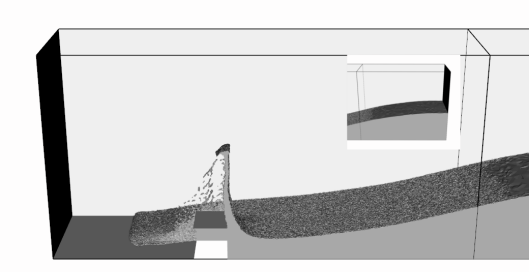


(b)

(i) $t = 0.56$



(ii) $t = 0.61$



(iii) $t = 0.56$



Figure 4.33: Two timeframes (a) and (b) from the numerical (i, ii) and experimental (iii) three-dimensional water column collapse. Close-ups of the tank region of the domain are shown with smaller insets from the reservoir end. Numerical results are shown from the same time (i) as the experimental snapshots (iii) and at the slightly later point where they look more visually alike (ii). The presence of water, $\alpha^1 = 1$, is indicated as a grey region in the numerical results. Experimental snapshots taken from Kleefsman (2005).

UltraC-S coupled to a $P0P1_{CV}$ momentum discretisation provides a suitable framework for modelling three-dimensional flows.

4.5 Summary

In this chapter the role of pressure within the momentum and mass equations has been discussed and some numerical discretisations suggested for the coupled velocity-pressure system. Several of these were implemented and tested on simplex meshes. Their suitability for modelling multiple material flows using a control volume discretisation for the volume fraction was then considered and it was noted that the pressure discretisation must be modified to ensure conservation of the volume fraction. This led to the development of several novel element pairs, which were then tested on a simple buoyancy force balance problem. On the basis of this test the new $P0P1_{CV}$ element pair was selected as the most suitable discretisation method for the velocity-pressure system in multi-material models on simplex meshes. The implementation of this element pair was verified and shown to have optimal convergence properties.

Using the $P0P1_{CV}$ element pair the multi-material model was tested on the commonly used dam break or water collapse validation problem. In two dimensions this was coupled to a volume fraction advected using HyperC while in three dimensions UltraC-S was used. In both cases the results were found to be comparable both to the corresponding experiment and to published data from other numerical models and methods. Some discrepancies in the times of events between the numerical results and the experiments warrant further investigation.

Having successfully modelled two materials, the next chapter is devoted to extending the advection scheme to be able to cope with arbitrary numbers of volume fractions. This involves preventing the sum of the volume fractions occupying any cell from exceeding the physical bound of one and is achieved by coupling the advection of successive volume fractions to each other. Introducing this capability allows arbitrary numbers of materials to be considered in a simulation.

Chapter 5

Multiple Materials

In the previous two chapters a two material model of incompressible fluid flow on unstructured, adaptive meshes was developed and validated against both experimental and analytical solutions. In a two material model only one volume fraction needs to be advected around the domain in order to uniquely specify the locations of both materials and the interface between them. Similarly, in a K material model, only $K-1$ prognostic volume fractions are required. Each of these represents the location of a material and the interface between it and all the other materials. The K^{th} material can then be found through Equation 3.1.

Advecting more than one volume fraction using any of the material tracking schemes discussed in Chapter 3 generally leads to non-physical negative volume fractions for the K^{th} material, α^K. While the result of the advection of each volume fraction is bounded beneath 1 the sum of the volume fractions is not guaranteed to be, except when each is advected using first-order upwinding. In regions where more than one material is present this may produce negative values for α^K.

Assuming the sum of the prognostic volume fractions is bounded prior to the advection of each material, then an unbounded sum will occur if overall, through the successive advection of each volume fraction, more material is transported into a volume than the total transported out in that timestep. This problem happens because each volume fraction is advected independently. The material tracking algorithm will ensure that no more of one material is propagated out of a cell than should be in a timestep but it has no knowledge of how much material has been or will be accumulated by the advection of the other volume fractions. It is the sum of these fluxes that must not exceed the total flux rather than their individual components. A truly multi-material advection algorithm must somehow couple the transport of all the materials together.

This is normally achieved through the imposition of priority ordering on the volume fractions. In this approach, an order is imposed or devised for each material. The highest priority volume fraction is then advected as normal from a cell. The lower priority materials must then satisfy not only their own boundedness criteria but also additional restrictions that ensure that their fluxes, when added to those of the higher priority materials, do not exceed the total allowed over that timestep. This approach is referred to as an *onion skin* model (Benson, 2002) and in terms of interface reconstruction can be

thought of as deciding the relative positions of the interfaces in cells that contain more than two materials.

In the simplest onion skin model the user specifies the priority ordering of the materials. For example, when modelling a landslide generated wave it may be decided that the position of the water is the most important quantity and it should always be advected first. The landslide might be considered second, taking into account the preceding fluxes of water. Finally, the air is considered as the lowest priority material which is calculated by Equation 3.1. Such priority listing has been found to work well in many applications but if interface construction is required it can lead to the interpenetration of different materials (Benson, 2002).

Dynamic priority ordering of the fluxes is also possible on a cell-by-cell basis. For example, when using operator splitting the materials may be ordered by decreasing volume fraction at the downwind node (McGlaun *et al.*, 1989). Alternatively, as an operator split algorithm is one-dimensional the possible configurations of the interfaces may be categorised and prioritised depending on the values in the upwind and downwind cells and thus the proximity of the interface to the current face. For instance Bell & Hertel (1992) recognised seven possible configurations of an interface and established priorities depending on which configuration a volume fraction belonged to. A more sophisticated multi-dimensional approach uses the centre of mass of the material, calculated using information from all the neighbouring cells, to decide which interface is closest to a face and should therefore be advected first (Benson, 1998).

Beyond the onion skin model, Choi & Bussmann (2007) suggested a method of calculating triple point junctions (where three materials meet at a point) when using the Y-VOF (Youngs, 1982) material advection scheme. This overcame the problem of intersecting angled interfaces and was generalised by Schofield *et al.* (2008) to more materials by using the material centre of mass and a Voronoi diagram. Moment of fluid (MOF) is an extension to the VOF material tracking algorithms which tracks both the volume of the material and the material centroid, using the extra information to more accurately reproduce the interface (Ahn & Shashkov, 2009). As the centre of mass is available to establish priority ordering MOF has also been generalised to truly multi-material flows (Schofield *et al.*, 2009).

On unstructured meshes interface reconstruction techniques are not considered feasible. Instead, in this chapter a new flux limiter is designed to cope with the issue of material advection for arbitrary numbers of volume fractions. This is built on the design of the total variation bounded advection scheme, UltraC, developed in Chapter 3. As operator splitting is also not possible, the use of this coupled material advection algorithm is demonstrated using a pre-specified priority ordering. It is then used to reproduce some experimental results modelling the generation of waves by landslides.

5.1 Coupled Flux Limiting

Using a flux limited approach to material advection, the issues associated with advecting multiple materials are easily demonstrated. In Figure 5.1 two volume fractions, α^1 and

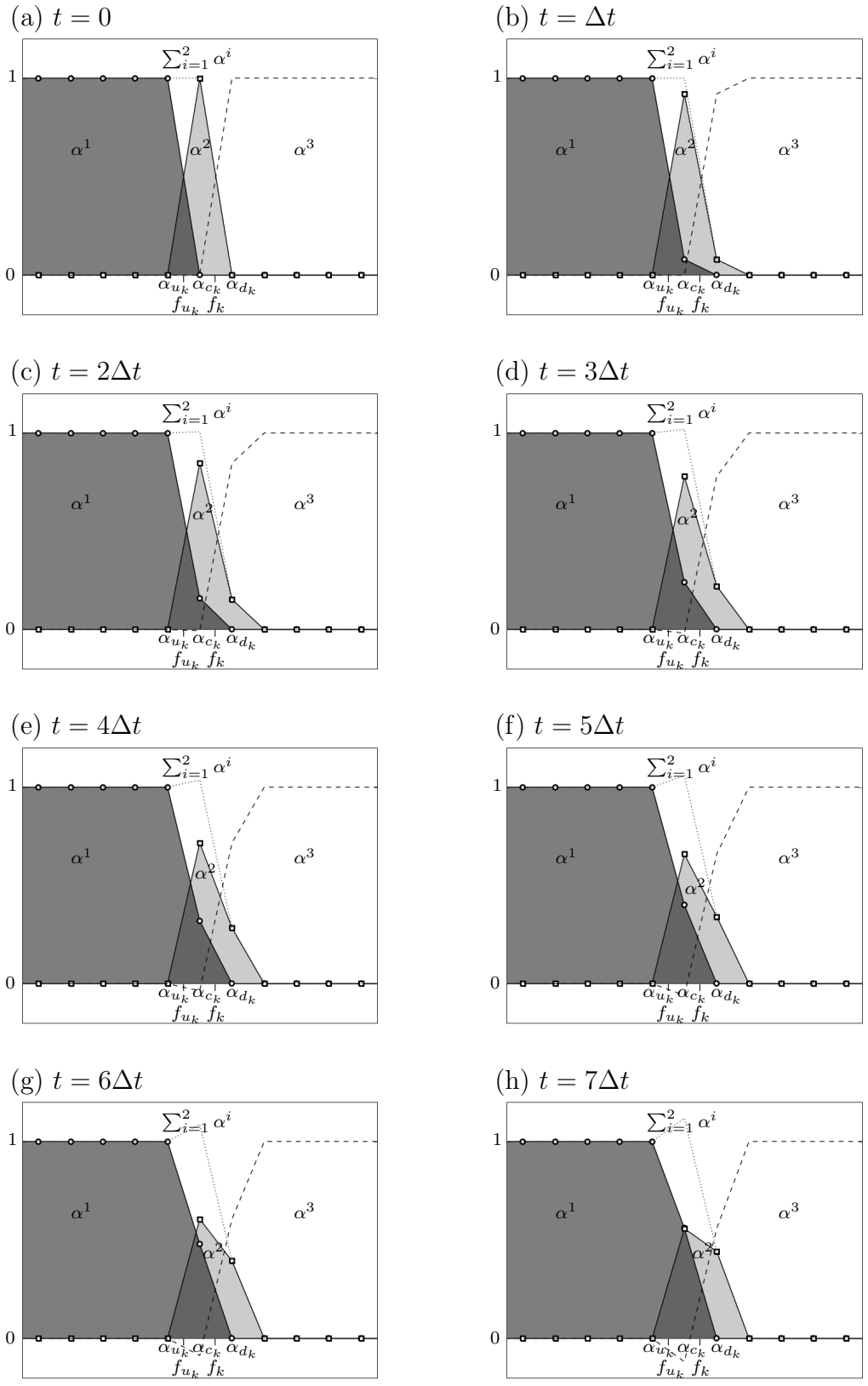


Figure 5.1: *See next page.*

Figure 5.1: *cont.* Eight timesteps from the advection of two volume fractions, α^1 (solid line, circles, dark grey) and α^2 (solid line, squares, light grey), using HyperC at a constant Courant number of 0.08 with $\tilde{u} > 0$. The sum of α^1 and α^2 is also shown (dotted line) as well as the volume fraction of the third material, α_3 (dashed line), which is found by Equation 3.1.

α^2, are advected to the right at a Courant number of 0.08 using Equation 3.24 and the face value scheme HyperC. Consider the fluxes in and out of the cell marked α_{c_k}, the donor cell to the face f_k. At the first timestep (Figure 5.1(a)) α^1 is upwinded through the face f_{u_k} into α_{c_k} and α^2 is upwinded through f_k out of α_{c_k}. As the initial values in each donor node are the same, this produces the same flux in and out of the cell α_{c_k} and the value of the sum of the two volume fractions, $\sum_{i=1}^{2} \alpha^i$, remains the same at that node.

On the second timestep (Figure 5.1(b)) the flux into the cell α_{c_k} through f_{u_k} of α^1 does not match the flux out of the cell through f_k of α^2. Both fields are still upwinding through the respective faces but their donor values are no longer the same. While, for each field individually this results in bounded behaviour, the sum of the volume fractions, $\sum_{i=1}^{2} \alpha^i$, is unbounded leading to non-physical negative values in α^3 through Equation 3.1. The error is initially small (Figure 5.1(c)) but as each field continues to upwind the discrepancy between their fluxes increases leading to clear undershoots in α^3 (Figure 5.1(d–h)).

From Figure 5.1 it is clearly necessary to somehow adjust the fluxes of α^1 and α^2 so that their sum, $\sum_{i=1}^{2} \alpha^i$, remains bounded. This can be thought of as either forcing more of α^2 to be advected through f_k or adjusting the flux of α^1 through f_k so it can advect past α^2 in Figure 5.1(b). In the first scenario α^1 would have the priority and it would, in some sense, push α^2 out of the way. In the second scenario α^2 has priority and refuses to get out of the way for α^1, which must then squeeze past it.

Generalising to arbitrary numbers of materials a summation field, α^{Σ_I}, is defined as:

$$\alpha^{\Sigma_I} = \sum_{i=1}^{I} \alpha^i, \tag{5.1}$$

so that, in Figure 5.1, $\alpha^{\Sigma_2} = \sum_{i=1}^{2} \alpha^i$. Ordering the volume fractions in terms of priority, α^1 is always the highest priority field while α^2 is the second highest priority and so on until α^K, which is the lowest priority 'background' material. α^K can be found using Equation 3.1.

In the onion skin approach to advecting multiple materials the problem is always treated as if it consisted of two materials. Splitting up the summation in Equation 5.1:

$$\underbrace{\alpha^{\Sigma_I}}_{\text{parent sum}} = \underbrace{\alpha^{\Sigma_{I-1}}}_{\text{daughter sum}} + \underbrace{\alpha^I}_{\text{sibling field}}, \tag{5.2}$$

the two materials are then a daughter summation, $\alpha^{\Sigma_{I-1}}$, of all the lower priority volume fractions and its sibling, the field currently being considered, α^I.

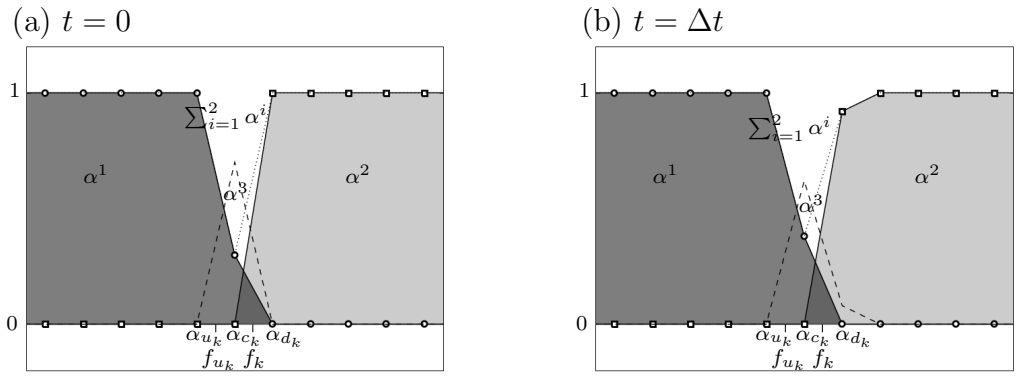


Figure 5.2: Two timesteps from the advection of two volume fractions, α^1 (solid line, circles, dark grey) and α^2 (solid line, squares, light grey), using HyperC at a constant Courant number of 0.08 with $\tilde{u} > 0$. The sum of α^1 and α^2 is also shown (dotted line) as well as the volume fraction of the third material, α_3 (dashed line), which is found by Equation 3.1.

The face value for the sibling field, α^I, must be chosen so that the parent summation, $\alpha^{\sum_I}$, remains, in some sense, bounded. This can be seen by considering the highest priority material summation:

$$\underbrace{\alpha^{\sum_1}}_{\text{parent sum}} = \underbrace{\alpha^1}_{\text{sibling field}} . \tag{5.3}$$

No daughter summation exists because there are no higher priority fields. The parent summation is therefore bounded automatically so long as α^1 is bounded. Moving onto the next field:

$$\underbrace{\alpha^{\sum_2}}_{\text{parent sum}} = \underbrace{\alpha^{\sum_1}}_{\text{daughter sum}} + \underbrace{\alpha^2}_{\text{sibling field}} . \tag{5.4}$$

The daughter summation is known from the advection of the highest priority field, α^1. Therefore, the only way to ensure boundedness of $\alpha^{\sum_2}$ is by adjusting the fluxes of α^2, if necessary. This continues until field $K - 1$:

$$\underbrace{\alpha^{\sum_{K-1}}}_{\text{parent sum}} = \underbrace{\alpha^{\sum_{K-2}}}_{\text{daughter sum}} + \underbrace{\alpha^{K-1}}_{\text{sibling field}} , \tag{5.5}$$

where the fluxes of α^{K-1} are used to ensure the boundedness of $\alpha^{\sum_{K-1}}$. This in turn guarantees the boundedness of α^K.

When considering the material represented by the volume fraction α^I it is necessary to consider both the boundedness requirements of the field itself and those of its parent summation, $\alpha^{\sum_I}$. In order to bound the parent summation however, it is necessary to have a suitable definition of boundedness. In Chapter 3 two definitions were considered: a total variation diminishing (TVD) criterion and a total variation bounded (TVB) scheme.

Considering Figure 5.2 it becomes clear that a TVD criterion is unsuitable for limiting the parent summation. As in Figure 5.1, three materials are being advected to the right

by two volume fractions at a Courant number of 0.08. In this case however, using HyperC does not fail since the prognostic volume fractions, α^1 and α^2, are separated by the lowest priority material, α^3. However, it can also be seen that α^3 does not move as it would had it been advected using HyperC itself. Examining the conditions around the face, f_k, HyperC would consider α^3 to be a local maximum and would therefore enforce an upwinded face value $\hat{\alpha}^3|_{f_k}^{HO} = \alpha_{c_k}^3$. In Figure 5.2 the effective flux of α^3, although never explicitly calculated, is controlled by the amount of 'room' α^2 vacates in α_{d_k}. As α^2 is upwinding (which is the same as downwinding in this case) through the face f_{k+1}, the effective face value through the face f_k for α^3 is $\hat{\alpha}^3|_{f_k}^{HO} = \alpha_{d_k}^2$. As $\alpha_{d_k}^2 > \alpha_{c_k}^3$ the effective flux is higher than it would be using HyperC, a TVD scheme. The boundedness of the parent sum, $\alpha^{\sum_2}$, is directly linked to the boundedness of α^3. Therefore, had the face value of α^2 been adjusted to satisfy a TVD definition for the parent sum, $\alpha^{\sum_2}$, the flux would have been reduced. This would, in turn, have reproduced a situation similar to that seen in Figure 5.1 and α^2 would have become unbounded. It is therefore not generally possible to bound all the volume fractions so that they maintain a total variation diminishing property.

Figure 5.2 shows that it is possible to preserve 'boundedness' of all the volume fractions. The definition of boundedness however must be total variation bounded. If α^3 is considered to have been originally bounded by $[0, 1]$, then using a TVB scheme in the vicinity of f_k it is not an extremum as $\alpha_{c_k}^3$ still lies between these bounds. Therefore, the sensor variable, $\bar{\alpha}_{c_k}^3$ (using normalised variables), still falls within the monotone region and upwinding is not necessary.

In Chapter 3 a TVB flux limiting advection scheme, UltraC, was developed by introducing target upwind and downwind values. By considering the downwind and upwind values of the parent sum, $\alpha^{\sum_I}$, around the face f_k, these can be defined as:

$$\alpha_{tu_k}^{\sum_I} = \max\left(\alpha_0^{\sum_I}\right) \qquad \text{if } \alpha_{u_k}^{\sum_I} > \alpha_{d_k}^{\sum_I} \tag{5.6a}$$

$$\alpha_{tu_k}^{\sum_I} = \min\left(\alpha_0^{\sum_I}\right) \qquad \text{if } \alpha_{d_k}^{\sum_I} > \alpha_{u_k}^{\sum_I} \tag{5.6b}$$

and:

$$\alpha_{td_k}^{\sum_I} = \min\left(\alpha_0^{\sum_I}\right) \qquad \text{if } \alpha_{u_k}^{\sum_I} > \alpha_{d_k}^{\sum_I} \tag{5.7a}$$

$$\alpha_{td_k}^{\sum_I} = \max\left(\alpha_0^{\sum_I}\right) \qquad \text{if } \alpha_{d_k}^{\sum_I} > \alpha_{u_k}^{\sum_I} \tag{5.7b}$$

respectively. Generally for volume fractions the maximum value is taken to be 1 while the minimum is 0.

Using Equations 5.6 and 5.7, the donor parent sum value to face f_k, $\alpha_{c_k}^{\sum_I}$, can be considered as smooth if:

$$\alpha_{tu_k}^{\sum_I} > \alpha_{c_k}^{\sum_I} > \alpha_{td_k}^{\sum_I}, \tag{5.8a}$$

or:

$$\alpha_{td_k}^{\sum_I} > \alpha_{c_k}^{\sum_I} > \alpha_{tu_k}^{\sum_I}. \tag{5.8b}$$

As the sibling field α^I is being advected rather than the parent sum it is useful to restate Equations 5.8 as:

$$\alpha_{tu_k}^{\sum_I} - \alpha_{c_k}^{\sum_{I-1}} > \alpha_{c_k}^I > \alpha_{td_k}^{\sum_I} - \alpha_{c_k}^{\sum_{I-1}}, \tag{5.9a}$$

and:

$$\alpha_{td_k}^{\sum_I} - \alpha_{c_k}^{\sum_{I-1}} > \alpha_{c_k}^I > \alpha_{tu_k}^{\sum_I} - \alpha_{c_k}^{\sum_{I-1}}. \tag{5.9b}$$

Normalising with respect to α^I, such that:

$$\bar{\alpha} = \frac{\alpha - \alpha_{u_k}^I}{\alpha_{d_k}^I - \alpha_{u_k}^I}, \tag{5.10}$$

Equations 5.8 can be described using normalised variables as:

$$\overline{\alpha_{td_k}^{\sum_I} - \alpha_{u_k}^I} - \bar{\alpha}_{c_k}^{\sum_{I-1}} < \bar{\alpha}_{c_k}^I < \overline{\alpha_{tu_k}^{\sum_I} - \alpha_{u_k}^I} - \bar{\alpha}_{c_k}^{\sum_{I-1}}, \tag{5.11a}$$

and:

$$\overline{\alpha_{tu_k}^{\sum_I} - \alpha_{u_k}^I} - \bar{\alpha}_{c_k}^{\sum_{I-1}} < \bar{\alpha}_{c_k}^I < \overline{\alpha_{td_k}^{\sum_I} - \alpha_{u_k}^I} - \bar{\alpha}_{c_k}^{\sum_{I-1}}. \tag{5.11b}$$

In Chapter 3, the smoothness criteria (Equation 3.7 for example) collapsed to a single equation after being normalised (Equation 3.11 for example). Here the two conditions in Equation 5.8 fail to collapse and two conditions remain in Equation 5.11. This occurs because the slopes of the field being normalised, the parent sum $\alpha^{\sum_I}$, potentially differ from the field doing the normalising, the sibling field α^I. In other words there is a condition for when $\text{sign}\left(\alpha_{d_k}^{\sum_I} - \alpha_{u_k}^{\sum_I}\right) = \text{sign}\left(\alpha_{d_k}^I - \alpha_{u_k}^I\right)$ and one for when $\text{sign}\left(\alpha_{d_k}^{\sum_I} - \alpha_{u_k}^{\sum_I}\right) \neq \text{sign}\left(\alpha_{d_k}^I - \alpha_{u_k}^I\right)$. Overall, the region where the parent sum is considered smooth may be written:

$$\min\left(\begin{array}{c}\overline{\alpha_{td_k}^{\sum_I} - \alpha_{u_k}^I} - \bar{\alpha}_{c_k}^{\sum_{I-1}} \\ \overline{\alpha_{tu_k}^{\sum_I} - \alpha_{u_k}^I} - \bar{\alpha}_{c_k}^{\sum_{I-1}}\end{array}\right) < \bar{\alpha}_{c_k}^I < \max\left(\begin{array}{c}\overline{\alpha_{td_k}^{\sum_I} - \alpha_{u_k}^I} - \bar{\alpha}_{c_k}^{\sum_{I-1}} \\ \overline{\alpha_{tu_k}^{\sum_I} - \alpha_{u_k}^I} - \bar{\alpha}_{c_k}^{\sum_{I-1}}\end{array}\right). \tag{5.12}$$

Outside the region defined by Equation 5.12 the parent sum field is already unbounded. Hence, the equivalent of first-order upwinding must be enforced on the parent sum:

$$\hat{\alpha}_{f_k}^{\sum_I} = \alpha_{c_k}^{\sum_I}. \tag{5.13}$$

However, $\alpha^{\sum_I}$ is not being advected so this must instead be enforced through the sibling field. Rewriting Equation 5.13 in terms of α^I and normalising by Equation 5.10, the appropriate face value may be found from:

$$\bar{\hat{\alpha}}_{f_k}^I = \bar{\alpha}_{c_k}^{\sum_{I-1}} + \bar{\alpha}_{c_k}^I - \bar{\alpha}_{f_k}^{\sum_{I-1}}. \tag{5.14}$$

Inside the smooth region defined by Equation 5.12 there is much more scope for assigning different face values that will maintain a TVB definition of boundedness in the parent sum. Unlike when HyperC and UltraC were designed, the aim here is not to

enforce the most compressive or minimally diffusive values on the system. Instead, given the daughter sum of the higher priority fields, $\alpha^{\sum_{I-1}}$, a scheme is required which decides whether a normalised face value, $\bar{\alpha}^I_{f_k}$, will cause $\alpha^{\sum_I}$ to become unbounded or not. The face value can be assigned by any advection algorithm, such as HyperC or UltraC, independently of the check performed here. Therefore, it is the advection algorithm that attempts to control the properties of the advection. If the face value is found to maintain boundedness in the parent sum then it will be left unchanged. If, however, the face value is found to violate the boundedness criterion then it must be adjusted or limited somehow to prevent unbounded behaviour. In this sense the scheme being developed here will be used as a limiting function, such as those discussed in Section 3.2.3, rather than as a face value algorithm.

In Chapter 3, HyperC was described as the upper boundary on the explicit convective boundedness criterion for a TVD scheme. For a TVB scheme the equivalent upper boundary is UltraC. (The lower boundary in each case is first-order upwinding.) The upper boundary of the limiting function will therefore be defined by targeted downwinding and a Courant number controlled flux.

Considering downwinding first, the equivalent face value of the parent sum becomes:

$$\hat{\alpha}^{\sum_I}_{f_k} = \alpha^{\sum_I}_{d_k}. \tag{5.15}$$

In terms of the normalised sibling field, this becomes:

$$\hat{\bar{\alpha}}^I_{f_k} = \overline{\alpha^{\sum_I}_{td_k} - \alpha^I_{u_k}} - \bar{\alpha}^{\sum_{I-1}}_{f_k}. \tag{5.16}$$

As with UltraC, using this face value is considered safe when:

$$\alpha^{\sum_I}_{c_k} + c_{c_{f_k}}\alpha^{\sum_I}_{tu_k} - c_{c_{f_k}}\alpha^{\sum_I}_{td_k} \leq \alpha^{\sum_I}_{tu_k}, \quad \text{if } \alpha^{\sum_I}_{u_k} > \alpha^{\sum_I}_{d_k}, \tag{5.17a}$$

or:

$$\alpha^{\sum_I}_{c_k} + c_{c_{f_k}}\alpha^{\sum_I}_{tu_k} - c_{c_{f_k}}\alpha^{\sum_I}_{td_k} \geq \alpha^{\sum_I}_{tu_k}, \quad \text{if } \alpha^{\sum_I}_{d_k} > \alpha^{\sum_I}_{u_k}. \tag{5.17b}$$

In combination with the smoothness criterion (Equation 5.12) these conditions can be rewritten in terms of the normalised sibling field, becoming:

$$\min\left(\frac{\overline{\alpha^{\sum_I}_{td_k} - \alpha^I_{u_k}} - \bar{\alpha}^{\sum_{I-1}}_{c_k}}{\overline{\alpha^{\sum_I}_{tu_k} - \alpha^I_{u_k}} - \bar{\alpha}^{\sum_{I-1}}_{c_k}}\right) < \bar{\alpha}^I_{c_k} \leq \left(1 - c_{c_{f_k}}\right)\left(\overline{\alpha^{\sum_I}_{tu_k} - \alpha^I_{u_k}}\right) + c_{c_{f_k}}\left(\overline{\alpha^{\sum_I}_{td_k} - \alpha^I_{u_k}}\right) - \bar{\alpha}^{\sum_{I-1}}_{c_k} \tag{5.18a}$$

and:

$$\left(1 - c_{c_{f_k}}\right)\left(\overline{\alpha^{\sum_I}_{tu_k} - \alpha^I_{u_k}}\right) + c_{c_{f_k}}\left(\overline{\alpha^{\sum_I}_{td_k} - \alpha^I_{u_k}}\right) - \bar{\alpha}^{\sum_{I-1}}_{c_k} \leq \bar{\alpha}^I_{c_k} < \max\left(\frac{\overline{\alpha^{\sum_I}_{td_k} - \alpha^I_{u_k}} - \bar{\alpha}^{\sum_{I-1}}_{c_k}}{\overline{\alpha^{\sum_I}_{tu_k} - \alpha^I_{u_k}} - \bar{\alpha}^{\sum_{I-1}}_{c_k}}\right). \tag{5.18b}$$

Again, two criteria are present: Equation 5.18a corresponds to the case when:

$$\text{sign}\left(\alpha^{\sum_I}_{d_k} - \alpha^{\sum_I}_{u_k}\right) \neq \text{sign}\left(\alpha^I_{d_k} - \alpha^I_{u_k}\right),$$

while Equation 5.18b is valid when:

$$\text{sign}\left(\alpha_{d_k}^{\sum_I} - \alpha_{u_k}^{\sum_I}\right) = \text{sign}\left(\alpha_{d_k}^{I} - \alpha_{u_k}^{I}\right).$$

In the remaining region, when downwinding is not safe and upwinding is not necessary, a Courant number limited face value defines the upper boundary to the limiting function. As in UltraC (Equation 3.85) this is given by:

$$\hat{\alpha}_{f_k}^{\sum_I} = \frac{1}{c_{c_{f_k}}}\left(\alpha_{c_k}^{\sum_I} + \alpha_{tu_k}^{\sum_I}\right) + \alpha_{tu_k}^{\sum_I}, \tag{5.19}$$

which may be rewritten in terms of the normalised sibling field as:

$$\bar{\hat{\alpha}}_{f_k}^{I} = \frac{1}{c_{c_{f_k}}}\left(\alpha_{c_k}^{I} + \alpha_{c_k}^{\sum_{I-1}} + \left(c_{c_{f_k}} - 1\right)\left(\overline{\alpha_{tu_k}^{\sum_I} - \alpha_{u_k}^{I}}\right) - c_{c_{f_k}}\alpha_{f_k}^{\sum_{I-1}}\right). \tag{5.20}$$

Equations 5.16, 5.18 and 5.20 form a quadrilateral on the normalised variable diagram (shown as a grey shaded area in Figure 5.3). Within this area any face value assigned by the advection algorithm for $\alpha_{f_k}^{I}$ will not cause the parent sum to become unbounded and is therefore left untouched. Outside this region the face value will be limited vertically (along lines of equal normalised donor value, $\alpha_{c_k}^{I}$) until it either sits on the boundary of the quadrilateral or, if it fell outside the bounds of the smoothness criteria (Equation 5.12), on the upwinding line (Equation 5.14). This results in face values of the sibling field that cause unbounded behaviour being reset so that they have the same effect as downwinding, upwinding or Courant limited downwinding the parent sum.

Comparing Figure 5.3 to any of the previous normalised variable diagrams (Figure 3.4(a) for example) several differences can be seen. The upwinding line is offset from the diagonal, around which all other NVD schemes are arranged. This occurs because the coupled limiter is not imposing a certain behaviour on the field being limited, α^I. Instead, it is using α^I to impose boundedness on another field, $\alpha^{\sum_I}$. Hence, the 'upwinding' line in Figure 5.3 does not produce the same behaviour in the field as a standard upwind method would.

Since the range of physical face values lies between the downwind and donor values, the lower boundary on limiters is generally taken as the upwinding line. However, in Figure 5.3 the bounded region extends across the upwinding line to form a parallelogram. As discussed before, this occurs because the field being limited may have a different slope to the field doing the limiting. To distinguish between the configurations the diagram has been shaded blue in the 'normal' NVD region where the slopes are the same, $\text{sign}\left(\alpha_{d_k}^{\sum_I} - \alpha_{u_k}^{\sum_I}\right) = \text{sign}\left(\alpha_{d_k}^{I} - \alpha_{u_k}^{I}\right)$, and yellow where they are different, $\text{sign}\left(\alpha_{d_k}^{\sum_I} - \alpha_{u_k}^{\sum_I}\right) \neq \text{sign}\left(\alpha_{d_k}^{I} - \alpha_{u_k}^{I}\right)$. All the labels that fall within each region are specific to that configuration.

The potentially different slopes of the parent, $\alpha^{\sum_I}$, and sibling, α^I, fields do not add any extra complications to the implementation of the coupled limiter. Examining Figure 5.3, it can be seen that only the maximum and minimum values of the corners and sides of the quadrilateral are important. Hence, no special cases are required to deal with the two slope configurations. Instead the coupled limiter may be implemented as:

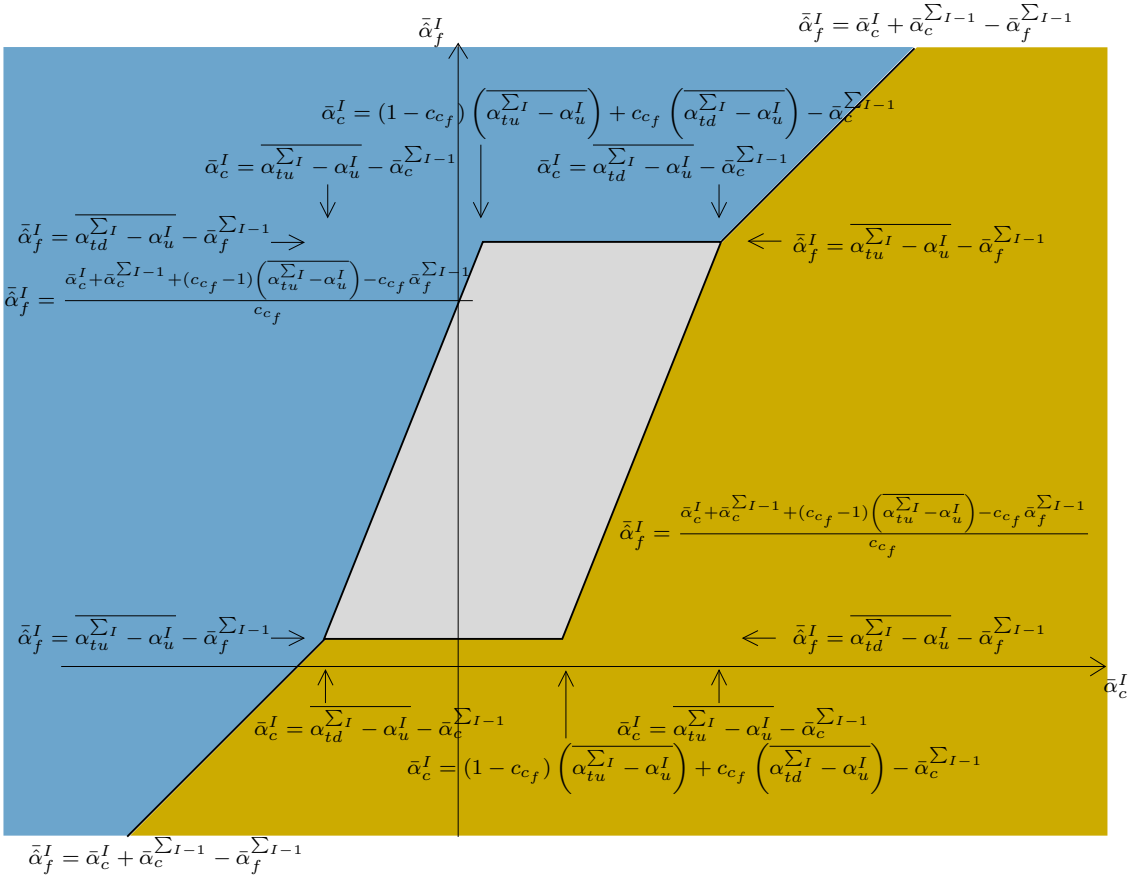


Figure 5.3: The coupled limiter for the field α^I represented by the grey shaded area on a normalised variable diagram (NVD). Labels in the upper left blue region refer to the case when the difference between the parent sum downwind and upwind values has the same sign as the limited field, $\text{sign}\left(\alpha_{d_k}^{\sum_I} - \alpha_{u_k}^{\sum_I}\right) = \text{sign}\left(\alpha_{d_k}^I - \alpha_{u_k}^I\right)$. Similarly, labels in the lower right yellow region refer to the case when the signs of the slopes are opposite, $\text{sign}\left(\alpha_{d_k}^{\sum_I} - \alpha_{u_k}^{\sum_I}\right) \neq \text{sign}\left(\alpha_{d_k}^I - \alpha_{u_k}^I\right)$. The regions are separated by the upwinding line, $\bar{\tilde{\alpha}}_f^I = \bar{\alpha}_{c_k}^I + \bar{\alpha}_{c_k}^{\sum_{I-1}} - \bar{\alpha}_f^{\sum_{I-1}}$.

$$\text{if } \min\left(\begin{array}{c}\overline{\alpha_{td_k}^{\sum_I}-\alpha_{u_k}^I}-\bar{\alpha}_{c_k}^{\sum_{I-1}}\\ \alpha_{tu_k}^{\sum_I}-\alpha_{u_k}^I-\bar{\alpha}_{c_k}^{\sum_{I-1}}\end{array}\right) < \bar{\alpha}_{c_k}^I < \max\left(\begin{array}{c}\overline{\alpha_{td_k}^{\sum_I}-\alpha_{u_k}^I}-\bar{\alpha}_{c_k}^{\sum_{I-1}}\\ \alpha_{tu_k}^{\sum_I}-\alpha_{u_k}^I-\bar{\alpha}_{c_k}^{\sum_{I-1}}\end{array}\right):$$

$$\bar{\tilde{\alpha}}_{f_k}^I = \min\left(\begin{array}{l}\max\left(\begin{array}{c}\overline{\alpha_{td_k}^{\sum_I}-\alpha_{u_k}^I}-\bar{\alpha}_{f_k}^{\sum_{I-1}}\\ \alpha_{tu_k}^{\sum_I}-\alpha_{u_k}^I-\bar{\alpha}_{f_k}^{\sum_{I-1}}\end{array}\right)\\ \max\left(\begin{array}{l}\frac{1}{c_{c_{f_k}}}\left(\alpha_{c_k}^I+\alpha_{c_k}^{\sum_{I-1}}+\left(c_{c_{f_k}}-1\right)\left(\overline{\alpha_{tu_k}^{\sum_I}-\alpha_{u_k}^I}\right)-c_{c_{f_k}}\alpha_{f_k}^{\sum_{I-1}}\right)\\ \frac{1}{c_{c_{f_k}}}\left(\alpha_{c_k}^I+\alpha_{c_k}^{\sum_{I-1}}+\left(c_{c_{f_k}}-1\right)\left(\overline{\alpha_{td_k}^{\sum_I}-\alpha_{u_k}^I}\right)-c_{c_{f_k}}\alpha_{f_k}^{\sum_{I-1}}\right)\end{array}\right)\\ \max\left(\begin{array}{l}\bar{\alpha}_{f_k}^I\\ \min\left(\begin{array}{c}\overline{\alpha_{td_k}^{\sum_I}-\alpha_{u_k}^I}-\bar{\alpha}_{f_k}^{\sum_{I-1}}\\ \alpha_{tu_k}^{\sum_I}-\alpha_{u_k}^I-\bar{\alpha}_{f_k}^{\sum_{I-1}}\end{array}\right)\\ \min\left(\begin{array}{l}\frac{1}{c_{c_{f_k}}}\left(\alpha_{c_k}^I+\alpha_{c_k}^{\sum_{I-1}}+\left(c_{c_{f_k}}-1\right)\left(\overline{\alpha_{tu_k}^{\sum_I}-\alpha_{u_k}^I}\right)-c_{c_{f_k}}\alpha_{f_k}^{\sum_{I-1}}\right)\\ \frac{1}{c_{c_{f_k}}}\left(\alpha_{c_k}^I+\alpha_{c_k}^{\sum_{I-1}}+\left(c_{c_{f_k}}-1\right)\left(\overline{\alpha_{td_k}^{\sum_I}-\alpha_{u_k}^I}\right)-c_{c_{f_k}}\alpha_{f_k}^{\sum_{I-1}}\right)\end{array}\right)\end{array}\right)\end{array}\right)$$

else:

$$\bar{\tilde{\alpha}}_{f_k}^I = \bar{\alpha}_{c_k}^{\sum_{I-1}} + \bar{\alpha}_{c_k}^I - \bar{\alpha}_{f_k}^{\sum_{I-1}}.$$

UltraC was shown in Chapter 3 to produce non-physical behaviour when advecting isolated sections of a volume fraction. These erroneous advection velocities were caused by the choice of the target upwind and target downwind values being poorly defined numerically when the slope of the actual upwind and downwind values approached each other. In the coupled limiter, target downwind, $\alpha_{td_k}^{\sum_I}$, and target upwind, $\alpha_{tu_k}^{\sum_I}$, values are also assigned to the parent sum. However, the algorithm above demonstrates that their absolute values do not matter. Only the maximum and minimum of the targets is included in the limiting process. Hence, no ambiguity is introduced in their definition.

In addition to the target value definitions the coupled limiter also introduces a dependency on the values of the daughter summation, $\alpha^{\sum_{I-1}}$. This not only includes the nodal sum values but also the face values, $\alpha_{f_k}^{\sum_{I-1}}$. To avoid having to store the face values for the entire domain, the coupled limiter was implemented so that all the advection equations for the volume fractions are assembled simultaneously. Therefore it is only necessary to maintain a running total of the daughter summation on the current face while looping over the materials. As a significant cost of assembly is the calculation of the integral transformations on the control volume faces, this implementation is also more efficient. The Jacobian of the transformation need only be calculated once per face for all the volume fractions.

Returning to the unbounded behaviour seen in Figure 5.1, Figure 5.4 shows the effect of applying the coupled limiter (Figure 5.3) to the same configuration of volume fractions. For each prognostic field, α^1 and α^2, HyperC is used to initially assign a face value for

Figure 5.4: *See next page.* Eight timesteps from the advection of two volume fractions, α^1 (solid line, circles, dark grey) and α^2 (solid line, squares, light grey), using HyperC and the coupled flux limiter at a constant Courant number of 0.08 with $\tilde{u} > 0$. The sum of α^1 and α^2 is also shown (dotted line) as well as the volume fraction of the third material, α_3 (dashed line), which is found by Equation 3.1.

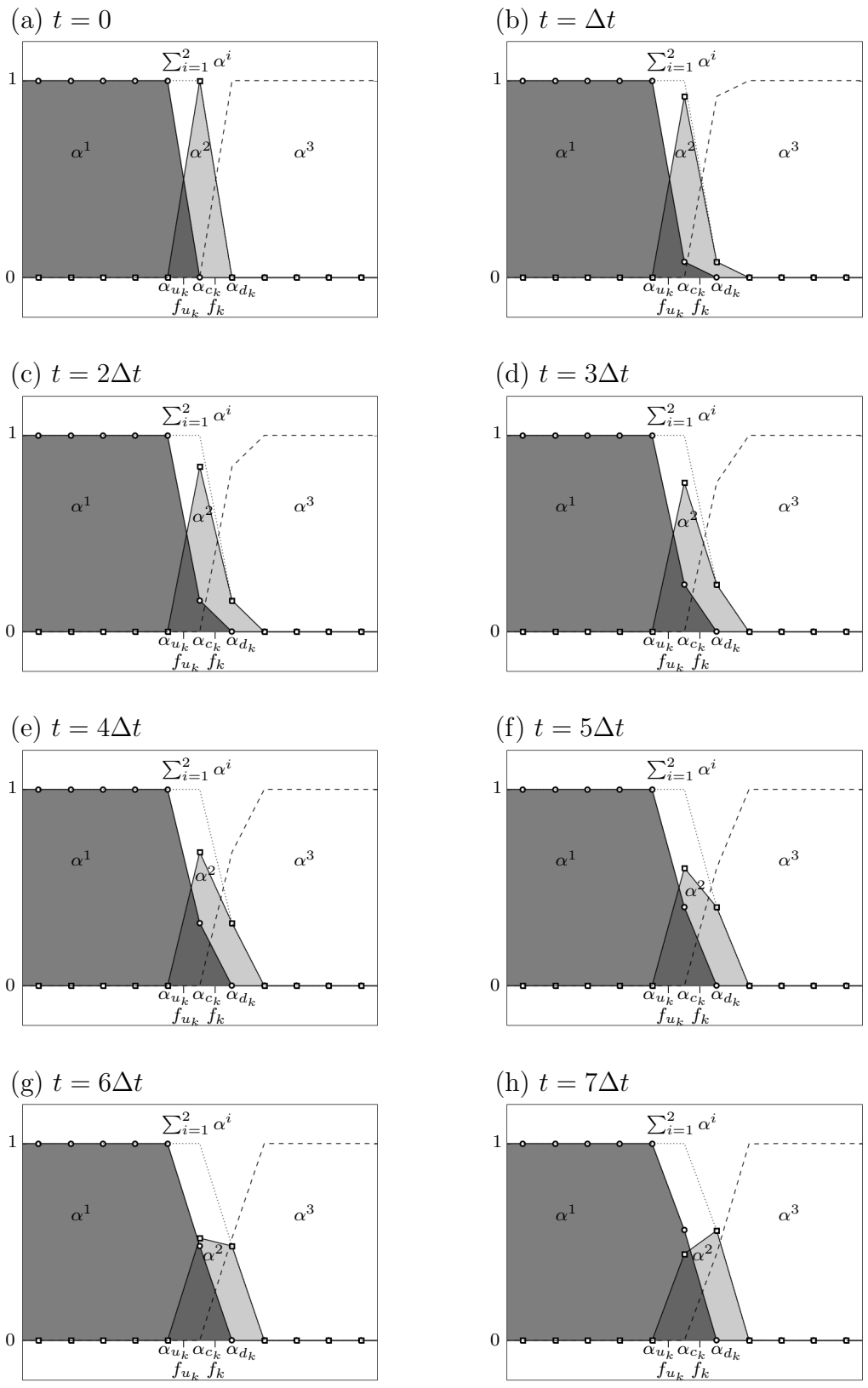


Figure 5.4: *cont.*

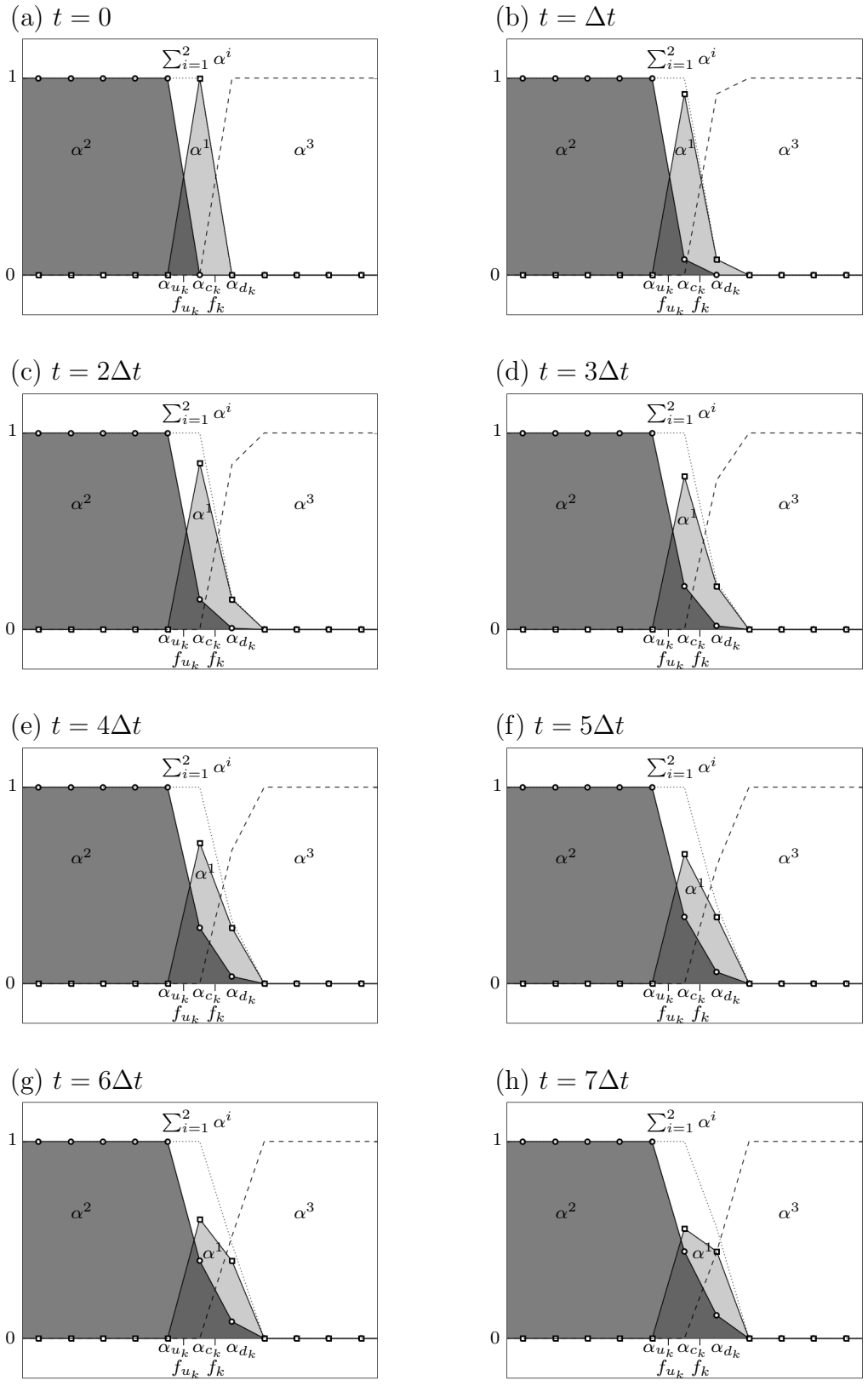


Figure 5.5: *See next page.*

Figure 5.5: *cont.* Eight timesteps from the advection of two volume fractions, α^1 (solid line, squares, light grey) and α^2 (solid line, circles, dark grey), using HyperC and the coupled flux limiter at a constant Courant number of 0.08 with $\tilde{u} > 0$. The priority ordering of the prognostic fields is the opposite of that in Figure 5.4. The sum of α^1 and α^2 is also shown (dotted line) as well as the volume fraction of the third material, α_3 (dashed line), which is found by Equation 3.1.

the face f_k. At the first timestep (Figure 5.4(a)) this leads to upwinding for α^2 through f_k and for α^1 through f_{u_k}. As this resulted in bounded behaviour in Figure 5.1 no change is made by applying the coupled limiter. However, at the second timestep (Figure 5.1(b, c)) continued upwinding resulted in an unbounded parent sum, α^{Σ_2}. In Figure 5.4(b, c) this does not occur. As the highest priority field the solution for α^1 remains the same and it still upwinds through f_{u_k}. However, the coupled limiter increases the flux of α^2 out of the cell to ensure that the sum of the two volume fractions does not exceed one (see dotted line in Figure 5.4(b, c)). This behaviour continues with α^1 upwinding through f_{u_k} and the limiter increasing the flux of α^2 to ensure boundedness of the coupled system (Figures 5.4(d–h)). Effectively the higher priority field, α^1, is pushing α^2 out of the way.

It is also possible to recreate the unbounded scenario in Figure 5.1 with the reverse priority ordering. This is shown in Figure 5.5. Again, the first timestep (Figure 5.5(a)) is identical. However, at the second timestep (Figure 5.5(b, c)) the lower flux of α^1 through f_k cannot be adjusted since it has priority. In this scenario the flux of α^2 through f_{u_k} cannot be diminished without either having a knock-on effect throughout the whole block of material or violating that volume fraction's own boundedness criterion. Therefore, the coupled limiter prevents α^2 from downwinding at f_k, which is what HyperC told it to do, and instead propagates sufficient material to prevent α^{Σ_2} from becoming unbounded. In a sense α^2 is locally squeezing past α^1. The effect of this can be seen in Figure 5.5(d–h) as the interface of α^2 becomes smeared over the same two cells α^1 is using to advect. This smearing remains limited to where the two materials share cells.

5.1.1 Multi-Dimensional Advection Tests

Just as HyperC and UltraC were extended to multiple dimensions, it is possible to apply the coupled limiter on the control volume dual mesh. As in one dimension, after each face value is assigned by the advection algorithm it is limited to ensure boundedness of its parent sum. The limiter is applied in the same form as Figure 5.3 with the definition of the Courant number taken from Equation 3.59, as in multi-dimensional HyperC and UltraC. To demonstrate this, and to investigate the properties of the coupled limiter in multiple dimensions on a larger number of materials, the shearing flow test case from Chapter 3 is repeated for multiple materials.

In the original shearing flow test case a circle of initial radius $\pi/5$ was centered at $(\pi/2, (1+\pi)/5)$ in the domain, Ω: $x_1 \in [0, \pi]$, $x_2 \in [0, \pi]$ and subjected to the prescribed

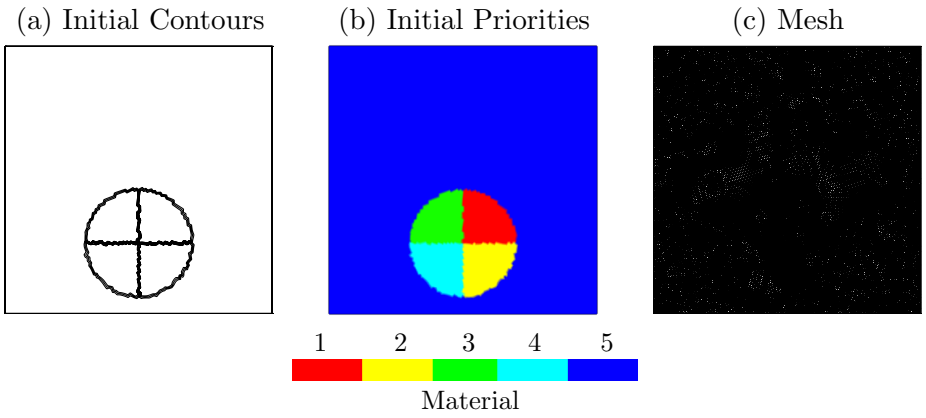


Figure 5.6: The initial condition (a), bulk density showing the material priority indices (b), and unstructured parent mesh (c) used in the multiple material shearing flow advection tests.

velocity field $\tilde{u} = (\sin(x_1)\cos(x_2), -\cos(x_1)\sin(x_2))$. After $N = 250$, 500, 1,000, 2,000, 3,000 and 4,000 timesteps at a Courant number of 0.25 the velocity field was reversed, $\tilde{u} = -(\sin(x_1)\cos(x_2), \cos(x_1)\sin(x_2))$, and after an equal number of timesteps the final result compared to the initial condition. Following Choi & Bussmann (2007) and Schofield *et al.* (2009) the initial circle is modified to contain more than one material. It is divided between four materials, one in each quarter, separated along lines aligned with the coordinate axes. These four materials are advected using HyperC and the coupled limiter. The surrounding background material is found using Equation 3.1.

Fixed Mesh

The multiple material shearing test case was performed on a fixed unstructured mesh with an average parent edge length of $\pi/100$, equivalent to the structured mesh used by Rudman (1997). The initial condition on this parent mesh is presented in two forms in Figure 5.6. Figure 5.6(a) shows the contours surrounding each prognostic volume fraction at 0.025, 0.5 and 0.975. Figure 5.6(b) displays the bulk density, calculated using

Figure 5.7: *See next page.* Results for the forward advection of Figure 5.6(a) in a shearing flow, $\tilde{u} = (\sin(x_1)\cos(x_2), -\cos(x_1)\sin(x_2))$, after $N = 250$ (i), 500 (ii), 1,000 (iii), 2,000 (iv), 3,000 (v) and 4,000 (vi) timesteps. Contours of the numerical solution using HyperC and the coupled limiter (a) and the 'exact' solution, found by numerically integrating the characteristic equations to a tolerance of 10^{-12} (b), are shown at values of 0.025, 0.5 and 0.975 for all the volume fractions. (c) The absolute difference, $|\alpha^I - \alpha^{I^e}|$, between the numerical solution (a) and the 'exact' solution (b). Differences greater than the 0.025 contour are shaded in grey for all volume fractions.

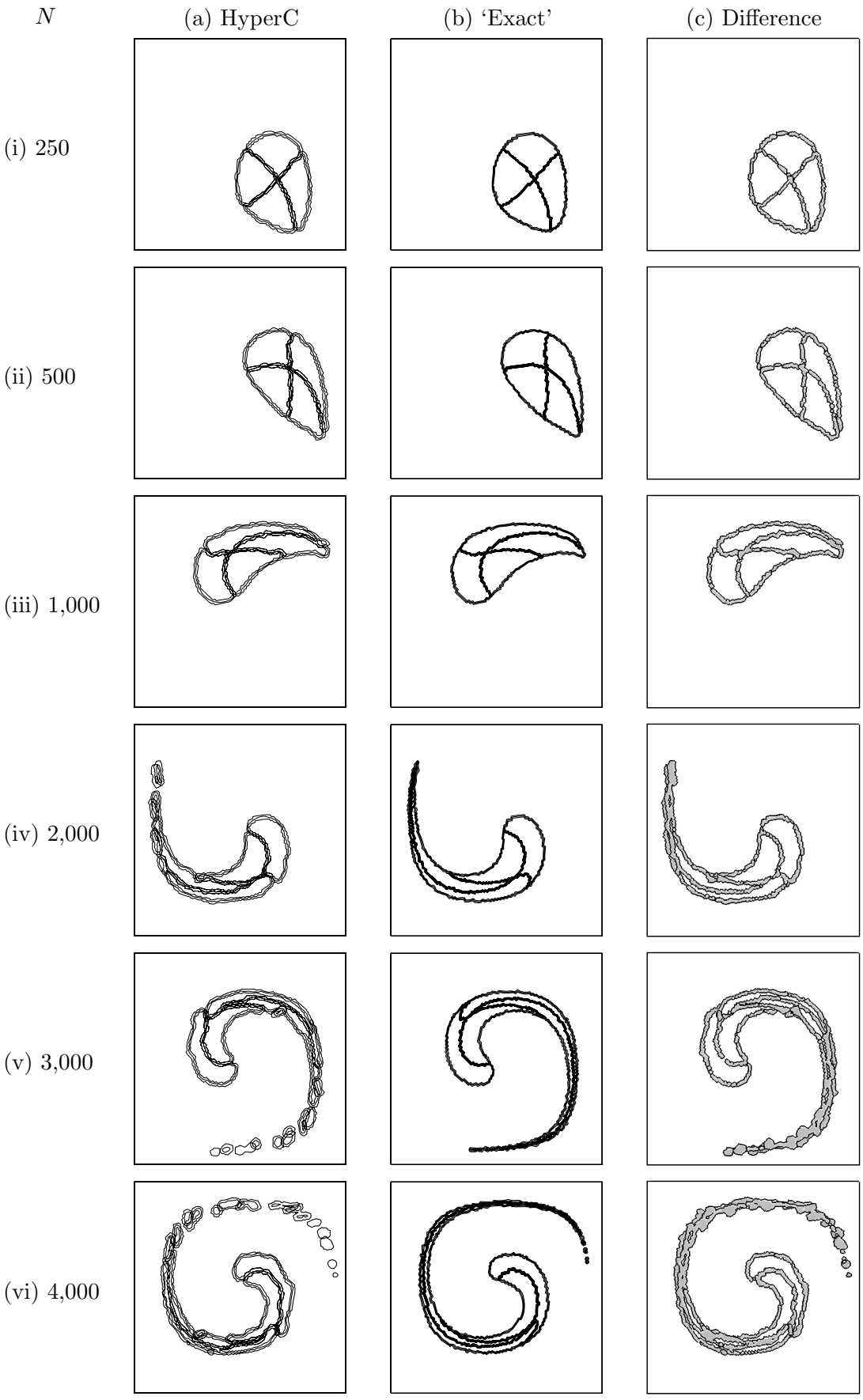


Figure 5.7: *cont.*

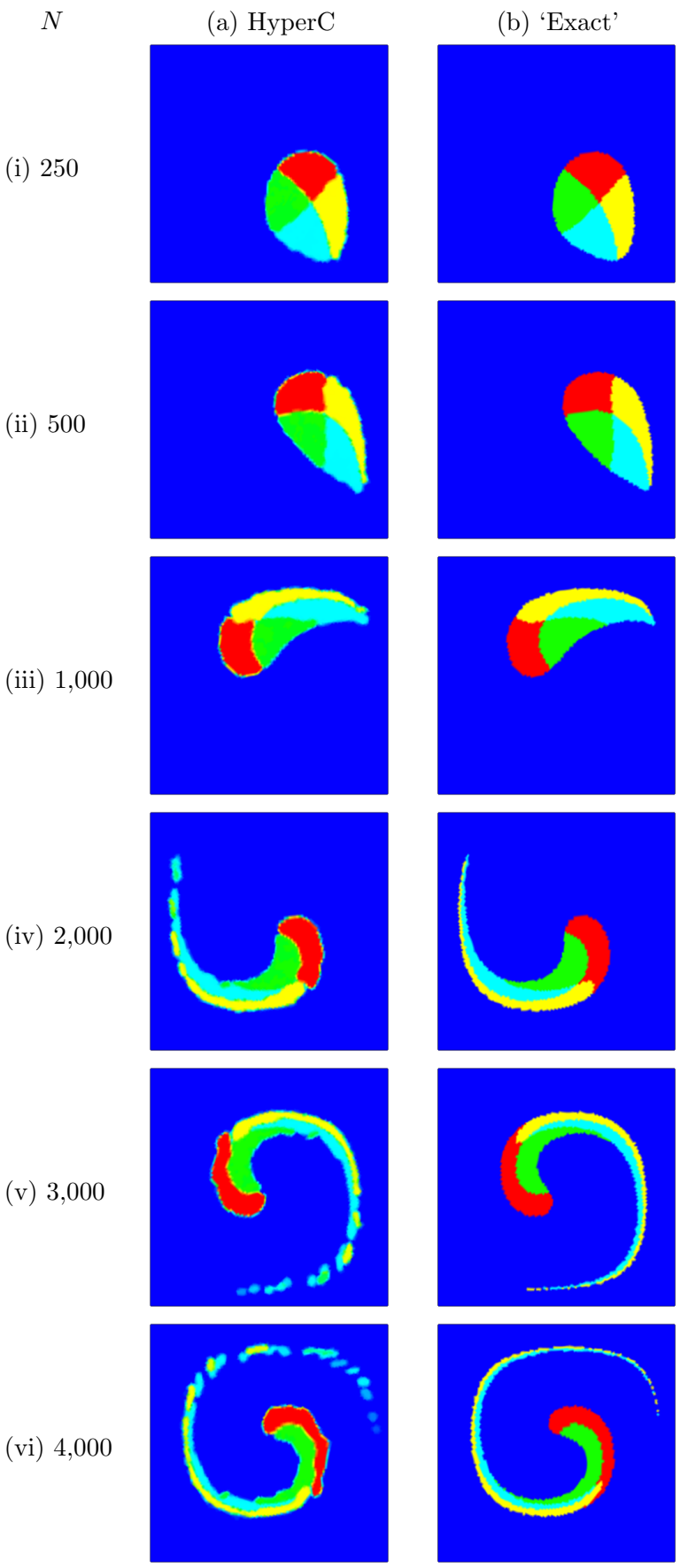


Figure 5.8: *See next page.*

Figure 5.8: *cont.* Same as in Figure 5.7 but showing the bulk density by Equation 3.2, where the density of each material is its priority number (scale shown in Figure 5.6). Numerical results using HyperC and the coupled limiter (a) are shown alongside the 'exact' solution (b).

Equation 3.2, where the density of each material has been assigned as its priority index from 1 to 5.

The results of advecting the four material circle forward in a shearing velocity field are shown in Figures 5.7(a) and 5.8(a) after $N = 250$, 500, 1,000, 2,000, 3,000 and 4,000 timesteps at a Courant number of 0.25. All the volume fractions, including that of the background material, α^5, remained bounded by $[0, 1]$ showing that no non-physical overlapping of the materials occurred. Furthermore, after the velocity field was projected to be discretely divergence free on the control volume mesh, using Equations 4.13c and 4.14, all the materials were conserved with a maximum absolute conservation error, $\max\left(|\int_\Omega \alpha^i - \alpha_0 \mathrm{d}V| / \int_\Omega \alpha_0 \mathrm{d}V\right), i = 1, ..., 5$, of 3×10^{-11}.

The results from the corresponding two material case can be found in Figure 3.30(b). The multiple material and two material cases were run at the same resolution and Courant number. Differences can be seen at the outer boundary of the sheared circle. In the multiple material case the boundary is not as smooth as in the two material case and notches occur where three or more materials meet. In these areas HyperC is unconstrained by the coupled limiter so the added deformation seen is a result of that section of interface being advected separately from the remainder of the boundary. Internally, the coupled limiter maintains the relationships between the materials and they are advected contiguously. This continuity breaks down after $N = 2,000$ in the multiple material case when the sheared tail generates isolated, low volume fraction, patches. Similar patches are seen in the two material case but only after $N = 3,000$ and they do not become as widely distributed. The difference arises because the circle is being sheared as four individual materials. As the resolution in each case is the same, each material in the multiple material case is being represented by a quarter of the number of cells used for the circle in the two material case. Thus, as the tail is sheared, the limit of the resolution is approached faster for each material, leading to increased shedding.

For comparison the 'exact' solutions, found by numerically integrating the characteristic equations to a tolerance of 10^{-12}, are shown in Figures 5.7(b) and 5.8(b). Using these as α^{I^e} for each volume fraction the absolute difference, $|\alpha^I - \alpha^{I^e}|$ (see Figure 5.7(c)), and error by Equation 3.65 may be found. In Table 5.1, the materials towards the outside of the domain are seen to accumulate error faster than those towards the inside. Due to the increased shearing they experience these approach the available resolution faster and hence start to shed isolated volume fractions earlier. Despite this the accumulated error for the sum of the four prognostic materials is still comparable to the two material case during forward advection (see Table 5.2).

The results applying the reverse shear velocity to Figures 5.7(a, b)(i–vi) and 5.8(a)(i–

N	Material 1 Error	Material 2 Error	Material 3 Error	Material 4 Error
250	6.23×10^{-2}	8.02×10^{-2}	5.10×10^{-2}	6.77×10^{-2}
500	7.00×10^{-2}	1.03×10^{-1}	6.60×10^{-2}	8.34×10^{-2}
1,000	7.97×10^{-2}	1.53×10^{-1}	8.10×10^{-2}	1.13×10^{-1}
2,000	1.06×10^{-1}	3.09×10^{-1}	1.20×10^{-1}	2.97×10^{-1}
3,000	1.56×10^{-1}	4.67×10^{-1}	1.78×10^{-1}	5.36×10^{-1}
4,000	2.33×10^{-1}	6.13×10^{-1}	2.66×10^{-1}	7.43×10^{-1}

Table 5.1: Errors between the 'exact' solution (see Figure 5.7(b)) and the numerical solution using HyperC and the coupled limiter after N timesteps of forward advection.

N	Two Material Error	Sum Multiple Materials Error
250	2.72×10^{-2}	3.11×10^{-2}
500	3.00×10^{-2}	3.42×10^{-2}
1,000	3.77×10^{-2}	4.30×10^{-2}
2,000	6.50×10^{-2}	9.34×10^{-2}
3,000	1.16×10^{-1}	1.70×10^{-1}
4,000	1.64×10^{-1}	2.41×10^{-1}

Table 5.2: Errors between the 'exact' solution and the numerical solution after N timesteps of forward advection. Results for the two material case (Figure 3.30) and the sum of the prognostic volume fractions in the multi-material case (Figure 5.7) using HyperC and the coupled limiter are shown.

N	Material 1 Error	Material 2 Error	Material 3 Error	Material 4 Error
250	5.79×10^{-2}	5.79×10^{-2}	5.37×10^{-2}	7.46×10^{-2}
500	7.00×10^{-2}	7.72×10^{-2}	6.07×10^{-2}	8.14×10^{-2}
1,000	8.07×10^{-2}	1.03×10^{-1}	9.56×10^{-2}	1.22×10^{-1}
2,000	1.26×10^{-1}	2.04×10^{-1}	1.30×10^{-1}	2.71×10^{-1}
3,000	1.53×10^{-1}	3.72×10^{-1}	1.64×10^{-1}	5.45×10^{-1}
4,000	1.46×10^{-1}	5.06×10^{-1}	2.33×10^{-1}	6.81×10^{-1}

Table 5.3: Errors for a shearing flow which is reversed after N timesteps and compared to the initial condition. Results for each prognostic volume fraction using HyperC and the coupled limiter are shown. Errors at a lower maximum Courant number of 0.125 are presented in Table B.6.

vi) are shown in Figures 5.9(a)(i–vi)–5.10 after $N = 250$, 500, 1,000, 2,000, 3,000 and 4,000 timesteps using HyperC and the coupled limiter. Again, for each of the reverse runs the materials remain bounded by $[0, 1]$ and are conserved to the same level as the flow is discretely divergence free (10^{-11}). For comparison the reverse shear results from the two material case are reproduced from Figure 3.33(b) in Figure 5.9(c). For small numbers of timesteps the results are very similar, with only slight notches appearing in the outer boundary at the edges of the individual materials. For $N > 2,000$ the material that was shed off the tail of the circle begins to contaminate the final solution and the outer two materials, α^2 and α^4, are substantially more deformed than those areas in the two material case.

The effect of the increased deformation of materials 2 and 4 can be seen in Table 5.3. Here the errors calculated for each material according to Equation 3.65 with $\alpha^e = \alpha_0$ are presented. Materials 2 and 4 have a consistently higher error than materials 1 and 3. As the materials are numbered by priority index, with material 1 being the highest priority material, this shows that the priority is not the dominant control on the error in this case. Instead, the increased shear and higher Courant number experienced by the lower half of the initial circle produces the higher errors.

The increased deformation of the individual materials also leads to higher errors in

Figure 5.9: *See next page.* Results for the reverse advection of Figures 5.7(a)(i–vi)–5.8(a)(i–vi) in a shearing flow, $\tilde{u} = (-\sin(x_1)\cos(x_2), \cos(x_1)\sin(x_2))$, after $N = 250$ (i), 500 (ii), 1,000 (iii), 2,000 (iv), 3,000 (v) and 4,000 (vi) timesteps using HyperC and the coupled limiter. (a) Contours are shown at values of 0.025, 0.5 and 0.975 for all the volume fractions. (b) The absolute difference, $|\alpha^I - \alpha^{I^e}|$, between the numerical solution (a) and the exact solution (see Figure 5.6(a)). Differences greater than the 0.025 contour are shaded in grey for all volume fractions. (c) For comparison the results for the two material case are reproduced from Figure 3.33.

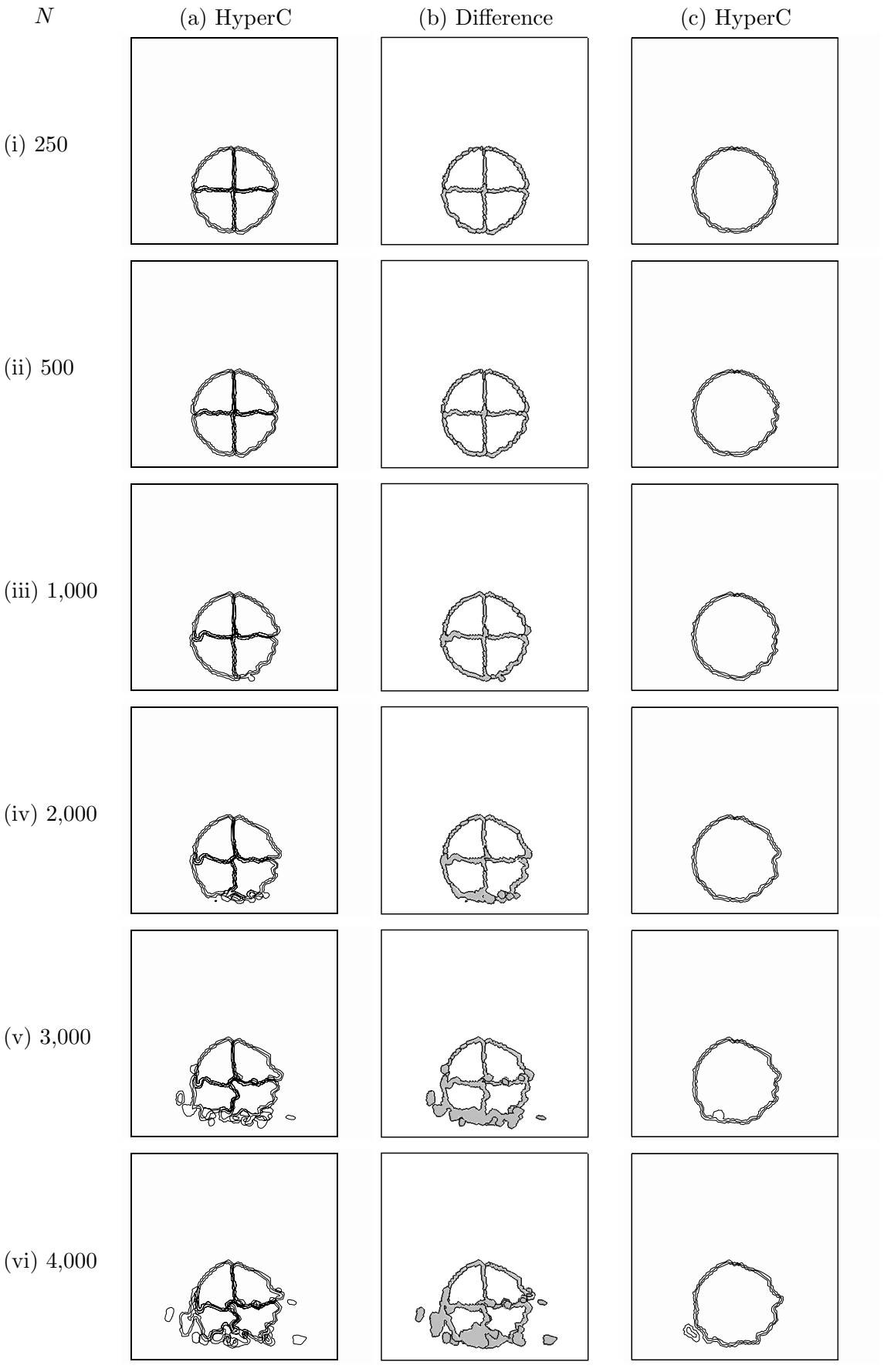


Figure 5.9: *cont.*

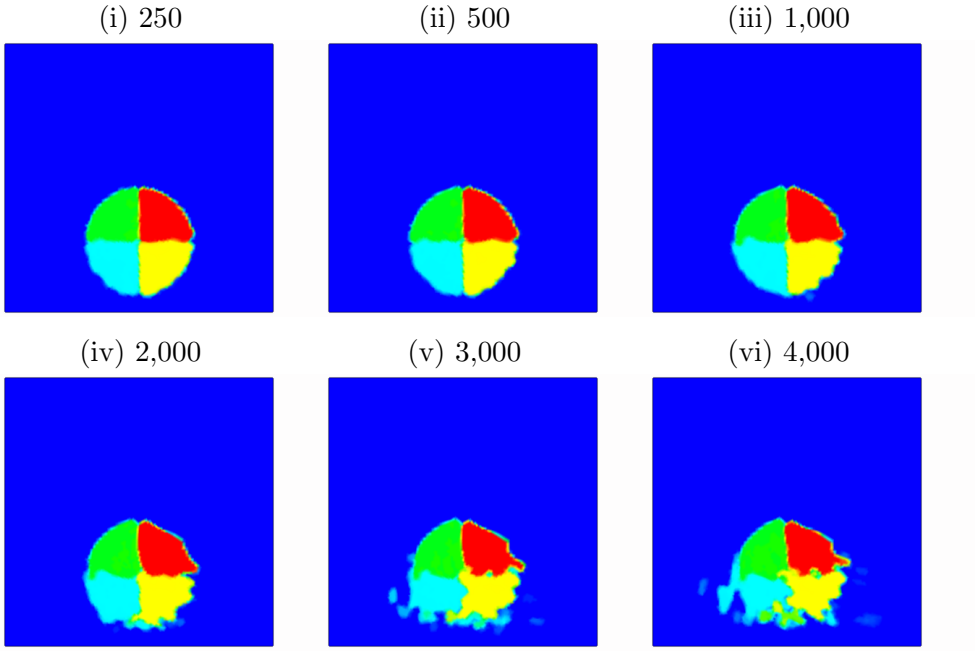


Figure 5.10: Same as in Figure 5.9(a) but showing the bulk density by Equation 3.2, where the density of each material is its priority number (scale shown in Figure 5.6).

	Two Material	Sum Multiple Materials
N	Error	Error
250	2.14×10^{-2}	2.66×10^{-2}
500	2.67×10^{-2}	3.35×10^{-2}
1,000	3.36×10^{-2}	4.30×10^{-2}
2,000	4.03×10^{-2}	7.46×10^{-2}
3,000	6.23×10^{-2}	1.48×10^{-1}
4,000	8.06×10^{-2}	1.90×10^{-1}

Table 5.4: Errors for a shearing flow which is reversed after N timesteps and compared to the initial condition. Results for the two material case (Figures 3.30, 3.33) and the sum of the prognostic volume fractions in the multi-material case (Figures 5.7–5.10) using HyperC and the coupled limiter are shown. Errors at a lower maximum Courant number of 0.125 are presented in Table B.7.

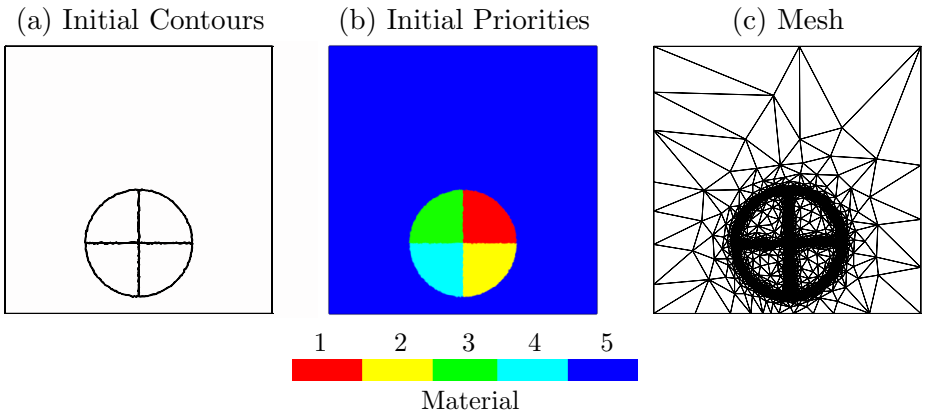


Figure 5.11: The initial condition (a), bulk density showing the material priority indices (b), and initial adapted parent mesh (c) used in the adaptive multiple material shearing flow advection tests.

the overall shape of the combined circle. This can be seen in Table 5.4 where the sum of the materials making up the circle, $\sum_{i=1}^{4} \alpha^i$, is compared with the two material case from Table 3.8. The level of the errors remains approximately the same for small numbers of timesteps. However, once the resolution becomes insufficient to resolve the sheared tail on the forward run of the multi-material case the relative error increases.

Adaptive Mesh

In Section 3.5 an adapted mesh was used to reduced the problem of shedding isolated regions of low volume fraction during the shearing flow test case. This resulted in the tail region remaining contiguous with closely packed contours throughout the simulation. Furthermore, it was possible to achieve similar errors using an adaptive mesh with a quarter of the number of nodes in the fixed mesh. Here the multiple material version of the test is repeated using the same adaptivity settings as those used in Section 3.5, including metric advection and edge length gradation. The initial condition, in contours and in terms of material priorities, is shown alongside the initial adapted parent mesh in Figure 5.11.

The coupled limiter ensures that given a bounded advection algorithm and bounded fields from the previous timestep, the volume fractions will be advected ensuring that their sum is also bounded. The minimally dissipative bounded projection algorithm used so far to transfer the volume fraction fields between successive adapted meshes guarantees the boundedness of each field individually. However it does not maintain the upper bound of the sum of the fields.

The upper bound used by the bounded projection algorithm is specified by $\alpha_{\max}$. At any given node, A, this is the maximum value of the lumped mass projection at A and at the nodes directly connected to A on the parent mesh. The bounding iterations (see Section 2.4) and any subsequent redistribution (see Section 3.5) use this value to decide which nodes are unbounded and if not, how much capacity they have to accept material

from other cells. It is therefore possible to adjust the value of $\alpha^I_{\max}$ to ensure that the maximum bound on the field I will not cause the upper bound on the sum of the fields, $\alpha^{\sum_I}$, to be violated. This is achieved by projecting each field in the same priority order as they are advected. After each field has been projected it can be used to calculate a cap for the maximum bound of the next field, such that:

$$\alpha^I_{\max_A} = \min \left(\begin{array}{c} \alpha^I_{\max_A} \\ \max\left(\alpha^{\sum_I}_{tu}, \alpha^{\sum_I}_{td}\right) - \alpha^{\sum_{I-1}}_{A} \end{array} \right). \tag{5.21}$$

As $\alpha^{\sum_I}_{td}$ and $\alpha^{\sum_I}_{tu}$ are the maximum and minimum bounds on the parent sum this capping ensures its bounds will not be violated by the projection of α^I.

Using a coupled bounded projection algorithm the adaptive test case is run forward for a length of time equivalent to $N = 250$, 500, 1,000, 2,000, 3,000 and 4,000 timesteps on the fixed mesh version of the test. These results are presented in Figure 5.12 showing the contours of each prognostic volume fraction, the bulk density describing the priority of each material, and the adapted parent mesh at each timeframe. As in the fixed mesh case (Figures 5.7–5.8), some notches are again apparent in the outer boundary of the sheared circle where the different materials meet. However, unlike the fixed mesh example, the sheared tail does not disintegrate, remaining contiguous throughout the simulation. Furthermore the coupled limiter and coupled projection maintain the bounds on each volume fraction as $[0, 1]$. If the velocity field is adjusted to be discretely divergence free (to 10^{-9}) after each step of mesh optimisation then the maximum absolute conservation error for all the fields is 10^{-10}.

The results from the reverse advection of Figure 5.12 are presented in Figure 5.13. Again, the bounds of all the fields are preserved and the materials are conserved. Unlike in the fixed mesh run, the contours are closely packed with very few mixed cells, even after the equivalent of 4,000 timesteps. While remaining individually contiguous, materials 2 and 4 are still highly deformed for $N > 2,000$. Additionally, some of the background material, represented by α^5, has been introduced between the other materials. This behaviour was not seen in the fixed mesh run where neighbouring materials maintained their relationship to each other, except in the region of the notches that developed around the outer boundary. In the adaptive case, the background material redistribution occurs during the projection between successive meshes. Since it can always be found by Equation 3.1, α^5 is not projected itself. Instead, it fills in any gaps between the other materials following the field transfer. The bounded projection algorithm acts on each field individually, redistributing it to achieve certain bounds for that field and its parent summation alone. Therefore, it cannot maintain the field's relationships to its neighbours between successive meshes.

If a sufficiently wide gap is established between materials following a projection step the coupled limiter will allow HyperC to treat the gap as two free interfaces which, as with the notches on the outside of the circle, will in some sense establish separate identities. In the case of materials 2 and 4, which experience the highest levels of shearing deformation, successive advection and projection leads to the accumulation of error with increasing

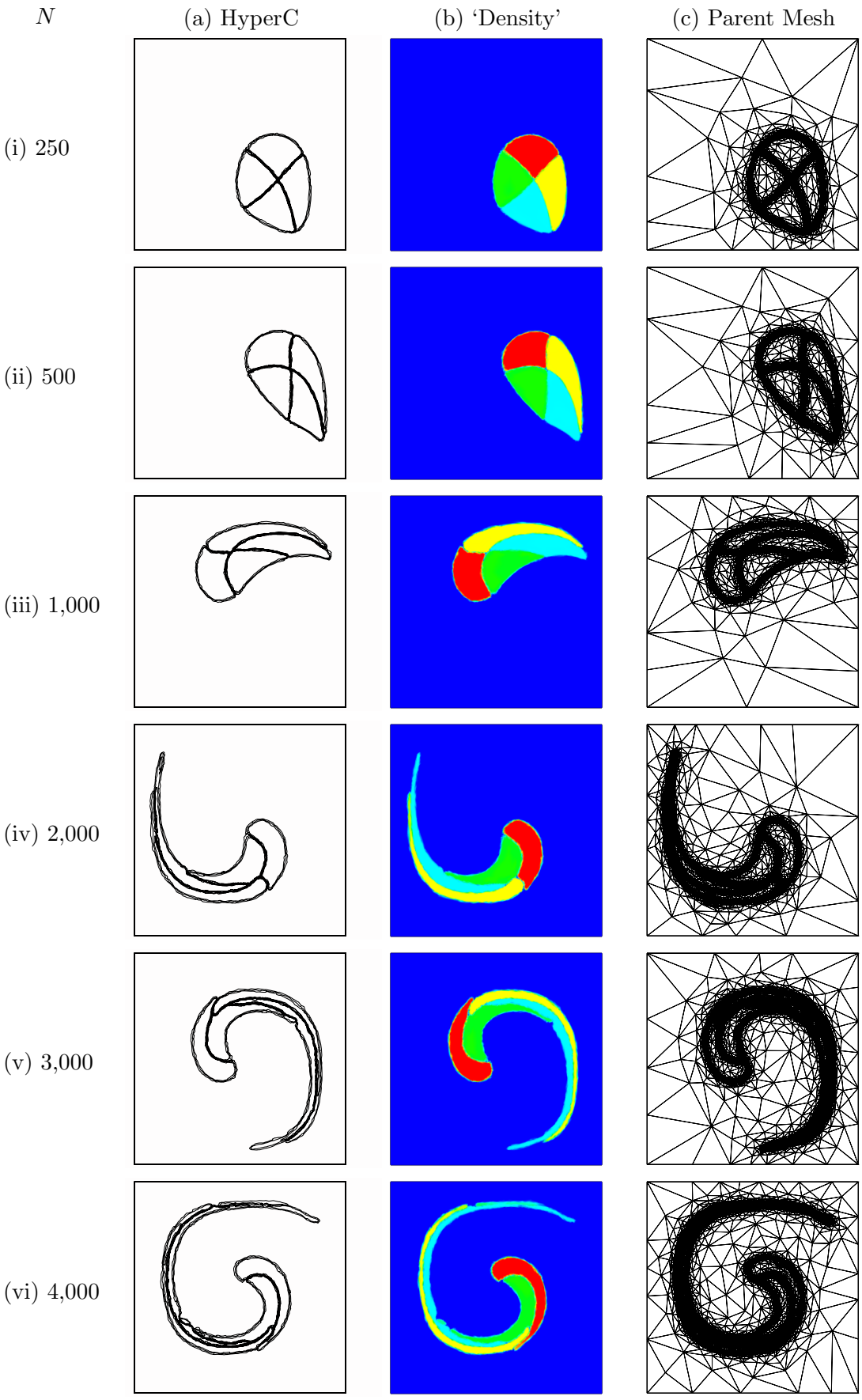


Figure 5.12: *See next page.*

Figure 5.12: *cont.* Results for the forward advection of Figure 5.11(a) in a shearing flow, $\tilde{u} = (\sin(x_1)\cos(x_2), -\cos(x_1)\sin(x_2))$, after $N = 250$ (i), 500 (ii), 1,000 (iii), 2,000 (iv), 3,000 (v) and 4,000 (vi) equivalent timesteps using HyperC and the coupled limiter. (a) Contours are shown at values of 0.025, 0.5 and 0.975 for all the volume fractions. (b) The bulk density by Equation 3.2, where the density of each material is its priority number (scale shown in Figure 5.6). (c) Adapted parent mesh at each timeframe.

numbers of timesteps. Hence in Figure 5.13(vi) the gap appears widest and the interfaces are the most deformed.

Consistent interpolation, using the parent mesh piecewise linear basis functions, performs no redistribution of material. Instead, for any point on the new mesh it locally samples the data within the element at the equivalent point on the old mesh. As, in this case, the basis functions are linear it maintains the maximum and minimum bounds of the field and its parent summation. Furthermore, as it does not redistribute, it cannot introduce information about a field that is not within the old element so gaps between the volume fractions cannot be introduced. However, it is not conservative so it is not suitable for physical multi-material flows.

The lumped mass projection also preserves the relationships between neighbouring fields. Using the lumped mass matrix, $\mathbf{M}^L$ and solving the lumped mass projection:

$$\mathbf{M}^L \alpha^I = r^I, \tag{5.22}$$

produces a bounded, conservative field, α^I (Farrell *et al.*, 2009), where r^I is the right hand side produced by multiplying the mixed mass matrix from the supermesh by α^I on the old mesh. Similarly:

$$\mathbf{M}^L \alpha^{\sum_{K-1}} = r^{\sum_{K-1}}, \tag{5.23}$$

produces a bounded, conservative solution for $\alpha^{\sum_{K-1}}$. However, since the mixed mass and lumped mass matrices are the same for all the fields this can be rewritten:

$$\sum_{I=1}^{K-1} \mathbf{M}^L \alpha^I = \sum_{I=1}^{K-1} r^I. \tag{5.24}$$

Figure 5.13: *See next page.* Results for the reverse advection of Figure 5.12(a)(i–vi) in a shearing flow, $\tilde{u} = (-\sin(x_1)\cos(x_2), \cos(x_1)\sin(x_2))$, after $N = 250$ (i), 500 (ii), 1,000 (iii), 2,000 (iv), 3,000 (v) and 4,000 (vi) equivalent timesteps using HyperC and the coupled limiter. (a) Contours are shown at values of 0.025, 0.5 and 0.975 for all the volume fractions. (b) The bulk density by Equation 3.2, where the density of each material is its priority number (scale shown in Figure 5.11). (c) The absolute difference, $|\alpha^I - \alpha^{I^e}|$, between the numerical solution (a) and the exact solution (see Figure 5.11(a)). Differences greater than the 0.025 contour are shaded in grey for all volume fractions.

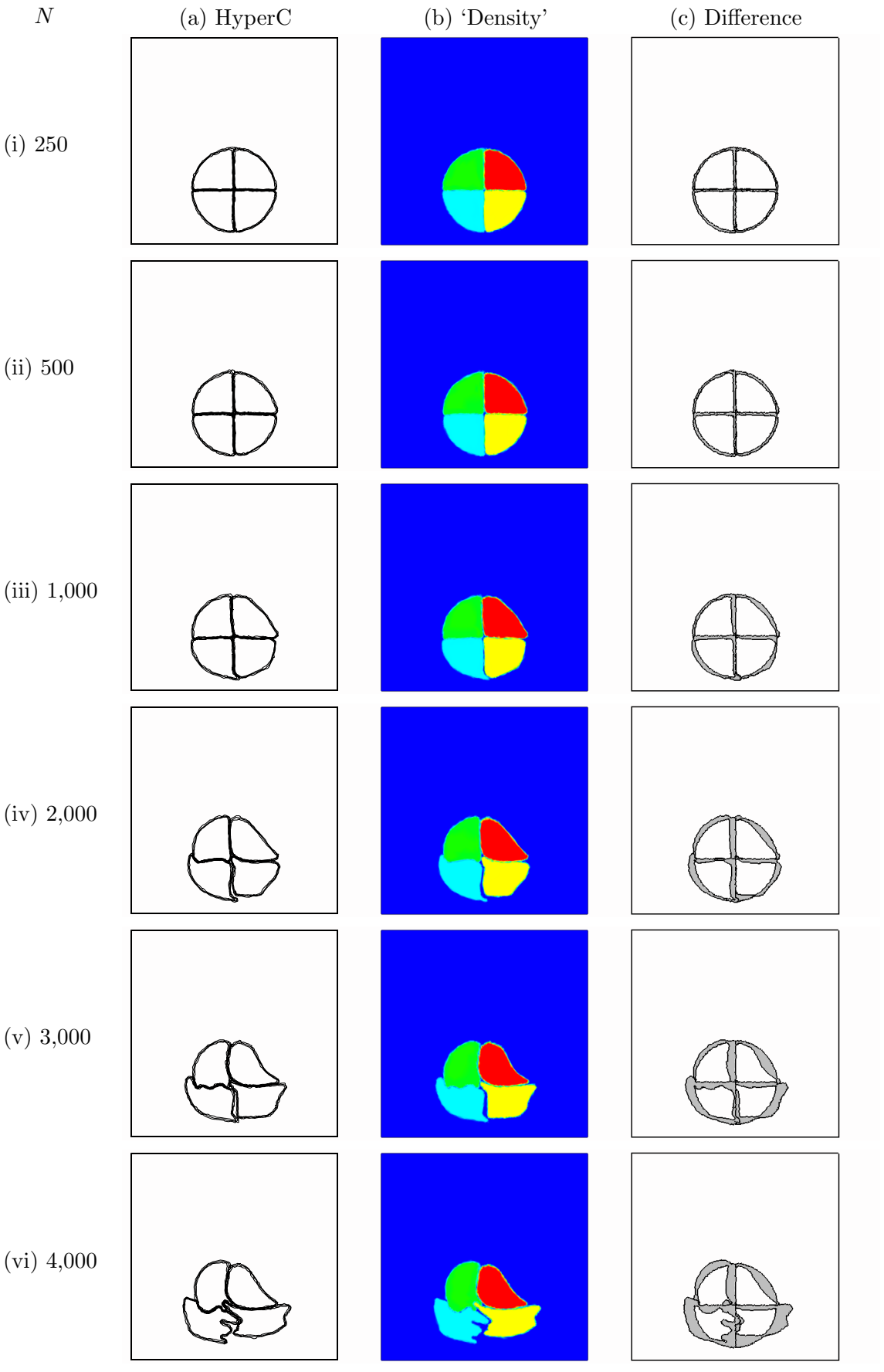


Figure 5.13: *cont.*

N	Material 1 Error	Material 2 Error	Material 3 Error	Material 4 Error
250	4.41×10^{-2}	5.22×10^{-2}	5.93×10^{-2}	8.00×10^{-2}
500	7.76×10^{-2}	9.84×10^{-2}	8.01×10^{-2}	1.15×10^{-1}
1,000	2.04×10^{-1}	1.66×10^{-1}	1.67×10^{-1}	2.08×10^{-1}
2,000	2.87×10^{-1}	2.31×10^{-1}	2.87×10^{-1}	3.08×10^{-1}
3,000	4.14×10^{-1}	4.59×10^{-1}	3.76×10^{-1}	4.12×10^{-1}
4,000	4.45×10^{-1}	6.10×10^{-1}	4.93×10^{-1}	6.24×10^{-1}

Table 5.5: Errors for a shearing flow which is reversed after the equivalent of N timesteps and compared to the initial condition. Results for each prognostic volume fraction using HyperC and the coupled limiter on an adaptive mesh are shown. Errors at a lower maximum Courant number of 0.125 are presented in Table B.8.

Hence, solving $K-1$ equations independently for α^I using the lumped mass matrix also guarantees the boundedness of the sum of the fields, $\alpha^{\sum_{K-1}}$. As boundedness of both the field and parent summation is guaranteed no further redistribution is necessary. Additionally, for any given node, r^I only samples elements in the old mesh that have overlapping support with the new element containing that node. This means that material not contained within the overlapping support cannot be placed at the nodes. The lumped mass projection therefore preserves the relationships between fields. However, it was found to be too diffusive to maintain a sharp interface throughout the simulation and was therefore considered unsuitable for multi-material flows.

The effect of the introduction of gaps between the materials can be seen in the errors presented in Tables 5.5–5.6. As in Section 3.5, these errors are calculated on the supermesh (Figure 5.14(b)(i–vi)). The supermesh contains all the nodes in the initial adapted mesh (Figure 5.11(c)) and the final timestep's adapted mesh (Figure 5.14(a)(i–vi)). It is therefore possible to linearly interpolate the initial (Figure 5.11(a)) and final (Figure 5.13(a)(i–vi)) conditions onto it conservatively before the absolute difference (see Figure 5.13(c)) and errors are calculated (using Equation 3.65, with $\alpha^e = \alpha_0$).

Table 5.5 shows the error for each volume fraction individually while Table 5.6 presents the error of all the materials summed together and compared with the adaptive single material run. In both cases the error has been increased by the failure to preserve the relationships between neighbouring fields. In the case of the individual materials the correlation between location and error is still present but weaker than was seen in the fixed mesh run, presumably due to the deformation of all the materials around their interfaces following projection. Table 5.6 also presents the maximum number of nodes used during the forward and reverse shearing stages. Due to the initial internal structure of the circle this has almost doubled in comparison with the two material case.

Despite the limitations of the bounded coupled projection the results presented above show that HyperC, in combination with the coupled limiter, can be used to advect arbitrary numbers of materials on unstructured adaptive meshes. Important physical parameters of conservation and boundedness are preserved, which is essential for any physical

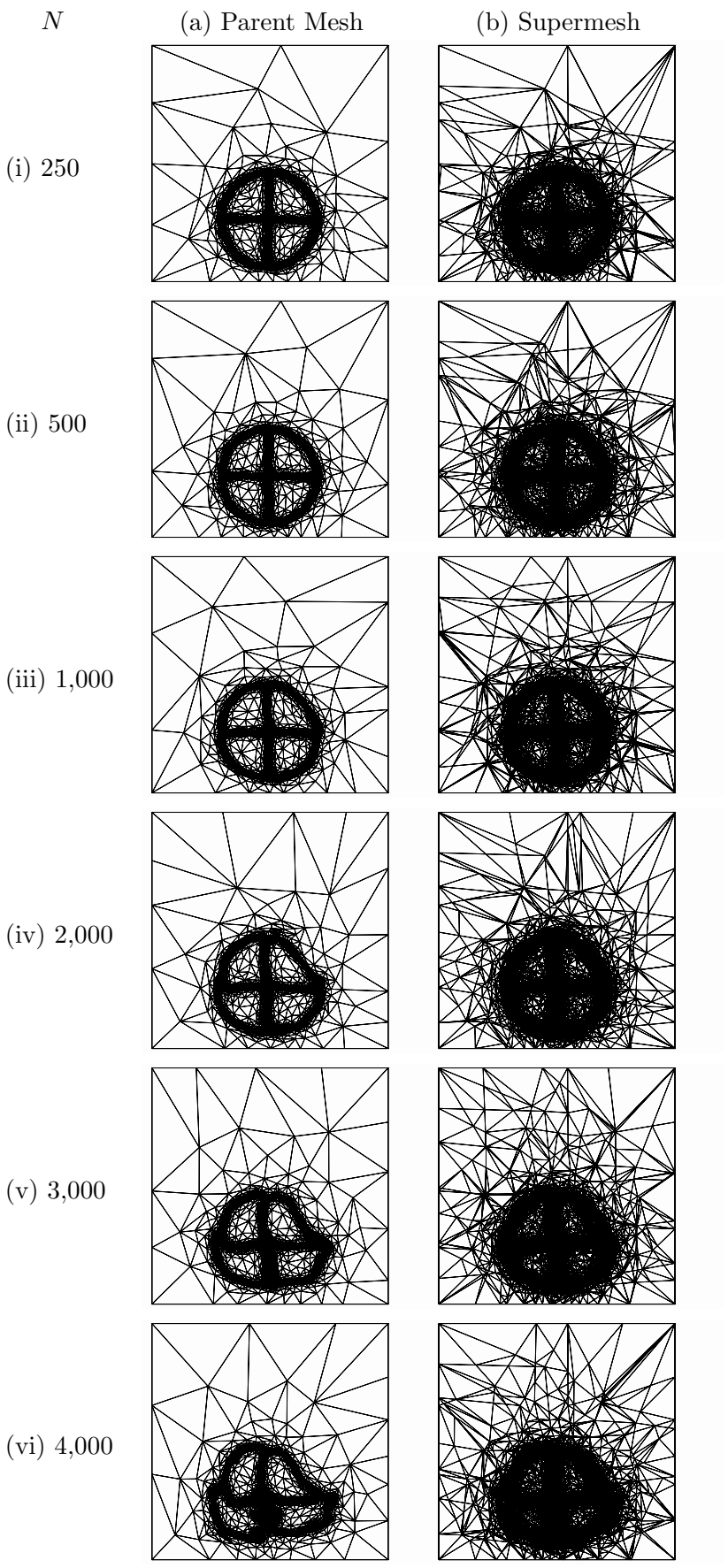


Figure 5.14: *See next page.*

Figure 5.14: *cont.* (a) Adapted parent meshes corresponding to the timeframes shown in Figure 5.13(i–vi), the final timestep in the reverse advection of Figure 5.12(a)(i–vi) in a shearing flow, $\tilde{u} = (-\sin(x_1)\cos(x_2), \cos(x_1)\sin(x_2))$, after $N = 250$ (i), 500 (ii), 1,000 (iii), 2,000 (iv), 3,000 (v) and 4,000 (vi) equivalent timesteps using HyperC and the coupled limiter. (b) The supermesh of the meshes shown in (a) with the initial mesh from Figure 5.11(c), which was used to calculate the errors.

	Two Material		Sum Multiple Materials	
N	Error	Max. Nodes	Error	Max. Nodes
250	2.00×10^{-2}	1,360	3.85×10^{-2}	2,209
500	1.99×10^{-2}	1,360	6.10×10^{-2}	2,209
1,000	3.98×10^{-2}	1,360	1.06×10^{-1}	2,369
2,000	3.99×10^{-2}	1,834	1.76×10^{-1}	3,208
3,000	6.09×10^{-2}	2,371	2.52×10^{-1}	4,015
4,000	1.16×10^{-1}	2,956	3.59×10^{-1}	5,313

Table 5.6: Errors for a shearing flow which is reversed after the equivalent of N timesteps and compared to the initial condition. Results for the two material case (Figures 3.55–3.56) and the sum of the prognostic volume fractions in the multi-material case (Figures 5.12–5.14) using HyperC and the coupled limiter are shown. Errors at a lower maximum Courant number of 0.125 are presented in Table B.9.

model. In the next section a physical example of a three material flow will be investigated: a landslide generated wave.

5.2 Landslide Generated Waves

Landslides impacting into or occurring under water generate waves. If the landslide occurs at a coast then the immediate wave can have a devastating effect in the vicinity of the generation site (Tinti *et al.*, 2006). Depending on the characteristics of the landslide the waves can have significant amplitude (Ward, 2001) and potentially also propagate over much larger distances, becoming tsunami-like. The run-up of these longer distance waves can have an impact far from the location of the original landslide.

Classical earthquake-generated tsunami models are generally based on shallow water theory (Saito & Furumura, 2009). Such models cannot reproduce the highly nonlinear generation mechanisms required to accurately predict the consequences of landslide-generated tsunami. Laboratory-scale experimental investigation is only possible in simple geometries reproducible in a tank. The time scale of experiments is also limited to that period before wave reflections off the tank walls contaminate the data.

A full multi-material model coupled to the Navier-Stokes equations can simulate the

generation of tsunami by landslides at realistic scales. However, depending on the speed of the landslide and depth of the water (Ward, 2001), the first wave may propagate a significant distance before movement in the vicinity of the landslide has stopped and ceased to generate further waves. This suggests that, as in experimental investigations (DiRisio *et al.*, 2009b), a large domain is necessary relative to the size of the landslide. The computational resources required in a structured mesh model scale poorly with increasing domain size. This is especially true when the same level of resolution is required everywhere to capture the wave propagation. Additionally, structured meshes struggle to adequately capture complex bathymetries or coastlines (Ham *et al.*, 2005). Unstructured meshes allow landslide generated waves to be modelled at the same resolution for a lower computational cost. Using adaptive mesh optimisation, resolution may be initially concentrated at the site of the landslide before additional grid points are added to track the propagating wave. Unstructured meshes are also better suited to discretising complex geometries (Gorman *et al.*, 2007).

In this section, the unstructured adaptive multi-material model developed in the last three chapters is tested in two scenarios related to landslide generated waves. First, the run-up of a solitary wave against a steeply sloping coastline is tested and compared to both theoretical and experimental results. Then, using a three material model the full wave generation mechanism as a landslide impacts into water is simulated. This is compared with an experimental model which recreated the generation of the 1958 Lituya Bay mega-tsunami (Fritz *et al.*, 2001).

5.2.1 Wave Run-up on a Sloping Coast

A key element of modelling landslide generated waves is the ability to accurately predict the run-up of waves on a sloping coastline. Given an isolated, or solitary wave of amplitude H, travelling in water of depth D then Synolakis (1987) derived a run-up law to predict the maximum vertical run-up, R, on a coastline sloping at an angle β:

$$\frac{R}{D} = 2.831\sqrt{\cot\beta}\left(\frac{H}{D}\right)^{\frac{5}{4}}. \tag{5.25}$$

Furthermore, numerous experimental results exist (for example Hall & Watts, 1953) validating this theory, which has also been used to test other numerical models (Heitner & Housner, 1970; Kim *et al.*, 1983; Pedersen & Gjevik, 1983).

Here a two material model of water and air was initialised with a single waveform, of maximum amplitude H at the point $x_1 = X_0$, standing on the interface. Using the first approximation of solitary waves (Laitone, 1960; Piggott *et al.*, 2005), the surface height, η, above the standard water depth, D was specified as:

$$\eta = H\mathrm{sech}^2\left(\sqrt{\frac{3H}{4D^3}}\left(x_1 - X_0\right)\right). \tag{5.26}$$

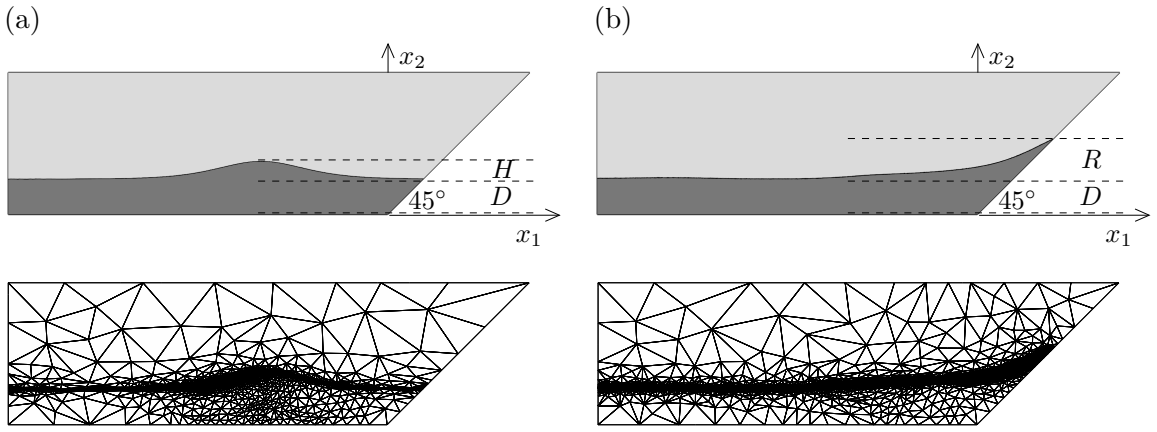


Figure 5.15: (a) Initial condition for a solitary wave with $H/D = 1/2$ impinging on a coastline with $\beta = 45°$. (b) Point of maximum vertical run-up, R, after the wave has hit the coast. The water is represented by the dark grey region and contours of the water volume fraction are shown at 0.025, 0.5 and 0.975. The initial and final adapted meshes are also shown.

Additionally, the initial condition for the velocity of the water was given by:

$$u_{0_1} = \eta \frac{\sqrt{|g|D}}{D}, \tag{5.27a}$$

$$u_{0_2} = \frac{\sqrt{3|g|D}}{D^2} x_2 \sqrt{\frac{H}{D}} \eta \tanh\left(\sqrt{\frac{3H}{4D^3}}\,(x_1 - X_0)\right), \tag{5.27b}$$

where u_{0_1} and u_{0_2} are the initial horizontal and vertical components of velocity in the water respectively. u_{0_i} was set to zero everywhere else, although an initial projection of the velocity field to be discretely divergence free on the control volume mesh ensured that the air was also flowing at the start of the simulation. In his experimental studies, Synolakis (1987) pointed out that, for consistency between the different wave heights, it was necessary to ensure that all waves propagated through the same relative distance. Taking the base of the slope as the origin, X_0 was therefore set to be dependent on the wave height (Synolakis, 1987):

$$X_0 = -\sqrt{\frac{4D^3}{3H}} \cosh^{-1}\left(\sqrt{\frac{1}{0.05}}\right). \tag{5.28}$$

Using the initial conditions described above, a series of two-dimensional numerical experiments were performed for normalised wave heights, H/D, ranging from 5/80 to 1, impinging on a coastline sloping with $\beta = 45°$. An example case using $H/D = 1/2$ is shown in Figure 5.15(a). The water layer was represented by a prognostic volume fraction, α^1, and was assumed to have a density of $\rho^1 = 1,000 kg\,m^{-2}$, while the air had density $\rho^2 = 1 kg\,m^{-2}$. Both materials were treated inviscidly while gravity was taken to have magnitude $|g| = 9.81$. Each simulation was run with $\theta = 1/2$ and $\theta_i = 1/2$ in the momentum equation and a timestep that attempted to ensure a maximum Courant

$\frac{H}{D}$	$\frac{R}{D}$ Numerical	$\frac{R}{D}$ Theoretical
$\frac{5}{80}$	0.127	0.088
$\frac{1}{8}$	0.258	0.210
$\frac{1}{4}$	0.527	0.500
$\frac{1}{2}$	1.122	1.190
1	2.263	2.831

Table 5.7: Comparison of the numerical and theoretical run-up of a solitary wave on a 45° coast.

number of 2.5. The volume fraction advection was subcycled approximately 10 times per timestep to maintain a maximum Courant number of 0.25. All boundaries except the top had no normal flow, $d_i n_i = 0$, boundary conditions applied. The top of the domain was left open, $p = 0$. The position of the left boundary of the domain varied between simulations to ensure that larger waves were adequately represented and to prevent any reflections contaminating the solution. Hence the maximum extent of the domain (ignoring the corner cut out for the slope) was defined as Ω: $x_1 \in [3X_0, 4D]$, $x_2 \in [0, 4D]$.

The mesh was adapted every 10 timesteps and metric advection and edge length gradation were enabled as before to ensure sufficient resolution was available between each step of mesh optimisation. The effect of metric advection at the first timestep can be seen in Figure 5.15(a) where the mesh is asymmetric about the wave, preferentially increasing the resolution in the direction the wave is travelling. The minimum edge length was constrained to be $D/200$ while the maximum edge length in each dimension was prevented from being larger than approximately a quarter of the size of the domain in that direction. The pressure and volume fraction were directly adapted to using interpolation error bounds, $\hat{\epsilon}$, of $100Pa$ and 0.09 respectively. The vertex based projection of the velocity was also adapted to an interpolation error bound of $0.04m\,s^{-1}$ for each component. Given the expected ranges of these fields these correspond to desired errors in the range of 1–10%. As before the volume fraction is transferred between successive meshes using the bounded projection algorithm. The velocity is transferred using a straightforward projection while the pressure is consistently interpolated using the linear basis functions from its parent mesh.

An example of the point of maximum run-up is shown in Figure 5.15(b) for $H/D = 1/2$ along with the adapted mesh using the settings described above. The ratio of the maximum run-up to water depth, R/D, for all the values of H/D used here is shown in Table 5.7 alongside the theoretical result from Equation 5.25. These results are also shown graphically in Figure 5.16 with additional data from the experimental results of Hall & Watts (1953). The numerical results calculated here can be seen to be in close agreement with both the theoretical and experimental results. At lower values of H/D the numerical results trend away from the theoretical line, as do the experimental results.

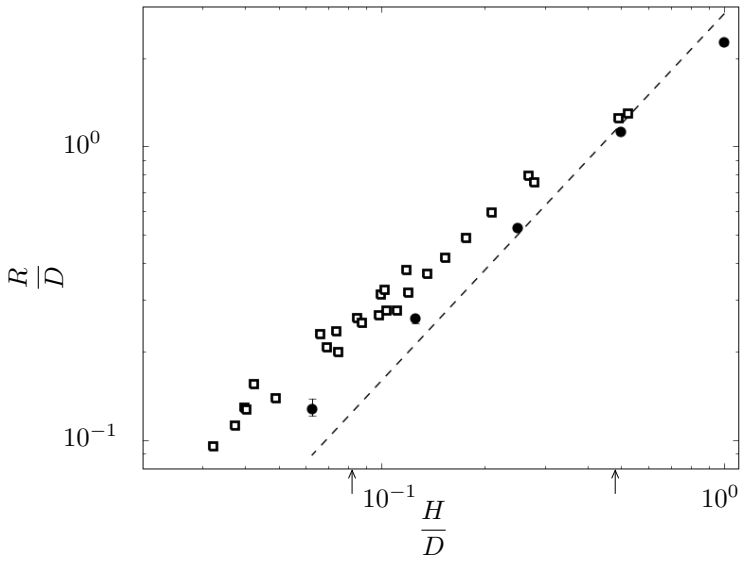


Figure 5.16: Normalised run-up versus normalised height of a solitary wave impinging on a coast sloping at 45°. Experimental (hollow squares, from Synolakis (1987)), theoretical run-up law (dashed line, from Hall & Watts (1953)) and numerical (solid circles) values are shown. The numerical results have associated error bars showing the limited variation in the result if the height of the water is taken at volume fraction values of 0.025 and 0.975 (rather than $\alpha^1 = 0.5$ at the central point). Arrows on the horizontal axis indicate the lower and upper validity bounds of the run-up law, by Equations 5.29 and 5.30 respectively.

This behaviour is expected as the run-up law is only valid when:

$$\frac{H}{D} \gg 0.082944 \tan^2 \beta. \tag{5.29}$$

Hence the lower values of the run-up law, presented in Figure 5.16, are not expected to be formally correct when $\tan \beta = 1$ (Synolakis, 1987). Similarly at the other end of the scale the behaviour of the wave run-up is anticipated to change from a non-breaking to breaking wave at approximately (Gjevik & Pedersen, 1981):

$$\frac{H}{D} = 0.479 \tan^{\frac{10}{9}} \beta. \tag{5.30}$$

The final numerical data point is therefore considered to be in transition between the two behaviours and the theoretical prediction is not valid for breaking waves. Furthermore, the first approximation to solitary waves is known to break down as the ratio of H/D tends toward unity (Daily & Stephan, 1952; Ippen & Kulin, 1955; Laitone, 1960). Therefore, the discrepancy between the theoretical and numerical results at both ends of the spectrum can be discounted and overall, lying between the theoretical and experimental values, the numerical results are considered to be an excellent fit.



Figure 5.17: Satellite image from August 2001 of Lituya Bay, Alaska. The yellow line shows the maximum height of the scar (Miller, 1960b) following the 1958 landslide at the site indicated in red. Image taken from Weiss *et al.* (2009).

5.2.2 Lituya Bay Landslide

Lituya Bay is a T-shaped tidal inlet on Alaska's southern coast (see Figure 5.17). The inlet was formed by a glacier and has a broad flat floor with steep walls. The two branches of the bay, the Gilbert and Crillon inlets, still contain the remnants of the glacier. These lie in a trench that represents the topographic expression of the Fairweather fault. The Gilbert inlet is just over $1km$ wide with a flat base and steep walls sloping at an angle of approximately $40°$ from the horizontal.

On 10 July, 1958 an earthquake on the Fairweather fault with moment magnitude 8.3 triggered a subaerial landslide on the northeast slope of Gilbert inlet (Tocher & Miller, 1959). The landslide ran downslope into the inlet, shearing the nose off the Lituya glacier and impacting with the water at considerable speed. This impact generated a large wave which travelled across the inlet. The run-up height on the opposite shoreline, to the southwest, was recorded by the trimline of the trees that were washed away by the water. Directly opposite the landslide the run-up was $524m$ (Miller, 1960a), the highest recorded in history (Fritz *et al.*, 2009). The destructive power of the landslide generated wave in other areas was also significant, with run-ups of up to $208m$ (Miller, 1960b) recorded in the main stem of the bay (see Figures 5.17–5.18).

The landslide was composed of material from the northeast slope of the Gilbert inlet up to a height of $915m$, where the rocks are mainly amphibole and biotite schists with an

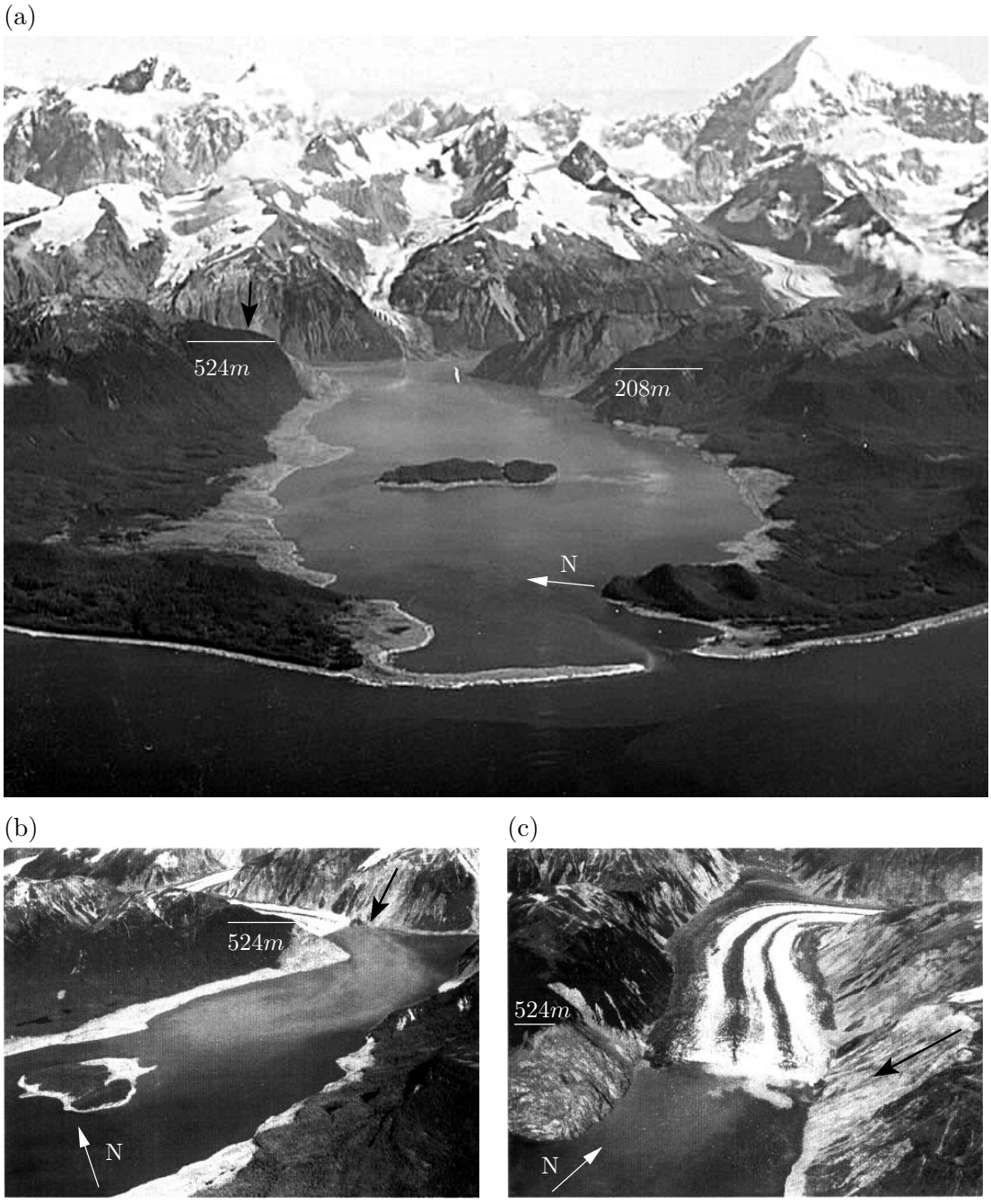


Figure 5.18: Aerial views of Lituya Bay taken by Tocher & Miller (1959). The landslide location is shown as a black arrow in each image. (a) The entire bay from the west, showing the two highest points of run-up. (b) The majority of the bay from the south west, showing Gilbert inlet and the Lituya Glacier in the background. (c) View up Gilbert inlet from the south east, showing Lituya glacier with its nose removed by the landslide. The point of maximum run-up is directly across the inlet from the landslide.

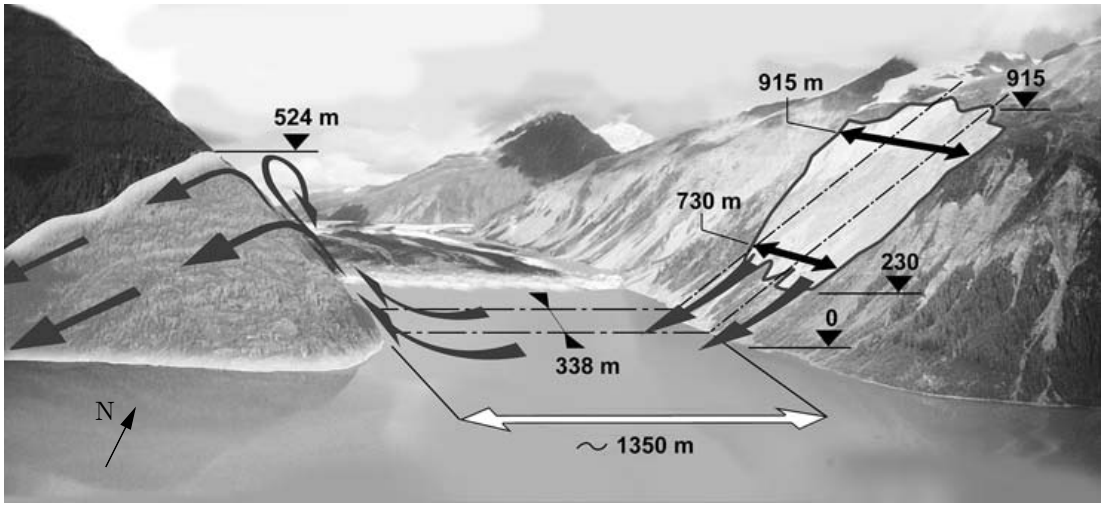


Figure 5.19: View up the Gilbert inlet showing the dimensions of the landslide and the near two-dimensional geometry of the subsequent run-up. Image taken from Fritz *et al.* (2009).

assumed density of $2,700kg\,m^{-3}$ (Miller, 1960b). The landslide varied in width between 732 and $915m$ and was estimated to have been $907m$ long measured down slope (Miller, 1960b; Slingerland & Voight, 1979). The cross sections of the slide are assumed to have been approximately triangular with a maximum thickness of about $92m$ normal to the slope with a centre of gravity at about $610m$ altitude (Miller, 1960b). Miller (1960b) estimated the total slide volume as $30.6 \times 10^6 m^3$, although the slide also accumulated material from a gravel delta at the nose of the Lituya glacier in the Gilbert inlet, which was sheared off during the impact.

Upon impacting the water, the landslide generated a wave which travelled away from the impact site, running up the opposite shoreline of Gilbert inlet and into the rest of Lituya Bay. As the maximum wave run-up occurred directly opposite the landslide in the direction of its propagation it is possible to reproduce the slide, impact and maximum run-up stages in a two-dimensional model (see Figure 5.19). Fritz *et al.* (2001) built a 1:675 scale model of a cross section of Gilbert inlet, approximating the glacially carved topography by a flat bottomed tank with sides sloping at 45°. The water depth was $122m$ and the distance between the model coastlines was $1,342m$. Note that all units of distance are given in the equivalent real-world scale of Lituya Bay.

The volume per unit width of the landslide was taken as $27.2 \times 10^3 m^2$ by averaging the total volume over an averaged slide width of $823m$. The additional material in the glacier nose and delta was ignored while the schist was approximated by a granular material with density of $2,640kg\,m^{-3}$, giving a mass per unit width of $98,200kg\,m^{-1}$. Due to a porosity of 39% this resulted in a bulk density of $1,610kg\,m^{-3}$ and a volume per unit width of $61.2 \times 10^3 m^2$.

Assuming the centre of mass of the landslide was at $610m$, Fritz *et al.* (2009) estimated that the impact velocity was $110m\,s^{-1}$. To achieve this a pneumatic acceleration

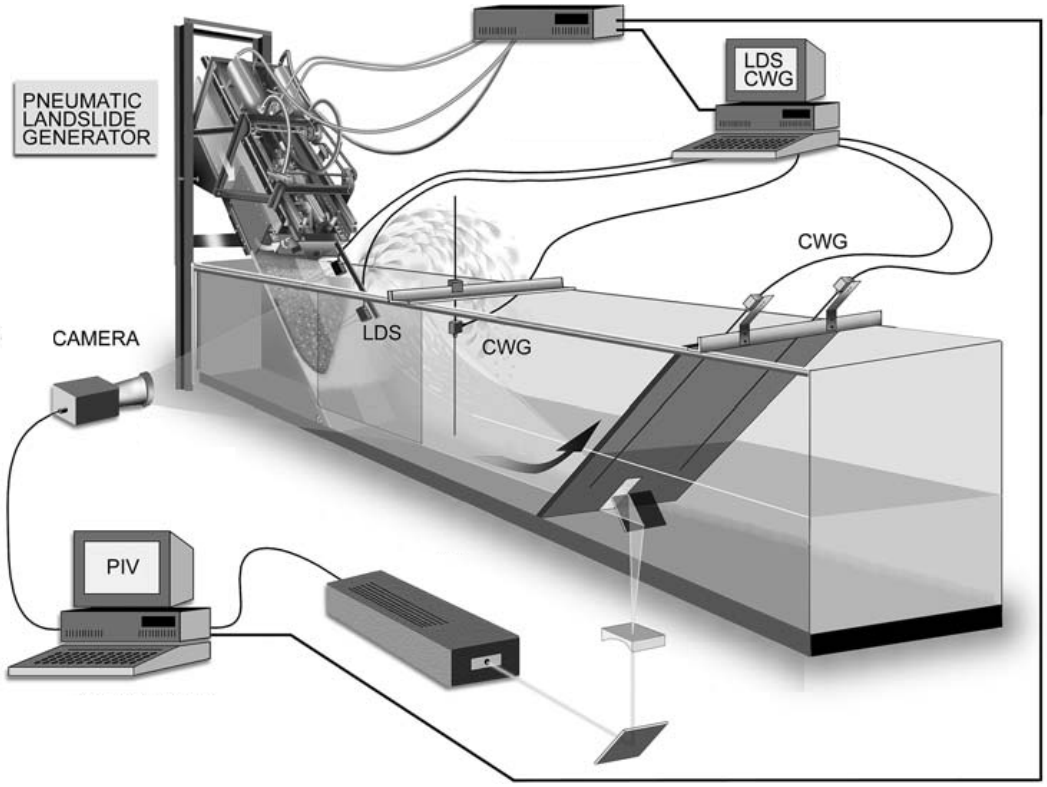


Figure 5.20: Experimental set-up taken from Fritz *et al.* (2009). The pneumatic landslide generator (Fritz & Moser, 2003), laser distance sensor (LDS), capacitance wave gauges (CWG) and particle image velocimetry (PIV) sensor are shown.

mechanism (Fritz & Moser, 2003) was used to first accelerate the landslide as a block before releasing it and allowing it to impact into the water as a granular flow (see Figure 5.20). Several measurements were taken throughout the experiment. A laser distance sensor was used to measure the thickness of the landslide just prior to it entering the water, at $(-67, 67)m$ on the landslide 'NE' slope. Two capacitance wave gauges were used: one measuring the wave height at $885m$ from the NE coastline and another measuring the maximum run-up on the opposite 'SW' shore. Particle image velocimetry images were also produced during the experiment showing close-up images of the velocity field in the vicinity of the impact and at the other end of the tank during the period of maximum run-up.

Several timeframes from the experiment can be seen in Figure 5.21. Before impact, the landslide has a smooth shape with increasing thickness away from the water (Figure 5.21(a)). However, the impact of the landslide with the water and with the bottom of the tank causes thickening of the granular material (Figure 5.21(b)). A large air cavity is opened in the water behind the slide (Figure 5.21(b, c)), the collapse of which causes a large degree of mixing between the water and air (Figure 5.21(d–f)). This mixing has little impact on the main wave propagation as it has already begun propagating away from the impact site by the time of collapse. This wave appears to begin to break before

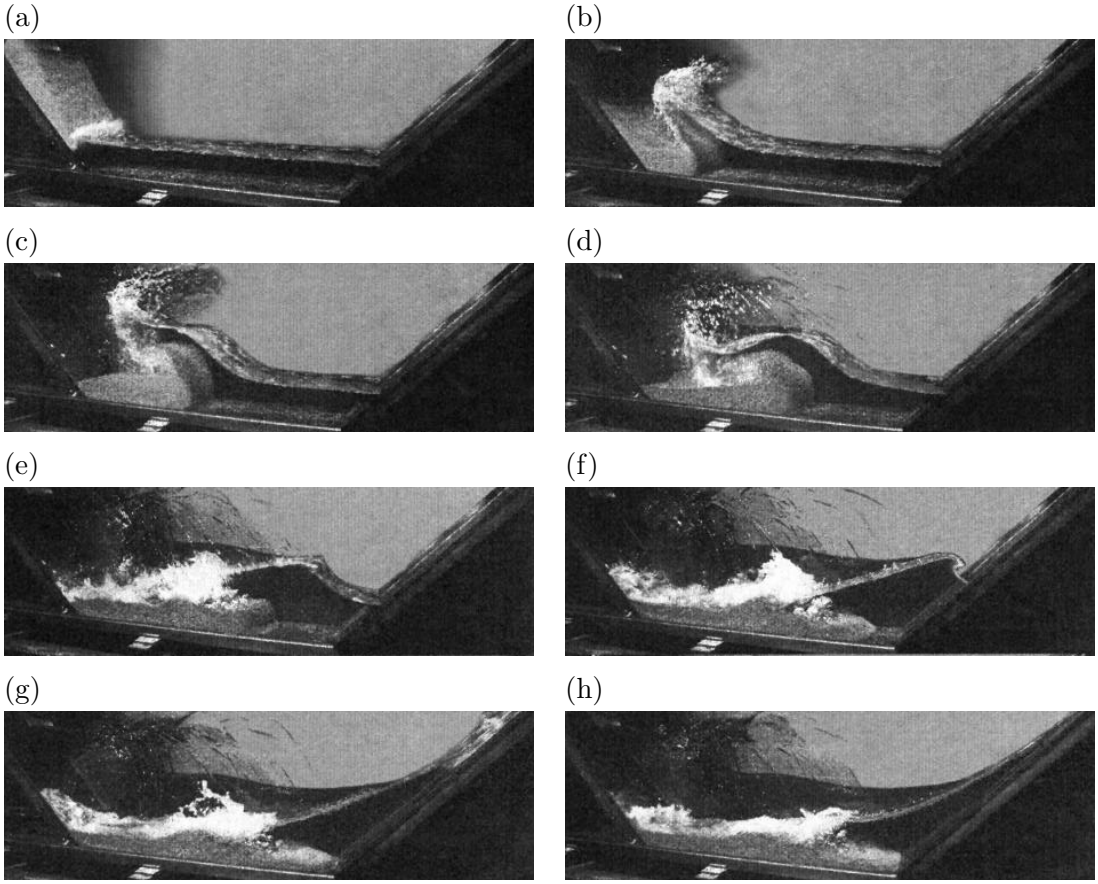


Figure 5.21: Time sequence from the experimental impact of a granular material with water in a 1:675 scale model of a cross section of Gilbert inlet. Photos are spaced $5s$ apart and are taken from Fritz *et al.* (2001).

striking the opposite slope and surging up it, with little air entrainment (Figure 5.21(f–h)). During the experiment the first wave gauge recorded a maximum wave height of $152m$ $16s$ after impact while the wave run-up gauge measured a maximum vertical run-up of $526m$ $35s$ after impact.

Several attempts have been made to numerically model the 1958 Lituya Bay landslide. Mader & Gitting (2001) first attempted to model the entire bay using a depth integrated, shallow water tsunami propagation model (SWAN, Booij *et al.*, 1996). The landslide was considered to have instantaneously displaced an equivalent volume of water up to a maximum height equal to the water depth. This resulted in the run-up on the opposite shoreline being underestimated by an order of magnitude and Mader & Gitting (2001) concluded that a full Navier-Stokes model was required to capture the wave generation process.

Subsequent numerical models have only attempted to reproduce the experimental behaviour in the two-dimensional cross section of Gilbert inlet. Mader & Gittings (2002) used the Eulerian compressible hydrodynamics code SAGE (Gisler *et al.*, 2006). Using a two-dimensional domain similar to that used by Fritz *et al.* (2001) and a landslide

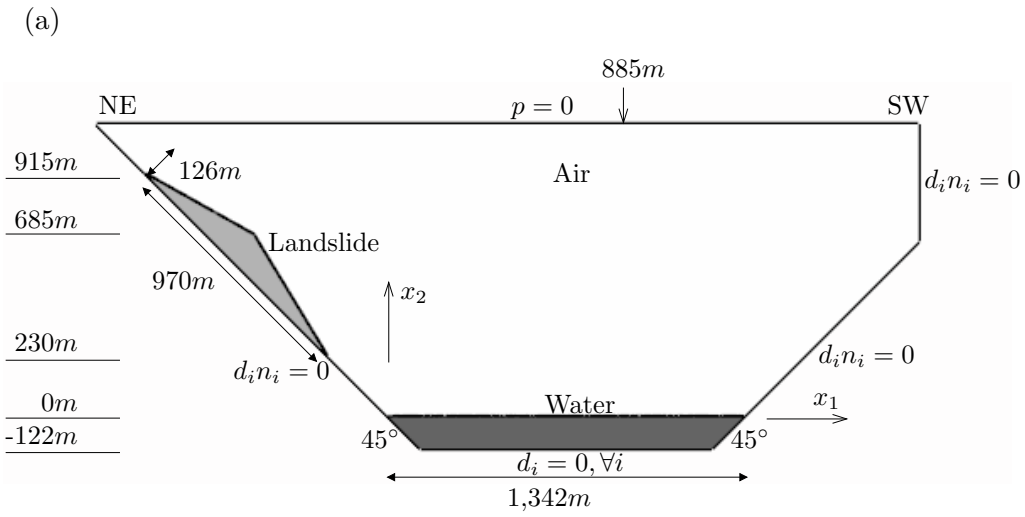
impacting at $100m\,s^{-1}$, a maximum wave height of $250m$ and run-up of $580m$ were produced. Several timeframes of the simulation were also presented which suggested that the material tracking algorithm used was particularly diffusive with the interfaces gradually losing coherence.

Quecedo *et al.* (2004) used a level set material tracking algorithm to better capture the interfaces between the landslide, water and air. This was done in the same domain as used by Fritz *et al.* (2001), tessellated using a triangular finite element mesh. The velocity-pressure system was discretised using a stabilised $\mathrm{P1_LP1}$ element pair. Despite being stabilised the pressure solution was reported to blow up in the vicinity of the interfaces and an extra smoothing function was applied to the bulk density to maintain stability. A variety of different rheologies of landslide were modelled which produced maximum wave heights at the $885m$ wave gauge between 206 and $266m$. The maximum wave run-up on the opposite shore varied between 550 and $730m$. Timeframes presented from the simulations suggested that the free slip boundary conditions used allowed the landslide to propagate too far in comparison to the experiments, often running up the opposite shore along with the wave.

Schwaiger & Higman (2007) used a particle based, Lagrangian SPH model to recreate the experiment. The landslide was approximated as a viscous fluid with a wedge-like cross section, which started from rest on the NE shore with the thin edge pointing downslope. As the viscosity was increased the maximum wave height and run-up were found to decrease. The closest wave gauge and run-up values were $156m$ and $530m$ respectively, using a viscosity of $10^{9/2}Pas$. However, the model failed to correctly reproduce the landslide shape during the impact or the wave gauge data after the first maximum.

Most recently, Weiss *et al.* (2009) modelled the experiment using the finite difference hydrocode iSALE (Amsden, 1980; Collins *et al.*, 2004; Wünnemann *et al.*, 2006), which uses VOF interface reconstruction techniques to track materials. The landslide was modelled as a wedge of material with a plastic rheology. The blunt end of the wedge was initially pointing downslope and an initial velocity was imposed to ensure that the impact occurred at $110m\,s^{-1}$. This produced a maximum run-up of $518m$ on the opposite shore and a close agreement between the $885m$ wave gauge data and the experimental results throughout the simulation.

Here, the landslide is approximated as a viscous fluid (as in Schwaiger & Higman (2007) and Fine *et al.* (2005)) with density $1,610kg\,m^{-2}$ and area $61.2 \times 10^3 m^2$. The viscosity was varied to achieve a best fit with the experimental data at the wave gauges. The slide material was initially arranged as a triangular prism with its shaped determined by the maximum and minimum limits of the physical landslide and the requirement that its centroid be at $610m$ altitude. To ensure that the landslide hits the water at $110m\,s^{-1}$ it was given an initial velocity of $(61.6, -61.6)m\,s^{-1}$ and the velocity boundary conditions on the NE slope were set to no normal flow, $d_i n_i = 0$. While the rest of the material in the domain was initially at rest, the projection of this velocity field to be discretely divergence free before the first timestep ensured that some flow also occurred in the air. The boundary condition on the base of the domain was set as no slip, $d_i = 0\,\forall i$. The initial set-up is shown in Figure 5.22(a).



(b)

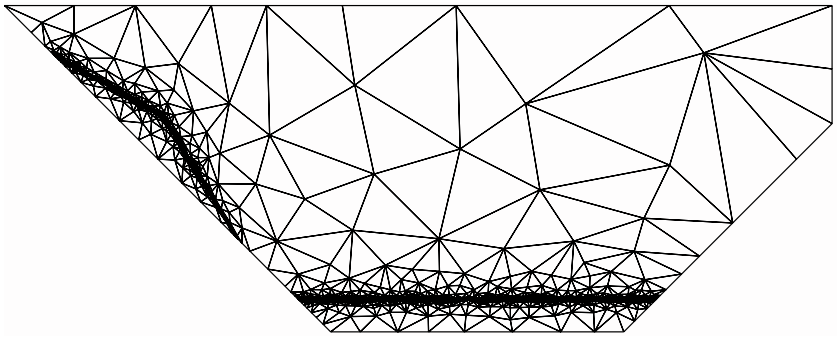


Figure 5.22: Initial set-up (a) and the initial adapted mesh (b) used to model the 1958 Lituya Bay landslide experiment. The location of the landslide and water are shown using grey shading and contours at 0.025, 0.5 and 0.975 of their respective volume fractions. The landslide is initially represented as a triangle with centre of mass at $610m$ height on the NE slope. The water depth is $122m$. The location of the wave gauge at $885m$ and the velocity and pressure boundary conditions used are also shown.

The water had an initial maximum depth of $122m$ and, like the air, was modelled inviscidly. The water and air had densities of $1,000kg\,m^{-2}$ and $1kg\,m^{-2}$ respectively. As the water was of most interest it was assigned the highest priority volume fraction, α^1, and was advected using HyperC. The landslide was the second highest priority volume fraction, α^2. As it was treated as a viscous fluid, the more compressive advection algorithm UltraC-S was used to ensure it did not contaminate the neighbouring materials. The air made up the background third material, α^3, found using Equation 3.1. Using the coupled limiter, bounded behaviour was preserved in all volume fractions, which were also conserved throughout each simulation. The temporal discretisation of momentum used $\theta = 1/2$ and $\theta_i = 1/2$ and a timestep that attempted to ensure the maximum global Courant number did not exceed 2.5. The advection of the volume fractions was subcycled approximately 10 times per timestep to prevent the maximum advective Courant number from exceeding 0.25.

The mesh was adapted every 10 timesteps using metric advection and edge length gradation as in Section 3.5. The minimum edge length was constrained to be $0.5m$ while the maximum could not exceed $500m$. The interpolation error bound on both volume fractions was 0.05 while the pressure and node-wise projection of the velocity fields used $10^6 Pa$ and $5.0m\,s^{-1}$ respectively. Based on the range of values seen, these correspond to errors of between 5 and 10%. The initial adapted mesh is shown in Figure 5.22(b) using these settings. Field data were transferred between meshes as in Section 5.2.1.

Several timeframes from the numerical simulation using a landslide viscosity of $4.5 \times 10^6 Pas$ are shown in Figure 5.23 alongside the adapted meshes from the same times. Similar behaviour to that seen in the experiment (Figure 5.21) can be seen in Figure 5.23. On entry into the water the landslide is wedge like, sliding in between the boundary and the water (Figure 5.23(a)). However interaction with the water and the base of the domain leads to thickening and the opening of an air cavity behind the slide material (Figure 5.23(b)). This air cavity collapses as a solitary wave propagates away from the impact site (Figure 5.23(c)). The wave inundates the opposite shore, surging up the slope (Figure 5.23(d, e)). It then begins to flow back down the slope turning into a steep wall of water at its base (Figure 5.23(f, g)). This becomes an overturning wave, trapping an air bubble and rushing back up the slope (Figure 5.23(h)).

A few differences can also be seen between Figures 5.23 and 5.21. The high levels of mixing as the air cavity collapses are not visible in the numerical results. All multi-material models attempt to resolve the interface between different materials rather than allowing mixing, which would be non-physical with a single velocity field. Therefore the failure to reproduce the behaviour behind the landslide is anticipated given the model assumptions. This is not expected to affect the initial wave results as both in the experiment and in the numerical simulation this has already begun propagating away from the impact site by the time the cavity collapses. The propagating wave in the numerical results however appears to be less nonlinear than the wave seen in the experiments. It therefore does not begin breaking as it reaches the shoreline, instead travelling up the slope without any overturning. This may affect the maximum run-up results even though the wave in the experiment did not break fully, instead surging up the slope (Fritz *et al.*,

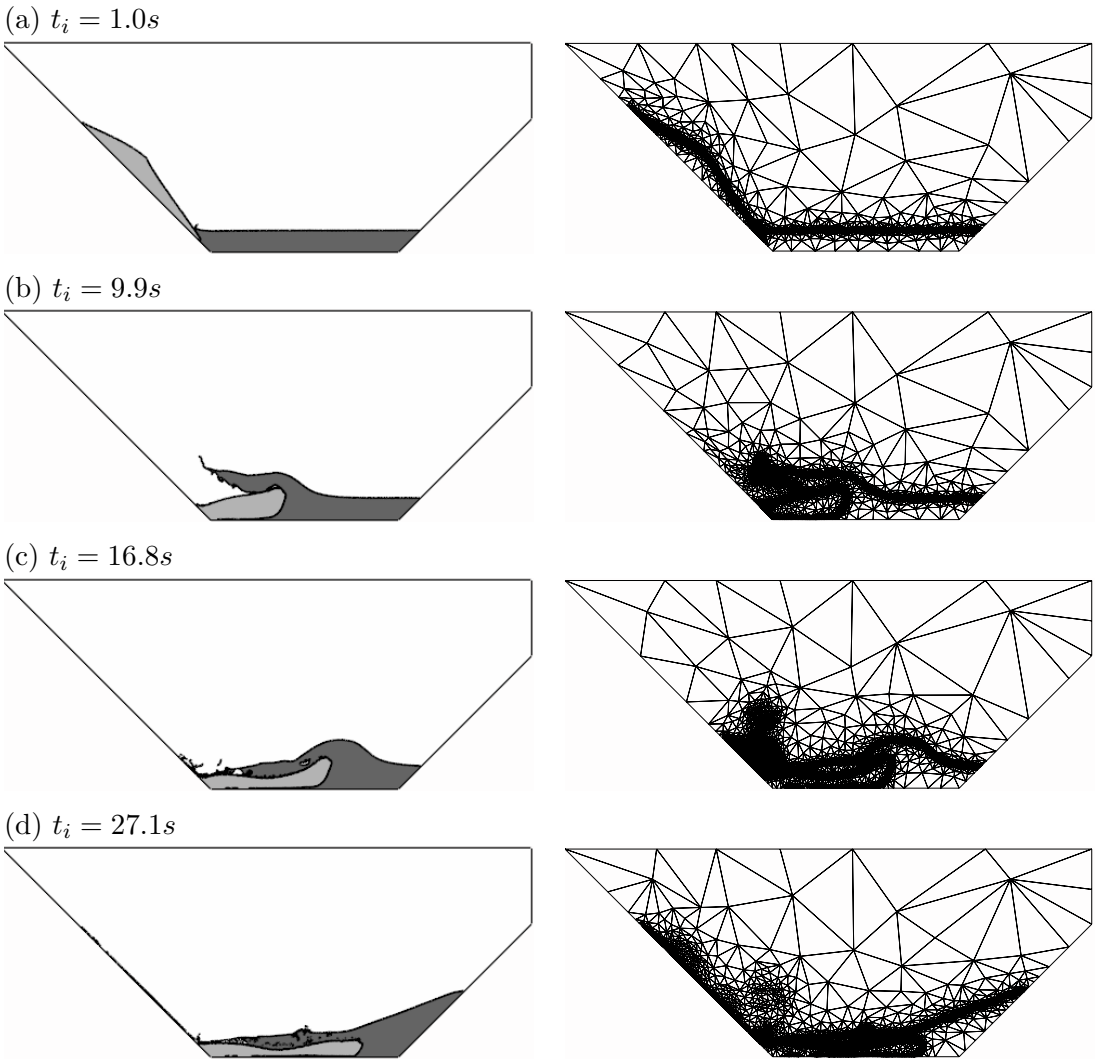


Figure 5.23: Several timeframes from the simulation of the 1958 Lituya Bay landslide using a landslide viscosity of $4.5 \times 10^6 Pas$. The times shown are in seconds after the initial impact of the landslide with the water, t_i. As in Figure 5.22(a) the landslide is shaded light grey while the water is dark grey. Contours are also shown at 0.025, 0.5 and 0.975 of their respective volume fractions.

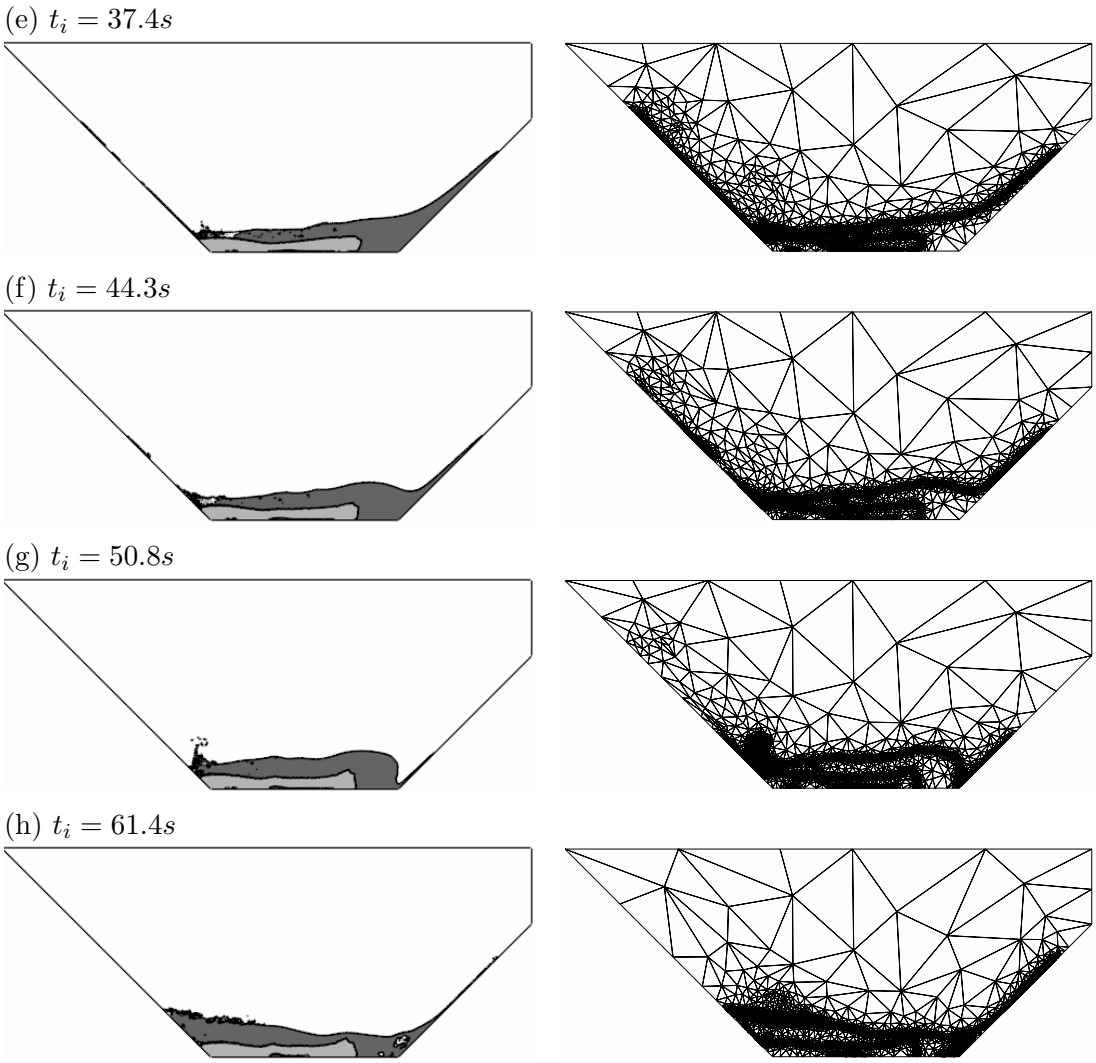


Figure 5.23: *cont.* Several timeframes from the simulation of the 1958 Lituya Bay landslide using a landslide viscosity of $4.5 \times 10^6 Pas$. The times shown are in seconds after the initial impact of the landslide with the water, t_i. As in Figure 5.22(a) the landslide is shaded light grey while the water is dark grey. Contours are also shown at 0.025, 0.5 and 0.975 of their respective volume fractions.

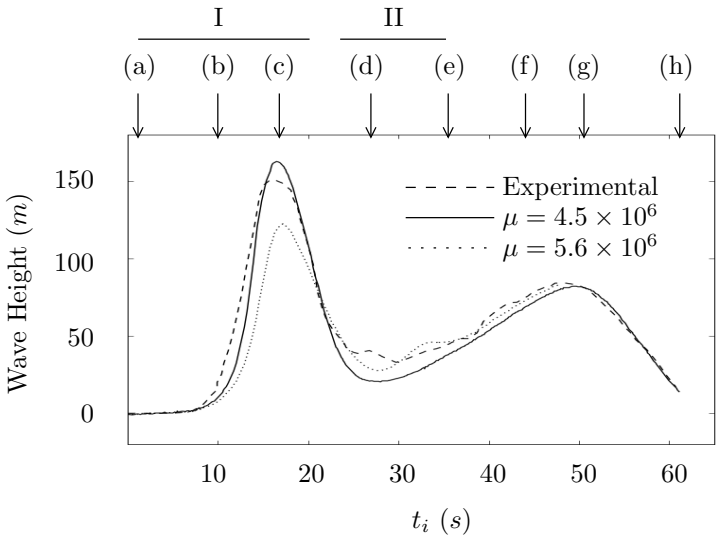


Figure 5.24: Data from the wave height gauge at $885m$ plotted against time after the landslide had impacted into the water, t_i. Numerical results for landslide viscosities of $4.5 \times 10^6 Pas$ (solid line) and $5.6 \times 10^6 Pas$ (dotted line) are shown alongside the experimental findings of Fritz *et al.* (2001) (dashed line). Letters (a–h) above the graph indicate the times corresponding to each frame in Figure 5.23. Roman numerals I and II indicate the time periods covered by the close-up images in Figures 5.27–5.28 and Figures 5.29–5.30 respectively.

2009).

Data from the wave height gauge at $885m$ from two numerical simulations are presented in Figure 5.24 alongside the experimental results taken from Fritz *et al.* (2001). A high landslide viscosity of $5.6 \times 10^6 Pas$ is seen to underestimate the initial wave height, only reaching $122m$. A lower value of $4.5 \times 10^6 Pas$ (as presented in Figure 5.23) gives a much better fit to the experimental data, with a maximum wave height of $163m$. The subsequent drop in wave height after the initial wave and generation of a second wave during the rundown off the SW shore is reproduced well by both simulations.

The data from the run-up gauge on the SW shore are shown in Figure 5.25, which compares the numerical results using two landslide viscosities with the experimental data from Fritz *et al.* (2001). Both viscosities underestimate the maximum run-up. The high viscosity case, $\mu = 5.6 \times 10^6 Pas$, gives a maximum value of $329m$ while the lower viscosity solution, $\mu = 4.5 \times 10^6 Pas$, reaches a peak at $467m$. The run-up is seen to occur at approximately the same time in all cases.

The final sensor data available from the experiment are the landslide thickness measurements over time as it passes $(-67, 67)m$, just before entering the water. A comparison between the two numerical simulations and the experimental results is presented in Figure 5.26. The viscosity difference is seen to have little impact on the shape of the landslide, which is broadly similar in the numerical and experimental cases. However, the experimental landslide appears to have travelled faster, completely passing the sensor point

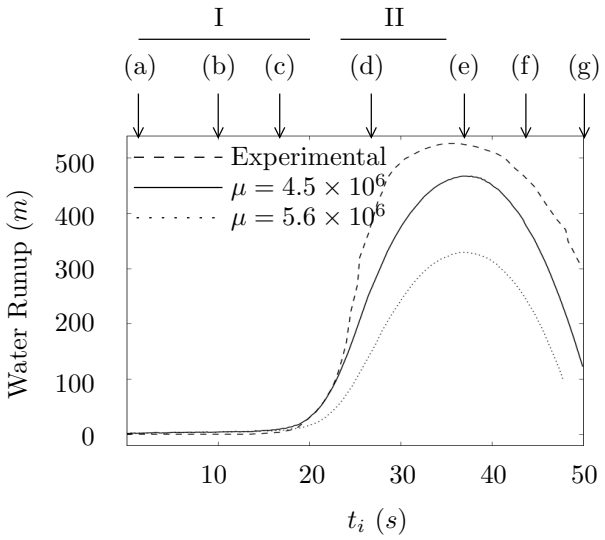


Figure 5.25: Data from the wave run-up gauge on the SW shoreline plotted against time after the landslide had impacted into the water, t_i. Numerical results for landslide viscosities of $4.5 \times 10^6 Pas$ (solid line) and $5.6 \times 10^6 Pas$ (dotted line) are shown alongside the experimental findings of Fritz *et al.* (2001) (dashed line). Letters (a–g) above the graph indicate the times corresponding to each frame in Figure 5.23. Roman numerals I and II indicate the time periods covered by the close-up images in Figures 5.27–5.28 and Figures 5.29–5.30 respectively.

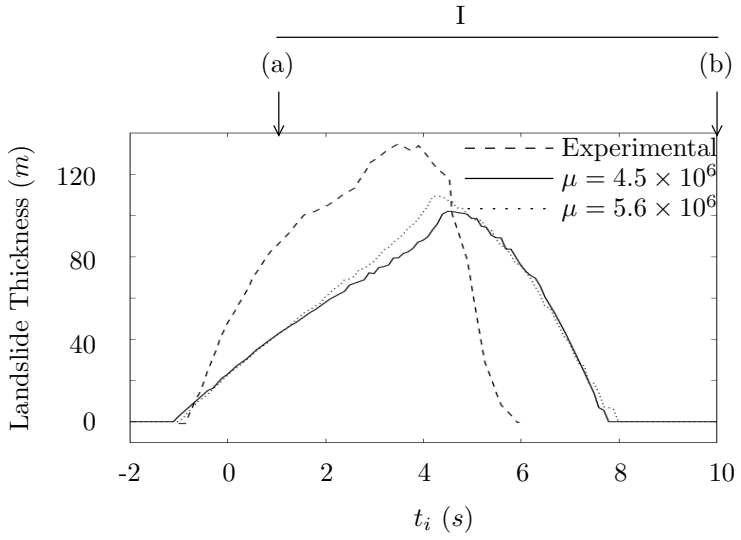


Figure 5.26: Data from landslide thickness sensor at $(-67, 67)m$ on the NE shoreline plotted against time after the landslide had impacted into the water, t_i. Numerical results for landslide viscosities of $4.5 \times 10^6 Pas$ (solid line) and $5.6 \times 10^6 Pas$ (dotted line) are shown alongside the experimental findings of Fritz *et al.* (2001) (dashed line). Letters (a, b) above the graph indicate the times corresponding to each frame in Figure 5.23. The roman numeral I indicates the time period covered by the close-up images in Figures 5.27–5.28.

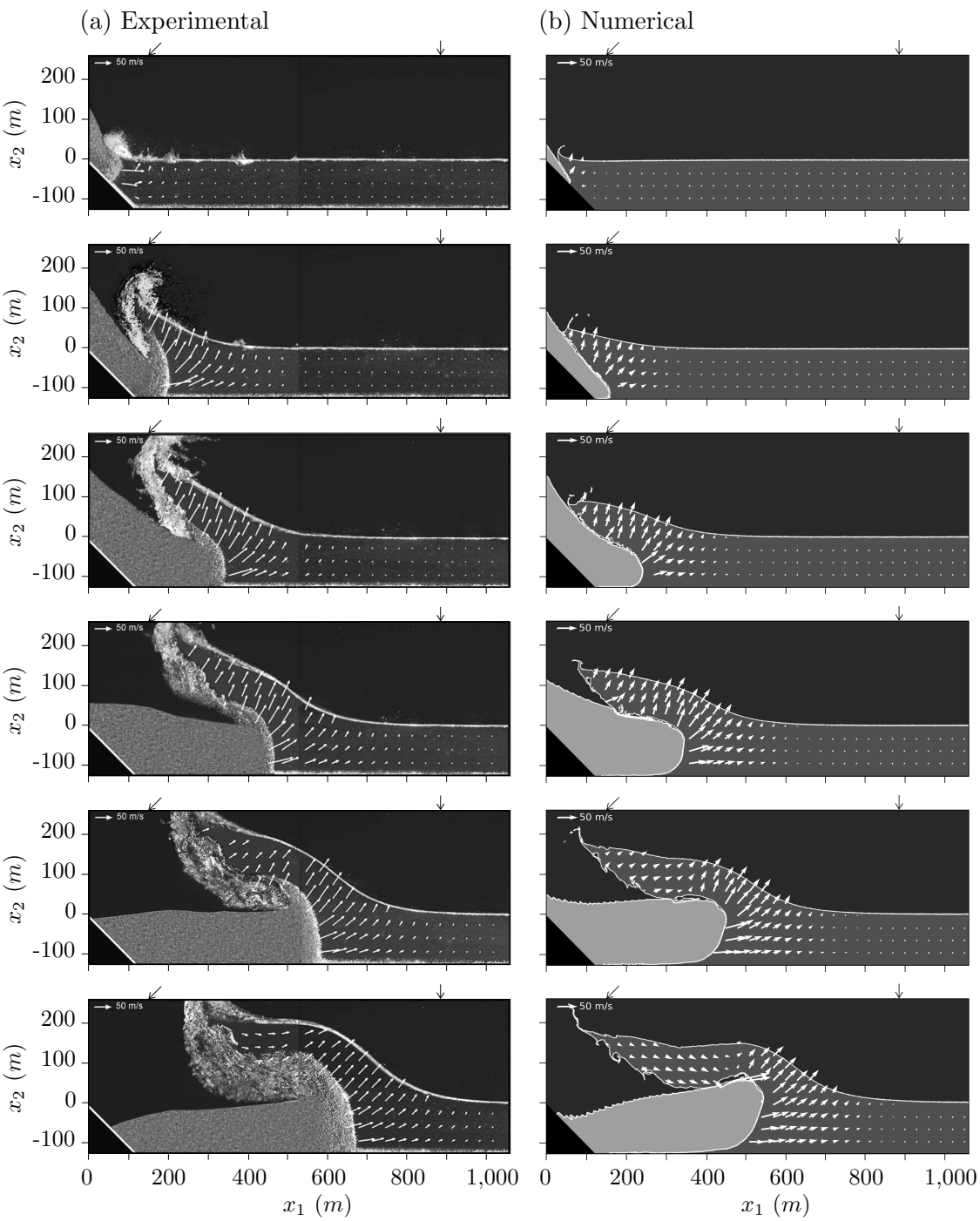


Figure 5.27: Close-up timeframes of the region around the landslide impact site from the experiment of Fritz *et al.* (2009) (a) and from the numerical result using a landslide viscosity of $4.5 \times 10^6 Pas$ (b). Contours are shown at 0.025, 0.5 and 0.975 of the water and landslide volume fractions. The white arrows are velocity vectors in the water measured every $50m$ in the experiment using particle image velocimetry and sampled on a similar scale from the numerical results. A velocity scale is given in the top left which corresponds to $50m\,s^{-1}$. The top image is from $0.76s$ after the initial impact of the landslide with the water and each subsequent image is spaced $1.73s$ apart. The diagonal arrow on top of each image indicates the direction of the line of sight of the landslide thickness sensor while the vertical arrow is at the wave gauge at $885m$.

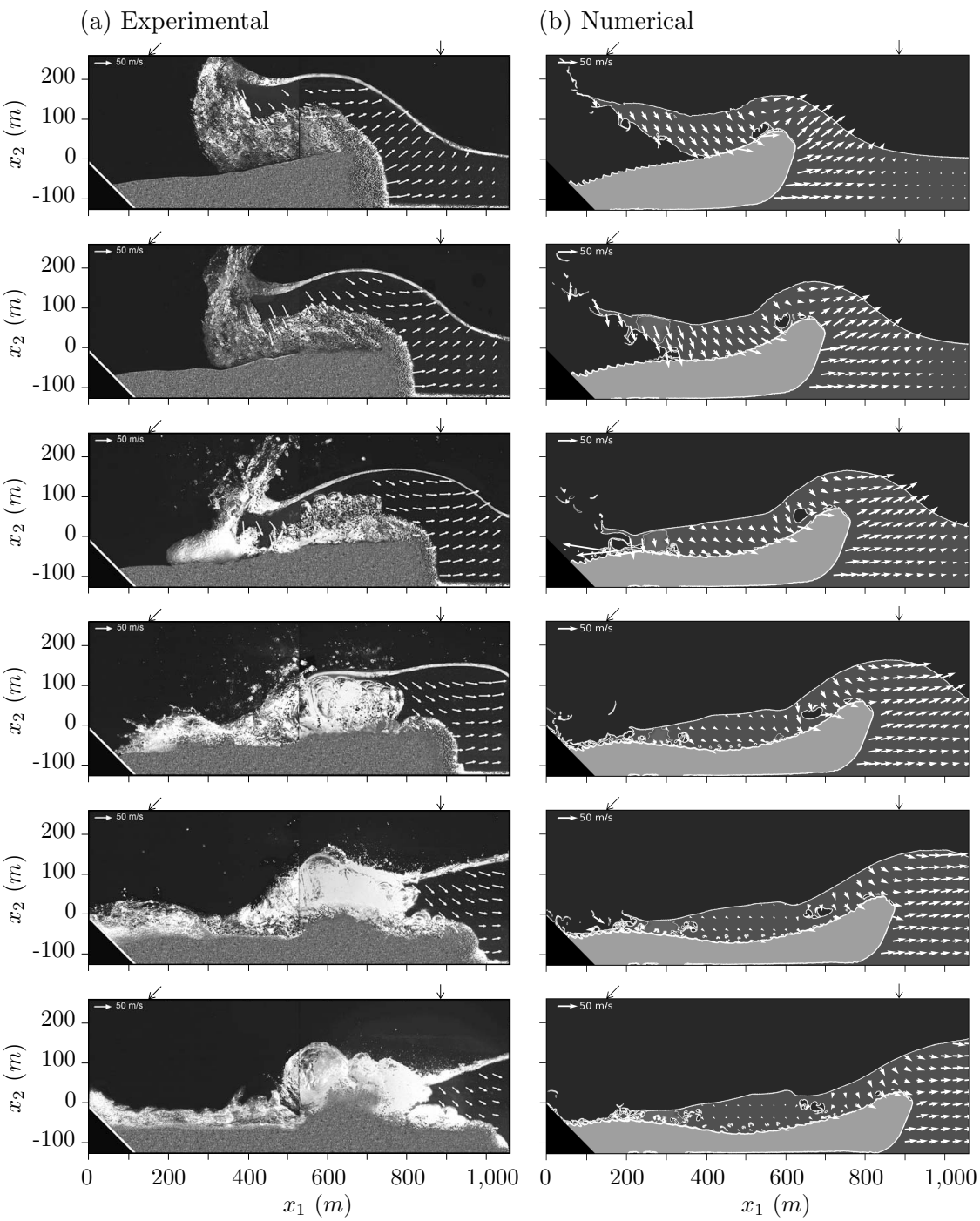


Figure 5.27: *cont.* Close-up timeframes of the region around the landslide impact site from the experiment of Fritz *et al.* (2009) (a) and from the numerical result using a landslide viscosity of $4.5 \times 10^6 Pas$ (b). Contours are shown at 0.025, 0.5 and 0.975 of the water and landslide volume fractions. The white arrows are velocity vectors in the water measured every $50m$ in the experiment using particle image velocimetry and sampled on a similar scale from the numerical results. A velocity scale is given in the top left which corresponds to $50m\,s^{-1}$. The top image is from $11.14s$ after the initial impact of the landslide with the water and each subsequent image is spaced $1.73s$ apart. The diagonal arrow on top of each image indicates the direction of the line of sight of the landslide thickness sensor while the vertical arrow is at the wave gauge at $885m$.

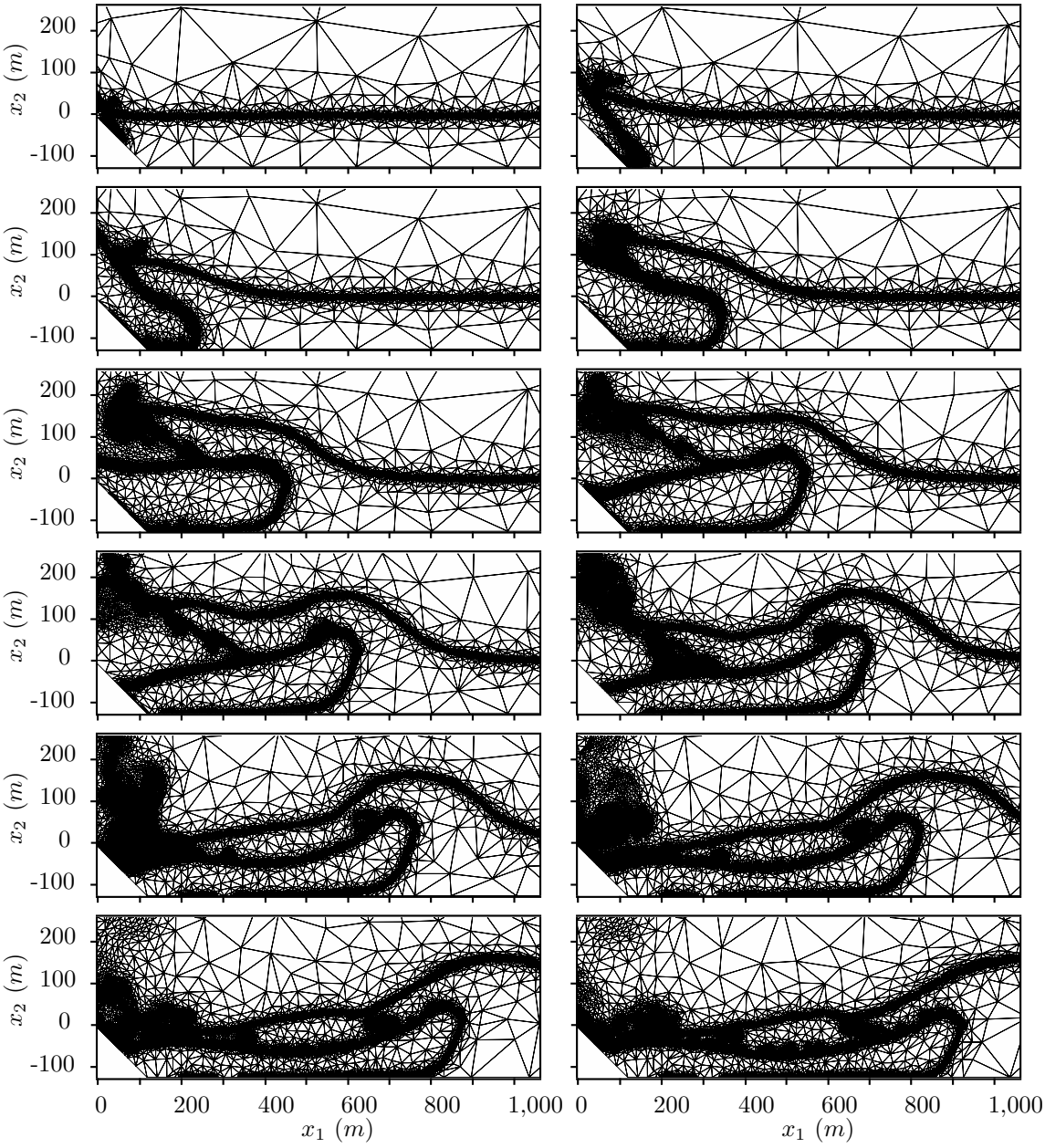


Figure 5.28: The adapted meshes in the vicinity of the impact site corresponding to the timeframes from the numerical results shown in Figure 5.27(b).

$6s$ after impact. The numerical simulations take until approximately $8s$ after impact to pass the sensor.

Several timeframes from the area around the impact site are shown for a landslide viscosity of $4.5 \times 10^6 Pas$ in Figure 5.27, alongside the images of the same area taken during the experiment (Fritz *et al.*, 2001). Close qualitative agreement can be seen between the two sets of results as the landslide enters the water and the air cavity opens behind it. Furthermore the velocity vectors, and hence the water movements recorded during the experiment, appear to be in close agreement throughout the time series. Initially the water is seen to have a significant upward component of motion, which gradually becomes more horizontally dominated as the wave propagates away from the impact site. Again however, the landslide is seen to travel slightly more slowly into the water in comparison to the experimental results. As discussed earlier, the high degree of mixing seen in the experiments is not reproducible in the numerical model, however the collapse of the air cavity happens simultaneously in both cases. Figure 5.28 shows the corresponding adapted meshes used at each timeframe of Figure 5.27, demonstrating how the high resolution zones closely follow the interfaces and, in the later stages, the areas of turbulent flow.

Figure 5.29 presents a series of close-up images of the SW shore during the period of maximum run-up, from $t_i = 23.28s$ to $t_i = 39.79s$, for the numerical simulation with a landslide viscosity of $4.5 \times 10^6 Pas$ and for the experimental results presented by Fritz *et al.* (2001). A close qualitative agreement between the two datasets is seen with similar timings of the surges up the slope. Additionally the velocity vectors show the same bifurcation behaviour as noted by Fritz *et al.* (2001) at the beginning of the return flow, with lower regions beginning to travel downwards while upper regions are still propagating upwards. As discussed before, the nascent breaking behaviour seen in the experiment is not reproduced in the numerical results leading to an apparently thicker surge of water up the slope. This may explain the slightly lower run-up values achieved in the numerical simulations. Figure 5.30 shows the corresponding adapted meshes for the timeframes in Figure 5.29, demonstrating that the high resolution zone closely tracks the interface between the air and water.

Overall the numerical simulations of the 1958 Lituya Bay landslide are in close agreement with the experimental results. Discrepancies can be explained by the large uncer-

Figure 5.29: *See next page.* Close-up timeframes of the region around the SW shore during the period of maximum run-up from the experiment of Fritz *et al.* (2001) (a) and from the numerical result using a landslide viscosity of $4.5 \times 10^6 Pas$ (b). Contours are shown at 0.025, 0.5 and 0.975 of the water volume fraction. The white arrows are velocity vectors in the water measured every $50m$ in the experiment using particle image velocimetry and sampled on a similar scale from the numerical results. A velocity scale is given in the top left which corresponds to $50m\,s^{-1}$.

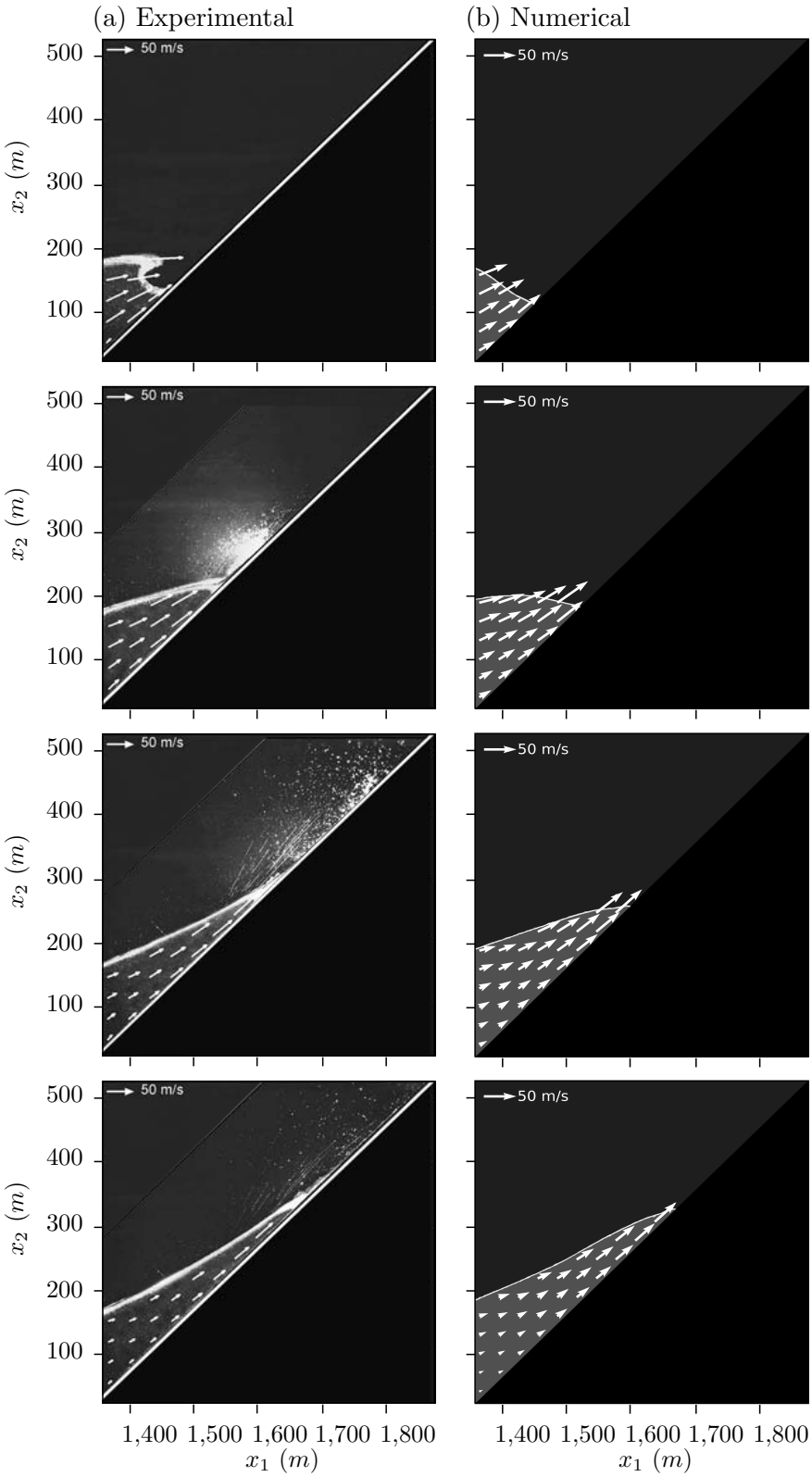


Figure 5.29: *cont.* The top image is from 23.28s after the initial impact of the landslide with the water and each subsequent image is spaced 1.73s apart.

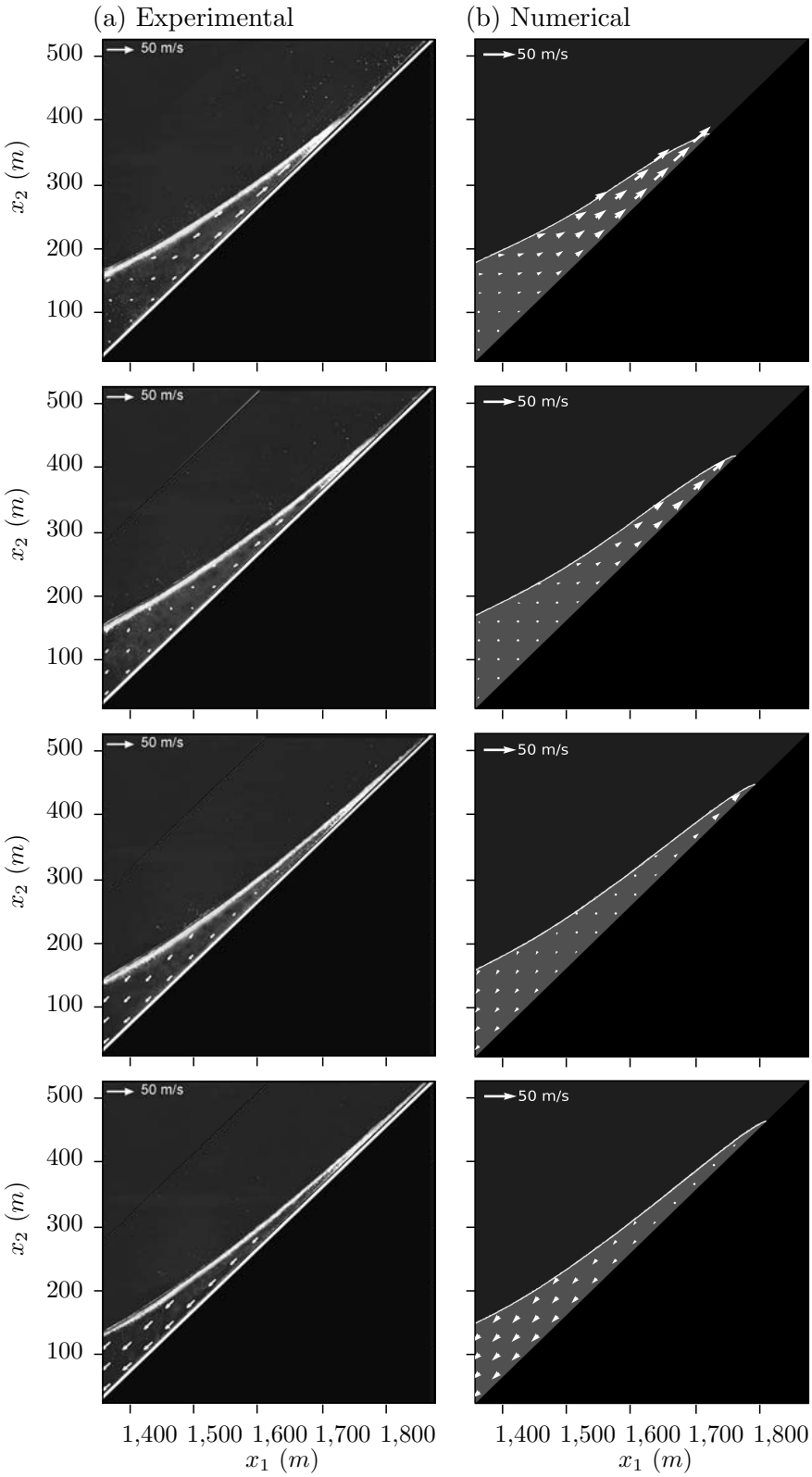


Figure 5.29: *cont.* The top image is from 30.2s after the initial impact of the landslide with the water and each subsequent image is spaced 1.73s apart.

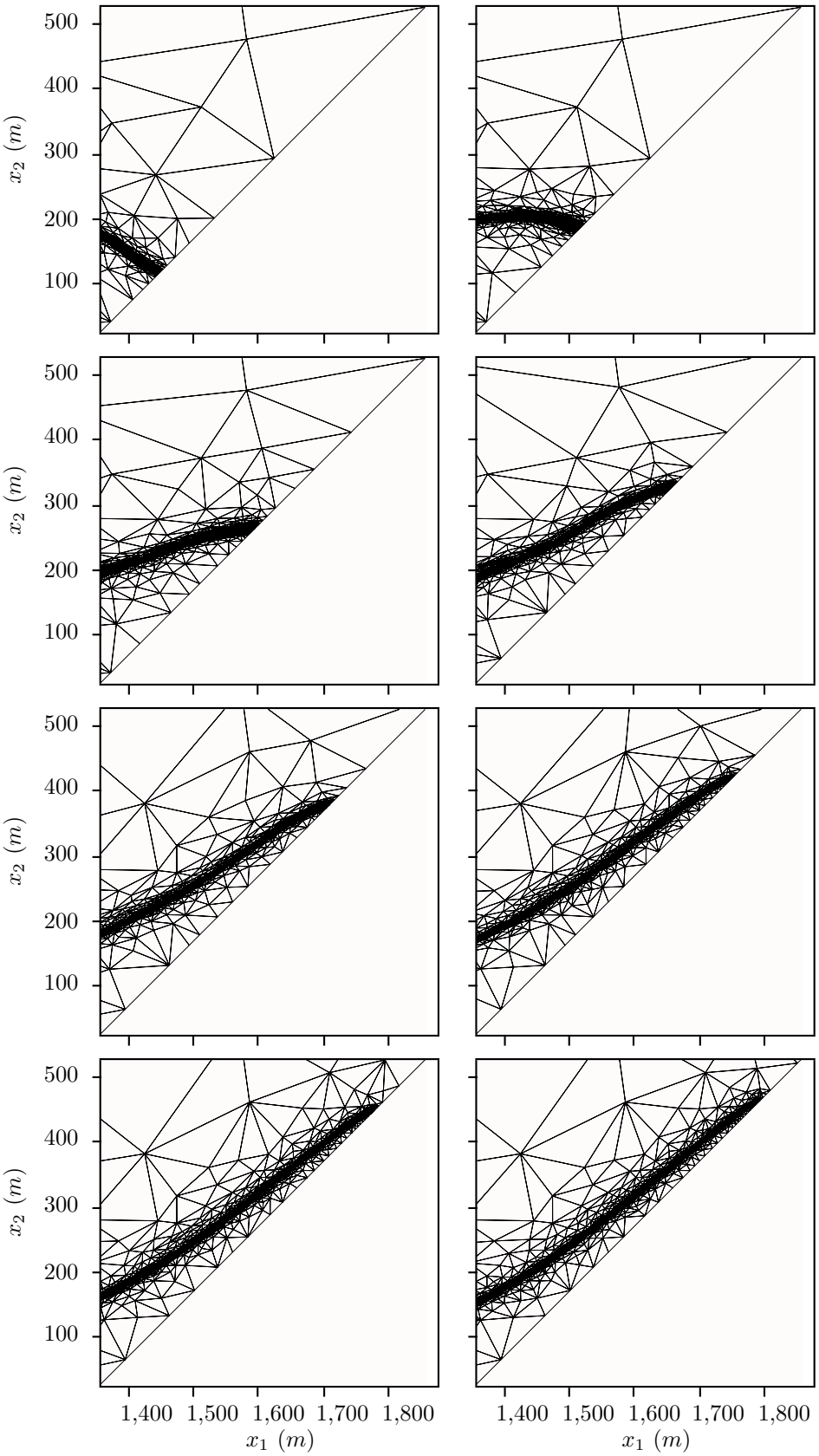


Figure 5.30: The adapted meshes in the region around the SW shore during the period of maximum run-up corresponding to the timeframes from the numerical results shown in Figure 5.29(b).

tainties in the model parameters. For example, although landslide shape in the numerical simulations was selected to achieve the correct centre of mass given in the experimental description, it is unclear whether this was still the correct shape after the experimental slide material had been accelerated and released into granular flow. Furthermore, the exact speed of this flow at the time of impact is not known. The thickening of the slide once it reaches the base of the domain is also influenced by bottom drag. This was approximated here as a no slip boundary condition but may be more accurately described using a frictional term instead. Any of these parameters of initial shape, velocity or bottom drag could explain the slower opening of the air cavity seen in the numerical simulations. This in turn may explain the smoother wave front seen and the lack of wave breaking during the period of run-up. Most importantly however for the purpose of this thesis, the multi-material model developed in the last three chapters was shown to have the essential features of stability, boundedness and conservation during a complex three material flow.

5.3 Summary

In this chapter the problem of material overlap leading to unbounded non-physical volume fractions during multi-material simulations was discussed. Using the onion skin approach with a prescribed priority ordering, a coupled flux limiter was developed based on the total variation bounded advection scheme UltraC, described in Chapter 3.

The coupled limiter was extended to multiple dimensions and demonstrated in a five material shearing flow problem. Due to the lower resolution per material the errors were found to be higher than in the two material case, however non-physical volume fractions were prevented and, in the fixed mesh case, materials preserved their relationships with their neighbours. These relationships broke down in the adaptive case because of the redistribution of material inherent in the bounded projection algorithm that interpolates from the old to the new mesh. Options for solving this were discussed however all were rejected due to difficulties preserving conservation or the interface sharpness during interpolation.

The multi-material model was then applied to the physical problem of landslide generated waves. First, a water-air model was shown to successfully reproduce both theoretical and experimental results of wave run-up on a sloping shoreline. Adding a third landslide material, an experimental model of the 1958 Lituya Bay landslide was then numerically simulated. This produced results in close qualitative and quantitative agreement with the experimental data, validating the multi-material model developed in this thesis.

In the next chapter the main conclusions of this thesis are summarised and possible future work, developing and using the multi-material model shown here, is discussed.

Chapter 6

Conclusions

In this thesis, a multiple material model suitable for the simulation of geophysical flows has been developed. This model meets the key requirements of any multi-material model: it conserves the materials, it maintains the bounds on the volume fractions and the pressure solution is stable. It was also specifically designed to take advantage of the flexibility provided by unstructured simplex meshes when discretising a domain. Unstructured simplex meshes allow complex geometries to be represented (Gorman *et al.*, 2006) as well as introducing the possibility of variable, anisotropic resolution. Combined with a suitable mesh optimisation algorithm (Vasilevski & Lipnikov, 1999; Pain *et al.*, 2001), the mesh can be designed both around the geometry of the domain and the fields represented on it. Repeatedly optimising or adapting the mesh throughout a simulation means that only the number of nodes required at any one time need be included. This increases efficiency, saving computer resources by reducing the memory required to represent the domain and increasing the speed of calculations by removing superfluous degrees of freedom.

Advecting different materials on any mesh is problematic because of numerical diffusion, which tends to smear interfaces into zones of non-physical mixed materials. Several schemes exist to minimise this diffusion and in one dimension it is possible to represent the step function behaviour between two materials optimally on the given resolution with a variety of algorithms. In multiple dimensions this becomes more difficult (Rudman, 1997) as the material can be smeared in different directions, making its movement harder to constrain. Additionally, the problem of corner coupling is introduced as cells in the mesh may no longer share faces with their neighbours. On element-centred, unstructured, simplex meshes corner coupling issues are increased by the arbitrary number of elements which intersect at a vertex. This also prevents the standard structured mesh solution of operator splitting from working on multi-dimensional simplex meshes.

In Section 3.3 the flux limited advection scheme HyperC (Leonard, 1991) was extended to multiple dimensions using a control volume mesh constructed using the centroid-based dual to a parent simplex mesh. For continuous piecewise linear and quadratic parent finite element meshes, the control volume mesh has the important property of sharing a face with all its nearest neighbours. This eliminates the problem of corner coupling, allowing information to propagate between neighbouring nodes directly. However, the problem of optimally defining a flux that minimises the numerical diffusion when advect-

ing in multiple directions at once is harder to solve. A 'worst-case scenario' approach was taken here that considered the maximum possible flux in and out of an entire cell at once. This is necessary in some situations where it is required to guarantee boundedness, but is over-diffusive in other scenarios. Future work should consider methods of optimally defining the limited downwind solution in multiple dimensions so that boundedness can be guaranteed while not introducing excessive diffusion.

One method of minimising diffusion was suggested in Section 3.4 in the form of the UltraC advection algorithm. By extending the definition of boundedness in HyperC from total variation diminishing to total variation bounded, this applies interface sharpening downwinding at more faces, decreasing the number of non-physical mixed cells generated during advection. Some of the adverse effects of this, such as the generation of flotsam and jetsam, can be counteracted through the use of a dual scheme, combining UltraC and HyperC. Despite reducing numerical diffusion, such schemes still suffered from poor shape preservation, distorting the interface. Future work on UltraC should examine the circumstances under which this wrinkling occurs. Previous attempts at using flux limiters in material advection have often introduced a switch to less compressive schemes when the interface and advection direction are aligned (Lafaurie *et al.*, 1994; Ubbink & Issa, 1999). Switching between UltraC and HyperC in this manner may improve the shape preservation qualities of the material advection.

Another way of minimising the generation of non-physical mixtures of material is to locally refine the mesh in the vicinity of the interface. This approach was examined in Section 3.5. Using a simple adaptive approach, some interaction between the anisotropic mesh and the advection algorithm was noted which led to distortion of the interface. This resulted because the mesh optimisation was based on the current state of the volume fraction rather than its anticipated state up to the next mesh adaptation. A combination of metric advection and edge length gradation substantially improved the results allowing similar errors to be achieved as in a fixed mesh but using a quarter of the number of nodes. Future work should examine the interplay between the mesh topology and the advection further, determining the optimal adaptivity settings that minimise both interface distortion and diffusion. The inclusion of coupling tensor twisting terms in the metric advection algorithm could also improve the predictive capabilities of the mesh optimisation. In Section 3.5 it was also suggested that only a minimally dissipative, bounded and conservative projection algorithm (Farrell *et al.*, 2009) was suitable for the transfer of volume fraction information between meshes. The algorithm used for this iteratively diffuses the solution, which makes it more expensive than alternative bounded, but not necessarily minimally dissipative or conservative, interpolation mechanisms. Methods of improving the efficiency of this method should be considered. For instance, mesh optimisation and subsequent data interpolation may be restricted to certain regions that fail some desired error tolerance rather than always interpolating over the entire mesh (Hassan *et al.*, 2000). Alternatively, in some situations mesh movement algorithms may allow the length of time between mesh optimisations to be increased, reducing the amount of interpolation required.

Having devised suitable schemes that guaranteed boundedness of the volume fraction

on unstructured, adaptive, simplex meshes, Chapter 4 considered the combined issues of pressure stability and material conservation for incompressible flows. A discretisation of velocity and pressure was sought that gave stability, in the LBB sense (Ladyzhenskaya, 1963; Babuška, 1971; Brezzi, 1974), so that the pressure solution would not blow up or become contaminated by numerical artefacts. Several suitable element pairs were implemented and tested, however, in order to achieve material conservation, it was necessary to define the divergence on the dual control volume mesh. As the divergence and gradient operators are the transpose of each other (Gresho & Sani, 1987) this meant that the pressure had to be discretised using control volumes. This led to the development of several modified element pairs, however, only one, $\mathrm{P0P1_{CV}}$, was found to pass simple force balance and convergence tests. This analysis was limited to incompressible flows for the purpose of simplicity. In the future, this should be generalised to incorporate compressible behaviour. This can be achieved by modifying the pressure equation to project the velocity to a specified rather than zero divergence subspace based on the full continuity equation.

The $\mathrm{P0P1_{CV}}$ element pair, coupled to HyperC and UltraC-S, was used to simulate the incompressible collapse of water columns in air, in two and three dimensions. With the exception of a few discrepancies, which were also seen in other models (Lee *et al.*, 2002; Park *et al.*, 2009; Colagrossi & Landrini, 2003), the numerical results were in close agreement with the experimental data (Zhou *et al.*, 1999; Kleefsman *et al.*, 2005). In two dimensions an adaptive mesh was able to reproduce comparable results to a fixed mesh run with the same minimum edge length while consistently using fewer nodes. Focussing the resolution around the interface, rather than in prescribed, fixed zones, also allowed the generation of mixed material cells to be minimised. Further investigation of some of the discrepancies between the numerical and experimental results would benefit from more information about the experimental set-up. It is also clear that some of the disagreements seen were a result of numerical simplifications. First-order upwinding is highly dissipative introducing large amounts of numerical viscosity (LeVeque, 1992). It was used in the momentum equation to allow easy generalisation when investigating stable element pairs. Having selected $\mathrm{P0P1_{CV}}$ alternative advective fluxes could be considered. This may improve the reduced water column head speeds when compared with the experiments. Similarly, although not used in the water column collapse simulations, the viscosity discretisation method was also selected for its applicability to all discontinuous Galerkin schemes. Alternative, finite volume discretisations (Mathur & Murthy, 1997) should be implemented.

In Chapter 5 the issue of materials overlapping during advection was discussed. As both operator splitting and interface reconstruction were ruled out on the unstructured mesh a coupled flux limiter was developed. This was based on the total variation bounded scheme used in Chapter 3 to develop UltraC. Given a bounded material advection scheme, such as HyperC, and a priority ordering in which the materials are to be advected, the coupled limiter checks whether the control volume face value assigned by the scheme will cause the sum of the materials to become unbounded. If it does then its value will be limited, either increasing or decreasing the flux by the minimal amount, to re-

gain bounded behaviour. When demonstrated in this thesis the coupled limiter used a prescribed priority ordering. Alternative methods of allowing each cell to optimally decide the material priority should be implemented. This is difficult without interface reconstruction, however simple methods such as ordering based on the content of the downwind cells (McGlaun *et al.*, 1989) could be investigated. It would also be interesting to determine if the coupled flux limiter could be extended to multi-phase flows, where each phase is advected by an independent velocity field. This would require taking into account the relative velocities of successive phases in order to determine the correct Courant based slope to be used in the limiter boundaries. If possible, this could allow a multi-material/phase model to be developed where inter-penetrating phases could be modelled alongside immiscible materials.

Modelling multiple materials using adaptive meshes not only requires the coupling of the materials during advection but also during interpolation. The minimally diffusive bounded projection algorithm (Farrell *et al.*, 2009) was modified to make a coupled bounded projection method. This enforces the same priority ordering used in advection for the interpolation and ensures that the maximum smoothed bound of the field is also aware of the fields that have already been interpolated. A five material test of this algorithm showed that it introduced 'relationship errors,' separating fields during the interpolation that were neighbours on the original mesh. This is a property of the redistribution inherent in the bounding scheme. Interpolation and projection algorithms that do not redistribute material do not exhibit this problem. However, those investigated here are either too diffusive, not conservative, or both. Alternative strategies for preserving the bounds of all the volume fractions and their local relationships should be investigated.

The ability of the full multi-material model to produce a physically realistic simulation of a landslide generated wave was tested in Section 5.2. Using a two material water-air model, the run-up of a solitary wave on a sloping coastline was compared against theoretical (Synolakis, 1987) and experimental (Hall & Watts, 1953) data and found to be in close agreement. Introducing a viscous landslide as a third material, a two-dimensional cross section of the 1958 Lituya Bay landslide and subsequent wave run-up was modelled. Comparing this to the experimental data from Fritz *et al.* (2001), a close qualitative and quantitative agreement was found. Disagreements in the wave run-up were conjectured to have been caused by uncertainties in the experimental set-up and approximations in the numerical solution. Varying parameters such as the shape and initial velocity of the landslide along with the bottom drag may allow the discrepancies to be minimised.

Having demonstrated the capabilities of the model in physically relevant scenarios there is significant scope for its application to geophysical flows. In particular, it is already implemented in two and three dimensions, allowing a larger range of situations to be considered. The use of unstructured, adaptive meshes means that the switch to three dimensions is computationally much more manageable than using structured chessboard-like meshes. For example, DiRisio *et al.* (2009a) produced an experimental model of a $0.8 \times 0.4 \times 0.05m$ landslide running down the side of a $4.45m$ radius conical island,

generating a wave upon impact with a water layer. Equivalent to a simplified 1:1,000 model of Stromboli (South Tyrrhenian Sea, Italy), the experiment measured the wave run-ups on the coast of the island itself, behind the wave generation site. Despite being mostly concerned with the near field wave behaviour, the tank used in the experiment was large, $50 \times 30 \times 3m$, in comparison to the landslide, to prevent wave reflections contaminating the data. The data collected in this experiment offer an excellent three-dimensional validation tool of the multi-material model. Given the large domain and small scale features to be resolved (wave run-ups were on the scale of $1cm$) an adaptive mesh is ideally suited to reproducing the experiment. Once validated, a numerical model additionally offers the possibility of augmenting the experimental scenario, making it more physically relevant by adding representations of real bathymetry and topography. In essence, having demonstrated the capabilities of the model developed in this thesis it is possible to extend it to scenarios not reproducible in laboratory experiments.

The main conclusions of this thesis can be summarised as follows.

- The problem of corner coupling when advecting materials in multiple dimensions may be overcome on a simplex mesh by constructing the centroid dual control volume mesh.

- Using the control volume mesh, and without the need for operator splitting, flux limited material advection algorithms may be used in multiple dimensions. In particular, in standard advection tests, the HyperC algorithm produces errors comparable to the standard operator split and interface reconstruction based VOF schemes.

- The boundedness criterion for the flux limited solution may be adjusted to ensure a total variation bounded rather than a total variation diminishing property. This UltraC scheme reduces numerical diffusion but has poorer shape preserving qualities.

- Advecting volume fractions on adaptive meshes requires the use of a minimally diffusive bounded projection method that ensures the material is conserved between successive meshes. Convergence of this algorithm is not monotonic and an extra repair step is often required to guarantee strict bounds and to speed up the interpolation.

- Overly refined meshes, based on data from the previous timestep, tend to interact with advection causing deformation of the material interfaces. This may be alleviated by advecting the error metric forward in time, to anticipate the future movement of the materials in the mesh design. Edge length gradation is also beneficial, preventing the element sizes from diminishing too rapidly away from the interface.

- The use of adaptive meshes with material advection reduces the generation of non-physical mixed material cells and can reduce the number of nodes required to represent the solution to the same degree of accuracy.

- The conservation properties of a multi-material model are dependent on the consistent discretisation of the velocity-pressure system, which also affects the stability of the model. The $\mathrm{P0P1_{CV}}$ element pair is suitable both for the model to remain stable and to ensure conservation of the material advected on the control volume dual mesh.

- Using a two material water-air model it is possible to reproduce experimental dam break problems. In these simulations the adaptive mesh allows the interface to be tracked more accurately while using fewer nodes.

- It is possible to advect multiple volume fractions, while overcoming the problem of material overlap, using a coupled flux limiter and a material priority ordering.

- Interpolation methods that redistribute material after mesh optimisation lead to the separation of materials and the introduction of extra error in the material advection solution. Methods that do not redistribute tend to be too diffusive or not conservative and are therefore even less suitable for modelling physical systems.

- A three material model of water, air and a viscous landslide can be used to realistically simulate the generation of a landslide generated wave.

- There is significant scope for further development, validation and use of the adaptive, unstructured mesh multi-material model presented in this thesis.

Acknowledgements

First, I would like to thank my supervisors: Gareth Collins, Christopher Pain, Matthew Piggott, Alan Dawes and Graham Ball. Gareth's office door has been open to me throughout my Ph.D. and I have always been able to depend on him for sound advice and guidance. Chris has provided an endless source of enthusiasm for any numerical development and none of this would have been possible without his original vision for Fluidity. I have enjoyed numerous spirited conversations with Matthew, who acted as an excellent sounding board for some of my slightly crazier ideas. The trips to London, made by my two supervisors from AWE, Alan and Graham, provided a much needed external perspective on developments and I appreciate all the time and effort they contributed repeatedly coming to hear about the ongoing work. AWE's interest in this project, both through a CASE award and through their visits to Imperial, has undoubtedly enhanced the final thesis.

Beyond my official supervisors, I have received help and guidance from many other people around Imperial College. In particular, all the members (and associated pseudo-members) of the Applied Modelling and Computation Group (AMCG) have been an endless source of information on topics ranging from Fortran standards to scuba diving. This project would not have been possible without their combined input. On the numerical side, special mention must go to Stephan Kramer, David Ham, Colin Cotter, Patrick Farrell, James Maddison and Timothy Bond. Stephan has been endlessly patient with all my questions and has always been ready to drop everything to come and help with an interesting problem. David, through shear unstoppable force of will, started the regeneration of Fluidity without which this thesis would have been much harder to complete. Colin has answered all my questions with ease, never hesitating to set me straight, which has benefited my understanding immensely. Without the contributions Patrick and James made to the code very few of the results presented here would have been possible. Last but certainly not least, Tim was always been ready and enthusiastic to dig me out of whatever mess I'd made on my computer.

On a more social note I have made many friends in AMCG and at Imperial who, in combination with frequent pub, concert and theatre trips, have helped keep me sane. Hannah's generous supplies of soup, brownies and hannahjacks provided essential and delicious sustenance while writing up and, though I may not have always said it at the time, our weekly runs around Hyde park were great fun. Jemma was always ready to listen to my rants about some bit of code or another even after baby Tom came along. Rhodri bach's recent arrival added some Welsh spice to the group and an endless source

of amusement and conversation. Although I could never keep up on the dance floor, trips to the union with Adam were always a great opportunity to muse randomly about the future, one of my favourite ways to pass the time. Tom, my office mate for the last two years (most recently in the fantastic batcave pod), put up with endless disturbances caused by my random musings, outbursts and visitors but we always remained united in our dislike of squeaky shoes and loud meetings.

Ben, Veronica, Caroline and Stephan had the dubious honour of both working and living with me at some point over the last two years. How they ever put up with me all that time I'll never know but they all made my time in 21 Paton House hugely enjoyable, despite my initial reservations about the sofa upholstery. Beyond the office, during my time living at Goodenough College, I met a wonderful array of interesting people who helped settle me into life in London. In particular I want to thank Misha for always forgiving me when I disappeared for months at a time and Marissa for always stopping by London to have a chat on her many jet setting trips.

Away from London, I want to thank my parents for all their support over the last 27 years. They have always encouraged me to achieve my goals and without that lesson this thesis wouldn't have been possible. Recently they, and my sister, Tania, have also put up with an ever diminishing number of phone calls and visits as work completely took over my life. When I did get away from it I enjoyed hearing tales of Lile Ossick off my Grandad and I was always made to feel welcome in Ambleside by Elsie. Her spirit for life will always be inspirational to me.

Of course, no acknowledgements would be complete without thanking Jen, who has heard endlessly about Fortran, discretisations and simulations for the last four years without complaint. She has been an incredible source of support and without her this project would have driven me mad years ago.

Bibliography

Ahn, H. T., & Shashkov, M. 2009. Adaptive moment-of-fluid method. *Journal of Computational Physics*, **228**(8), 2792–2821.

Amsden, A. 1980. *SALE: A simplified ALE computer program for fluid flow at all speeds.* Tech. rept. LA-8095. Los Alamos National Laboratories.

Arakawa, A., & Lamb, V. R. 1977. Computational design of the basic dynamical processes of the UCLA general circulation model. *In: General circulation models of the atmosphere.* New York, Academic Press, Inc.

Arnold, D., Brezzi, F., & Fortin, M. 1984. A stable finite element for the stokes equations. *Calcolo*, **21**(4), 337–344.

Arnold, D. N., Brezzi, F., Cockburn, B., & Marini, D. 2000. Discontinuous Galerkin methods for elliptic problems. *Lecture Notes in Computational Science and Engineering*, **11**, 89–102.

Babuška, Ivo. 1971. Error-bounds for finite element method. *Numerische Mathematik*, **16**(4), 322–333.

Balay, S., Gropp, W. D., McInnes, L. C., & Smith, B. F. 1997. Efficient Management of Parallelism in Object Oriented Numerical Software Libraries. *Pages 163–202 of:* Arge, E., Bruaset, A. M., & Langtangen, H. P. (eds), *Modern Software Tools in Scientific Computing.* Birkhuser Press.

Baliga, B. R., & Patankar, S. V. 1998. *Handbook of numerical heat transfer.* John Wiley & Sons, Inc.

Bassi, F., & Rebay, S. 1997. A High-Order Accurate Discontinuous Finite Element Method for the Numerical Solution of the Compressible Navier-Stokes Equations. *Journal of Computational Physics*, **131**(2), 267–279.

Batchelor, G. K. 2000. *An introduction to fluid dynamics.* Cambridge University Press.

Bell, R. L., & Hertel, E. S. 1992. *An improved material interface reconstruction algorithm for Eulerian codes.* Tech. rept. SAND 92-1716. Sandia National Laboratories, New Mexico.

Belytschko, T., Liu, W. K., & Moran, B. 2000. *Nonlinear finite elements for continua and structures.* Chichester: John Wiley.

Benson, D. J. 1998. Eulerian finite element methods for the micromechanics of heterogeneous materials: Dynamic prioritization of material interfaces. *Computer Methods in Applied Mechanics and Engineering*, **151**(3-4), 343–360.

Benson, D. J. 2002. Volume of fluid interface reconstruction methods for multi-material problems. *Applied Mechanics Reviews*, **55**(2), 151–165.

Booij, N., Holthuijsen, L. H., & Ris, R. C. 1996. The SWAN wave model for shallow water. *Pages 668–676 of: Coastal Engineering Conference*, vol. 1. ASCE American Society of Civil Engineers.

Brennen, C. E. 2005. *Fundamentals of Multiphase Flow.* Cambridge University Press.

Brezzi, F. 1974. On the existence, uniqueness and approximation of saddle-point problems arising from Lagrangian multipliers. *Rev. Francaise Automat. Informat. Recherche Operationnelle Ser. Rouge*, **8**, 129151.

Choi, B. Y., & Bussmann, M. 2007. A piecewise linear approach to volume tracking a triple point. *International Journal for Numerical Methods in Fluids*, **53**(6), 1005–1018.

Chorin, A. J. 1968. Numerical Solution of the Navier-Stokes Equations. *Mathematics of Computation*, **22**(104), 745–762.

Cockburn, B., & Shu, C.-W. 1998. The Runge-Kutta Discontinuous Galerkin Method for Conservation Laws V: Multidimensional Systems. *Journal of Computational Physics*, **141**(2), 199–224.

Colagrossi, A., & Landrini, M. 2003. Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *Journal of Computational Physics*, **191**(2), 448–475.

Collins, G. S., & Melosh, H. J. 2003. Acoustic fluidization and the extraordinary mobility of sturzstroms. *Journal of Geophysical Research*, **108**(B10), 2473.

Collins, G. S., Melosh, H. J., & Ivanov, B. A. 2004. Modeling damage and deformation in impact simulations. *Meteoritics and Planetary Science*, **39**(2), 217–232.

Cools, R. 1999. Monomial cubature rules since 'Stroud': a compilation – part 2. *Journal of Computational and Applied Mathematics*, **112**(1-2), 21 – 27.

Cools, R. 2003. An encyclopaedia of cubature formulas. *Journal of Complexity*, **19**(3), 445 – 453. Oberwolfach Special Issue.

Cools, R., & Rabinowitz, P. 1993. Monomial cubature rules since Stroud: a compilation. *Journal of Computational and Applied Mathematics*, **48**(3), 309 – 326.

Cotter, C. J., Ham, D. A., Pain, C. C., & Reich, S. 2009a. LBB stability of a mixed Galerkin finite element pair for fluid flow simulations. *Journal of Computational Physics*, **228**(2), 336–348.

Cotter, C. J., Ham, D. A., & Pain, C. C. 2009b. A mixed discontinuous/continuous finite element pair for shallow-water ocean modelling. *Ocean Modelling*, **26**(1-2), 86–90.

Courant, R., Friedrichs, K., & Lewy, H. 1959. On the partial difference equations of mathematical physics. *IBM J. Res. Develop.*, **11**, 215–234. English translation of an article originally published in German in Math. Ann. 100 (1928), 32-74.

Crouzeix, M., & Raviart, P. A. 1973. Conforming and nonconforming finite element methods for solving the stationary Stokes equations. *Rev. Francaise Automat. Informat. Recherche Operationelle Ser. Rouge Anal. Numer*, **7**, 3376.

Daily, J. W., & Stephan, S. C. 1952. The solitary wave. Its celerity, profile, internal velocities and amplitude attenuation in a horizontal smooth channel. *Page 1330 of: Proc. of the 3rd Conf. Coastal Eng., Univ. of California Berkley.*

Daly, B. J., Harlow, F. H., Shannon, J. P., & Welch, J. E. 1966. *The MAC method – A computing technique for solving viscous, incompressible, transient fluid-flow problems involving free surfaces.* Tech. rept. LA-3425. Los Alamos National Laboratories.

Darwish, M., & Moukalled, F. 2006. Convective Schemes for Capturing Interfaces of Free-Surface Flows on Unstructured Grids. *Numerical Heat Transfer, Part B: Fundamentals*, **49**(1), 19.

Darwish, M. S., & Moukalled, F. 2003. TVD schemes for unstructured grids. *International Journal of Heat and Mass Transfer*, **46**(4), 599–611.

Després, B., & Lagoutière, F. 2001. Contact Discontinuity Capturing Schemes for Linear Advection and Compressible Gas Dynamics. *Journal of Scientific Computing*, **16**(4), 479–524.

DiRisio, M., DeGirolamo, P., Bellotti, G., Panizzo, A., Aristodemo, F., Molfetta, M. G., & Petrillo, A. F. 2009a. Landslide-generated tsunamis runup at the coast of a conical island: New physical model experiments. *Journal of Geophysical Research*, **114**.

DiRisio, M., Bellotti, G., Panizzo, A., & DeGirolamo, P. 2009b. Three-dimensional experiments on landslide generated waves at a sloping coast. *Coastal Engineering*, **56**(5-6), 659–671.

Donéa, J., & Huerta, A. 2003. *Finite element methods for flow problems.* John Wiley and Sons.

Dritschel, D. G., Polvani, L. M., & Mohebalhojeh, A. R. 1999. The Contour-Advective Semi-Lagrangian Algorithm for the Shallow Water Equations. *Monthly Weather Review*, **127**(7), 1551–1565.

E, W., & Liu, J.-G. 1996. Projection Method II: Godunov-Ryabenki Analysis. *SIAM Journal on Numerical Analysis*, **33**(4), 1597–1621.

Eberly, D. H. 2001. *3D game engine design*. Morgan Kaufmann.

Eisenstat, S. C. 1981. Efficient Implementation of a Class of Preconditioned Conjugate Gradient Methods. *SIAM Journal on Scientific and Statistical Computing*, **2**(1), 1–4.

Elman, H. C., Silvester, D. J., & Wathen, A. J. 2005. *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*. OUP Oxford.

Fabri, A., Giezeman, G. J., Kettner, L., Schirra, S., & Schönherr, S. 2000. On the design of CGAL a computational geometry algorithms library. *Software Practice and Experience*, **30**(11), 11671202.

Farrell, P.E., Piggott, M.D., Pain, C.C., Gorman, G.J., & Wilson, C.R. 2009. Conservative interpolation between unstructured meshes via supermesh construction. *Computer Methods in Applied Mechanics and Engineering*, **198**(33-36), 2632–2642.

Fine, I.V., Rabinovich, A.B., Bornhold, B.D., Thomson, R.E., & Kulikov, E.A. 2005. The Grand Banks landslide-generated tsunami of November 18, 1929: preliminary analysis and numerical modeling. *Marine Geology*, **215**(1-2), 45–57.

Fritz, H. M., & Moser, P. 2003. Pneumatic landslide generator. *International Journal of Fluid Power*, **4**(1), 49–57.

Fritz, H. M., Hager, W. H., & Minor, H. E. 2001. Lituya Bay case: Rockslide impact and wave run-up. *Science of Tsunami Hazards*, **19**(1), 3–22.

Fritz, H. M., Mohammed, F., & Yoo, J. 2009. Lituya Bay Landslide Impact Generated Mega-Tsunami 50th Anniversary. *Pure and Applied Geophysics*, **166**(1), 153–175.

Gaskell, P. H., & Lau, A. K. C. 1988. Curvature-compensated convective transport: SMART, A new boundedness- preserving transport algorithm. *International Journal for Numerical Methods in Fluids*, **8**(6), 617–641.

Gisler, G., Weaver, R., & Gittings, M. L. 2006. Sage calculations of the tsunami threat from La Palma. *Science of Tsunami Hazards*, **24**(4), 288.

Gjevik, B., & Pedersen, G. 1981. Run-up of long waves on an inclined plane. *Preprint Series University of Oslo*, **2**, 82–553.

Gorman, G.J., Piggott, M.D., Pain, C.C., de Oliveira, C.R.E., Umpleby, A.P., & Goddard, A.J.H. 2006. Optimisation based bathymetry approximation through constrained unstructured mesh adaptivity. *Ocean Modelling*, **12**(3-4), 436–452.

Gorman, G.J., Piggott, M.D., & Pain, C.C. 2007. Shoreline approximation for unstructured mesh generation. *Computers & Geosciences*, **33**(5), 666–677.

Greaves, D. M. 2006. Simulation of viscous water column collapse using adapting hierarchical grids. *International Journal for Numerical Methods in Fluids*, **50**(6), 693–711.

Gresho, P. M. 1990. On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. Part 1: Theory. *International Journal for Numerical Methods in Fluids*, **11**(5), 587–620.

Gresho, P. M., & Sani, R. L. 1987. On pressure boundary conditions for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, **7**(10), 1111–1145.

Gresho, P. M., Sani, R. L., & Engelman, M. S. 1998. *Incompressible flow and the finite element method.* Chichester: Wiley, 2000.

Hall, J. V., & Watts, J. W. 1953. Laboratory investigation of the vertical rise of solitary waves on impermeable slopes. *US Army, Corps of Engrs, Beach Erosion Board, Tech. Memo*, **33**.

Ham, D. A., Pietrzak, J., & Stelling, G. S. 2005. A scalable unstructured grid 3-dimensional finite volume model for the shallow water equations. *Ocean Modelling*, **10**(1-2), 153–169.

Ham, D. A., Farrell, P. E., Gorman, G. J., Maddison, J. R., Wilson, C. R., Kramer, S. C., Shipton, J., Collins, G. S, Cotter, C. J., & Piggott, M. D. 2009. Spud 1.0: generalising and automating the user interfaces of scientific computer models. *Geoscientific Model Development*, **2**(1), 3342.

Hassan, O., Probert, E. J., Morgan, K., & Weatherill, N. P. 2000. Unsteady flow simulation using unstructured meshes. *Computer Methods in Applied Mechanics and Engineering*, **189**(4), 1247–1275.

Heitner, K. L., & Housner, G. W. 1970. Numerical model for tsunami run-up. *Journal of the Waterways, Harbors and Coastal Engineering Division*, **96**(3), 701–719.

Hestenes, M. R., & Stiefel, E. 1952. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Stand*, **49**(6), 409–436.

Heywood, J. G., & Rannacher, R. 1982. Finite Element Approximation of the Nonstationary Navier-Stokes Problem. I. Regularity of Solutions and Second-Order Error Estimates for Spatial Discretization. *SIAM Journal on Numerical Analysis*, **19**(2), 275–311.

Heywood, J. G., & Rannacher, R. 1990. Finite-Element Approximation of the Nonstationary Navier-Stokes Problem Part IV: Error Analysis for Second-Order Time Discretization. *SIAM Journal on Numerical Analysis*, **27**(2), 353–384.

Hirt, C. W., & Nichols, B. D. 1981. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, **39**(1), 201–225.

Hu, X.Y., & Adams, N.A. 2006. A multi-phase SPH method for macroscopic and mesoscopic flows. *Journal of Computational Physics*, **213**(2), 844–861.

Hughes, T. J. R. 2000. *The finite element method : linear static and dynamic finite element analysis*. Mineola, N.Y.: Dover Publications.

Ippen, A. T., & Kulin, G. 1955. *Shoaling and breaking characteristics of the solitary wave*. Tech. rept. Massachusetts Institute of Technology Hydrodynamics Laboratory, Cambridge.

Jasak, H., Weller, H.G., & Gosman, A.D. 1999. High resolution NVD differencing scheme for arbitrarily unstructured meshes. *International Journal for Numerical Methods in Fluids*, **31**(2), 431–449.

Jun, B.-I. 2000. A Simple Advection Scheme for Material Interface. *In: Nuclear Explosive's Code Developer's Conference*.

Karniadakis, G., & Sherwin, S. J. 1999. *Spectral/hp element methods for CFD*. Oxford University Press US.

Kim, S. K., Liu, P. L. F., & Liggett, J. A. 1983. Boundary integral equation solutions for solitary wave generation, propagation and run-up. *Coastal Engineering*, **7**(4), 299–317.

Kleefsman, K. M. T. 2005. *Water impact loading on offshore structures. - A numerical study*. Ph.D. thesis, Rijksuniversiteit Groningen.

Kleefsman, K. M. T., Fekken, G., Veldman, A. E. P., Iwanowski, B., & Buchner, B. 2005. A Volume-of-Fluid based simulation method for wave impact problems. *Journal of Computational Physics*, **206**(1), 363–393.

Ladyzhenskaya, O. A. 1963. *The mathematical theory of viscous incompressible flow*. Gordon and Breach.

Lafaurie, B., Nardone, C., Scardovelli, R., Zaleski, S., & Zanetti, G. 1994. Modelling Merging and Fragmentation in Multiphase Flows with SURFER. *Journal of Computational Physics*, **113**(1), 134–147.

Laitone, E. V. 1960. The Second Approximation to Cnoidal and Solitary Waves. *Journal of Fluid Mechanics Digital Archive*, **9**(03), 430–444.

Lakehal, D., Meier, M., & Fulgosi, M. 2002. *Interface tracking towards the direct simulation of heat and mass transfer in multiphase flows*. Vol. 23. Elsevier Science Inc.

Lee, T.-H., Zhou, Z., & Cao, Y. 2002. Numerical Simulations of Hydraulic Jumps in Water Sloshing and Water Impacting. *Journal of Fluids Engineering*, **124**(1), 215–226.

Leonard, B. P. 1988. Simple high-accuracy resolution program for convective modelling of discontinuities. *International Journal for Numerical Methods in Fluids*, **8**(10), 1291–1318.

Leonard, B. P. 1991. The ULTIMATE conservative difference scheme applied to unsteady one-dimensional advection. *Computer Methods in Applied Mechanics and Engineering*, **88**(1), 17–74.

LeRoux, D. Y., Rostand, V., & Pouliot, B. 2007. Analysis of Numerically Induced Oscillations in 2D Finite-Element Shallow-Water Models Part I: Inertia-Gravity Waves. *SIAM Journal on Scientific Computing*, **29**(1), 331.

LeVeque, R. J. 1992. *Numerical methods for conservation laws.* 2nd ed. edn. Lectures in mathematics ETH Zurich. Basel ; Boston: Birkhauser Verlag.

LeVeque, R. J. 2002. *Finite-volume methods for hyperbolic problems.* Cambridge: Cambridge University Press.

Li, X.-Y. 2003. Generating well-shaped d-dimensional Delaunay Meshes. *Theoretical Computer Science*, **296**(1), 145–165.

Mader, C., & Gitting, M. L. 2001. Modeling the 1958 Lituya Bay mega-tsunami. *Science of Tsunami Hazards*, **17**, 57.

Mader, C. L., & Gittings, M. L. 2002. Modeling the 1958 Lituya Bay mega-tsunami, II. *Science of Tsunami Hazards*, **20**(5), 241–250.

Manolopoulos, Y., Nanopoulos, A., Papadopoulos, A., & Theodoridis, Y. 2006. *R-trees: Theory and Applications.* Springer.

Martin, J. C., & Moyce, W. J. 1952. Part IV. An Experimental Study of the Collapse of Liquid Columns on a Rigid Horizontal Plane. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, **244**(882), 312–324.

Mathur, S. R., & Murthy, J. Y. 1997. A pressure-based method for unstructured meshes. *Numerical Heat Transfer, Part B: Fundamentals*, **31**(2), 195 – 215.

McGlaun, J. M., Thompson, S. L., & Elrick, M. G. 1989. CTH: a three-dimensional shock wave physics code. *International Journal of Impact Engineering*, **10**(1), 351–360.

Melosh, H. J. 1989. *Impact cratering : a geologic process.* Oxford University Press.

Miller, D. J. 1960a. The Alaska earthquake of July 10, 1958: Giant wave in Lituya Bay. *Bulletin of the Seismological Society of America*, **50**(2), 253–266.

Miller, D. J. 1960b. Giant waves in Lituya Bay, Alaska: US Geol. *Survey Prof. Paper*, 51–86.

Neri, A., Ongaro, T. E., Macedonio, G., & Gidaspow, D. 2003. Multiparticle simulation of collapsing volcanic columns and pyroclastic flow. *Journal of Geophysical Research*, **108**(B4).

Noh, W. F., & Woodward, P. 1976. SLIC (Simple Line Interface Calculation). *In: Lectures Notes in Physics*, vol. 59. Springer-Verlag.

Overmars, M. 1996. Designing the computational geometry algorithms library CGAL. *Pages 53–58 of: Applied Computational Geometry Towards Geometric Engineering*. Springer.

Pain, C. C., Umpleby, A. P., de Oliveira, C. R. E., & Goddard, A. J. H. 2001. Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations. *Computer Methods in Applied Mechanics and Engineering*, **190**(29-30), 3771–3796.

Park, I. R., Kim, K. S., Kim, J., & Van, S. H. 2009. A volume-of-fluid method for incompressible free surface flows. *International Journal for Numerical Methods in Fluids*.

Patil, D. V., & Lakshmisha, K. N. 2009. Finite volume TVD formulation of lattice Boltzmann simulation on unstructured mesh. *Journal of Computational Physics*, **228**(14), 5262–5279.

Pedersen, G., & Gjevik, B. 1983. Run-up of Solitary Waves. *Journal of Fluid Mechanics Digital Archive*, **135**(-1), 283–299.

Piggott, M. D., Gorman, G. J., Pain, C. C., Allison, P. A., Candy, A. S., Martin, B. T., & Wells, M. R. 2008. A new computational framework for multi-scale ocean modelling based on adapting unstructured meshes. *International Journal for Numerical Methods in Fluids*, **56**(8), 1003–1015.

Piggott, M.D., Pain, C.C., Gorman, G.J., Power, P.W., & Goddard, A.J.H. 2005. h, r, and hr adaptivity with applications in numerical ocean modelling. *Ocean Modelling*, **10**(1-2), 95–113.

Power, P.W., Piggott, M.D., Fang, F., Gorman, G.J., Pain, C.C., Marshall, D.P., Goddard, A.J.H., & Navon, I.M. 2006. Adjoint goal-based error norms for adaptive mesh ocean modelling. *Ocean Modelling*, **15**(1-2), 3–38.

Qiu, J., & Shu, C.-W. 2004. Hermite WENO schemes and their application as limiters for Runge-Kutta discontinuous Galerkin method: one-dimensional case. *Journal of Computational Physics*, **193**(1), 115–135.

Quecedo, M., Pastor, M., & Herreros, M. I. 2004. Numerical modelling of impulse wave generated by fast landslides. *International Journal for Numerical Methods in Engineering*, **59**(12), 1633–1656.

Rider, W. J., & Kothe, D. B. 1995. Stretching and tearing interface tracking methods. *Pages 806–816 of: AIAA Computational Fluid Dynamics Conference, 12th, and Open Forum, San Diego, California.*

Roache, P. J. 2002. Code Verification by the Method of Manufactured Solutions. *Journal of Fluids Engineering*, **124**(1), 4–10.

Rudman, M. 1997. Volume-tracking methods for interfacial flow calculations. *International Journal for Numerical Methods in Fluids*, **24**(7), 671–691.

Saad, Y. 2003. *Iterative methods for sparse linear systems.* Society for Industrial Mathematics.

Saad, Y., & Schultz, M. H. 1986. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, **7**(3), 856–869.

Saito, T., & Furumura, T. 2009. Three-dimensional simulation of tsunami generation and propagation: Application to intraplate events. *Journal of Geophysical Research*, **114**(Feb.).

Salari, K., & Knupp, P. 2000. *Code Verification by the Method of Manufactured Solutions.* Tech. rept. SAND2000-1444. Sandia National Labs., Albuquerque, NM (US); Sandia National Labs., Livermore, CA (US).

Satish, B., Buschelman, K., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. Curfman, Smith, B. F., & Zhang, H. 2001. *PETSc Web page.* http://www.mcs.anl.gov/petsc/petsc-2/.

Satish, B., Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. Curfman, Smith, B. F., & Zhang, H. 2004. *PETSc Users Manual.* Argonne National Laboratory.

Schofield, S. P., Garimella, R. V., Francois, M. M., & Loubère, R. 2008. Material order-independent interface reconstruction using power diagrams. *International Journal for Numerical Methods in Fluids*, **56**(6), 643–659.

Schofield, S. P., Christon, M. A., Dyadechko, V., Garimella, R. V., Lowrie, R. B., & Swartz, B. K. 2009. Multi-material incompressible flow simulation using the moment-of-fluid method. *International Journal for Numerical Methods in Fluids.*

Schwaiger, H. F., & Higman, B. 2007. Lagrangian hydrocode simulations of the 1958 Lituya Bay tsunamigenic rockslide. *Geochemistry, Geophysics, Geosystems*, **8**.

Shen, J. 1996. On Error Estimates of the Projection Methods for the Navier-Stokes Equations: Second-Order Schemes. *Mathematics of Computation*, **65**(215), 1039–1065.

Sherwin, S. J., Kirby, R. M., Peir, J., Taylor, R. L., & Zienkiewicz, O. C. 2006. On 2D elliptic discontinuous Galerkin methods. *International Journal for Numerical Methods in Engineering*, **65**(5), 752–784.

Silvester, D., Elman, H., Kay, D., & Wathen, A. 2001. Efficient preconditioning of the linearized Navier-Stokes equations for incompressible flow. *Journal of Computational and Applied Mathematics*, **128**(1-2), 261–279.

Slingerland, R. L., & Voight, B. 1979. Occurrences, properties, and predictive models of landslide-generated water waves. *Rockslides and Avalanches: Engineering sites*, 317.

Stein, W. A., *et al.* 2009. *Sage Mathematics Software (Version 3.3)*. http://www.sagemath.org.

Sulsky, D., Zhou, S.-J., & Schreyer, H. L. 1995. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications*, **87**(1-2), 236–252.

Sussman, M., Smereka, P., & Osher, S. 1994. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics*, **114**(1), 146–159.

Sweby, P. K. 1984. High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws. *SIAM Journal on Numerical Analysis*, **21**(5), 995–1011.

Synolakis, C. E. 1987. The Runup of Solitary Waves. *Journal of Fluid Mechanics Digital Archive*, **185**(-1), 523–545.

Taylor, C., & Hood, P. 1973. A numerical solution of the Navier-Stokes equations using the finite element technique. *Computers & Fluids*, **1**(1), 73–100.

Temam, Roger. 2001. *Navier-Stokes equations*. AMS Bookstore.

Tewarson, R. P. 1973. *Sparse matrices*. Academic Press New York.

Thuburn, John. 1996. Multidimensional Flux-Limited Advection Schemes. *Journal of Computational Physics*, **123**(1), 74–83.

Tinti, S., Maramai, A., Armigliato, A., Graziani, L., Manucci, A., Pagnoni, G., & Zaniboni, F. 2006. Observations of physical effects from tsunamis of December 30, 2002 at Stromboli volcano, southern Italy. *Bulletin of Volcanology*, **68**(5), 450–461.

Tocher, D., & Miller, D. J. 1959. Field Observations on Effects of Alaska Earthquake of 10 July 1958. *Science*, **129**(3346), 394–395.

Trefethen, L. N., & Bau, D. 1997. *Numerical linear algebra*. Philadelphia: Society for Industrial and Applied Mathematics.

Ubbink, O., & Issa, R. I. 1999. A Method for Capturing Sharp Fluid Interfaces on Arbitrary Meshes. *Journal of Computational Physics*, **153**(1), 26–50.

Umgiesser, G., Canu, D. M., Cucco, A., & Solidoro, C. 2004. A finite element model for the Venice Lagoon. Development, set up, calibration and validation. *Journal of Marine Systems*, **51**(1-4), 123–145.

Vasilevski, Y. V., & Lipnikov, K. N. 1999. An adaptive algorithm for quasioptimal mesh generation. *Computational Mathematics and Mathematical Physics*, **39**(9), 1468–1486.

Wan, T., Aliabadi, S., & Bigler, C. 2009. A hybrid scheme based on finite element/volume methods for two immiscible fluid flows. *International Journal for Numerical Methods in Fluids*.

Ward, S. N. 2001. Landslide tsunami. *Journal of Geophysical Research*, **106**(B6).

Waterson, N. P. 2008. A symmetric formulation for flux-limited convection schemes. *International Journal for Numerical Methods in Fluids*, **56**(8), 1575–1581.

Weiss, R., Fritz, H. M., & Wünnemann, K. 2009. Hybrid modeling of the mega-tsunami runup in Lituya Bay after half a century. *Geophysical Research Letters*, **36**(May).

Williams, R. T., & Zienkiewicz, O. C. 1981. Improved finite element forms for the shallow-water wave equations. *International Journal for Numerical Methods in Fluids*, **1**(1), 81–97.

Wünnemann, K., Collins, G.S., & Melosh, H.J. 2006. A strain-based porosity model for use in hydrocode simulations of impacts and implications for transient crater growth in porous targets. *Icarus*, **180**(2), 514–527.

Yih, C. S. 1965. *Dynamics of Nonhomogeneous Fluids*. Macmillan Series in Advanced Mathematics and Theoretical Physics. Macmillan.

Youngs, D. L. 1982. Time-dependent multi-material flow with large fluid distortion. *Numerical methods for fluid dynamics*, **1**(1), 41–51.

Zahedi, S., Gustavsson, K., & Kreiss, G. 2009. A conservative level set method for contact line dynamics. *Journal of Computational Physics*, **228**(17), 6361–6375.

Zalesak, S. T. 1979. Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of Computational Physics*, **31**(3), 335–362.

Zhang, Y., Zou, Q., & Greaves, D. 2009. Numerical simulation of free-surface flow using the level-set method with global mass correction. *International Journal for Numerical Methods in Fluids*.

Zhao, Y., Tan, H. H., & Zhang, B. 2002. A High-Resolution Characteristics-Based Implicit Dual Time-Stepping VOF Method for Free Surface Flow Simulation on Unstructured Grids. *Journal of Computational Physics*, **183**(1), 233–273.

Zhou, Z. Q., Kat, J. O. De, & Buchner, B. 1999. A nonlinear 3-D approach to simulate green water dynamics on deck. *In: Report No. 82000-NSH*, vol. 7.

Zienkiewicz, O. C. 1977. *The finite element method*. 3rd edn. London: McGraw-Hill.

Zienkiewicz, O. C., & Taylor, R. L. 2000a. *The finite element method. Vol. 1, The basis*. 5th edn. Oxford: Butterworth-Heinemann.

Zienkiewicz, O. C., & Taylor, R. L. 2000b. *The finite element method. Vol. 3, Fluid dynamics*. 5th edn. Oxford: Butterworth-Heinemann.

Appendix A

Model Code

The Fluidity code-base is available to download from the subversion repository at:
`http://amcg.ese.ic.ac.uk/svn/fluidity/trunk`
Additionally, long automated regression tests can be found at:
`http://amcg.ese.ic.ac.uk/svn/fluidity-longtests`
Either can be downloaded to a computer with a subversion client installed. On a Linux or Unix based system the command:
`svn co http://amcg.ese.ic.ac.uk/svn/fluidity/trunk fluidity`
will place the code in a folder called `fluidity` within the current directory.

This appendix contains lists that give an overview of which files were:

- created by the author as part of this thesis;
- created by other contributors to the code-base but required modification for use in this thesis; and
- created by other contributors and used as part of this thesis.

A list of automated regression tests added as developments were made is also provided.

Summary of important files created as part of this thesis:

`assemble/Diagnostic_Fields_Matrices.F90`	subroutines for taking the discrete divergence and gradient
`assemble/Divergence_Matrix_CG.F90`	continuous Galerkin divergence matrix assembly
`assemble/Divergence_Matrix_CV.F90`	control volume divergence matrix assembly
`assemble/Field_Equations_CV.F90`	assembly and solution of control volume field equations
`assemble/Gradient_Matrix_CG.F90`	continuous Galerkin gradient matrix assembly
`assemble/Momentum_CG.F90`	continuous Galerkin momentum assembly
`assemble/Momentum_Diagnostic_Fields.F90`	calculation of fields, like viscosity and density, for momentum
`assemble/Momentum_Equation.F90`	implementation of the projection method for the solution of the momentum and mass equations
`assemble/Multimaterials.F90`	calculation of background volume fraction and average material properties
`assemble/State_Matrices.F90`	access to important stored matrices, like divergence
`femtools/CV_Faces.F90`	control volume face information within a canonical element
`femtools/CV_Face_Values.F90`	calculation of the face values for control volumes using various schemes
`femtools/CV_Options.F90`	control volume options parser
`femtools/CV_Shape_Functions.F90`	translation of standard finite element quadrature and shape functions onto the control volume mesh
`femtools/CVTools.F90`	tools for the assembly of the control volumes, like orientation of the unit normal vector
`femtools/CV_Upwind_Values.F90`	calculation and storage of the upwind element and values

femtools/FEFields.F90	calculation of the lumped mass matrix/control volume mass matrix
femtools/Full_Projection.F90	projection using an outer and an inner solve (substantially modified by other contributors since its original creation)
femtools/Sparsity_Patterns_Meshes.F90	access to stored matrix sparsity patterns
femtools/State_Fields.F90	access to important stored fields, like the lumped mass matrix

Summary of important files modified as part of this thesis:

`assemble/Assemble_CMC.F90`	assembly of the projection matrix
`assemble/Diagnostic_fields_wrapper.F90`	wrapper to access various diagnostic fields, such as control volume Courant number, divergence and gradient
`assemble/Interpolation_manager.F90`	management of interpolations following mesh optimisation
`assemble/Momentum_DG.F90`	assembly of discontinuous Galerkin discretisation of the momentum equation, modified to allow P0
`assemble/Solenoidal_interpolation.F90`	projection of a vector field to its divergence free subspace
`femtools/Adaptive_Timestepping.F90`	adjustment of the timestep size based on a Courant number criterion
`femtools/Boundary_Conditions.F90`	management of boundary conditions
`femtools/Bound_field.F90`	bounding of the field in the bounded projection algorithm
`femtools/Conservative_interpolation.F90`	interpolation between meshes using a conservative projection
`femtools/Diagnostic_Fields.F90`	calculation of diagnostic fields, such as control volume Courant number
`femtools/Diagnostic_variables.F90`	output of key diagnostics, such as the L_2 norm on the control volume mesh
`femtools/Element_Numbering.F90`	library of local numbering of elements
`femtools/Elements.F90`	library of finite elements
`femtools/FETools.F90`	calculation of various finite element bilinear forms
`femtools/Field_Options.F90`	handling of options for fields
`femtools/Fields.F90`	wrapper for modules dealing with the allocation and manipulation of fields
`femtools/Mixing_Statistics.F90`	mixing statistics for a field on the mesh

femtools/Sparse_Matrices_Fields.F90	multiplication of matrices with fields
femtools/State.F90	field and mesh data and matrix storage
femtools/Transform_elements.F90	Jacobian transformation calculation
femtools/VTK_interfaces.F90	output from simulation
main/Fluids.F90	main time loop
preprocessor/Initialise_Fields.F90	initialisation of fields
preprocessor/Populate_State.F90	initialisation of simulation

Summary of important files used:

`assemble/Adapt_Integration.F90`	interface to adaptive remeshing
`assemble/Adapt_State.F90`	interface to adaptive remeshing and interpolation
`assemble/Mba2d_Integration.F90`	interface to two-dimensional adaptive remeshing
`femtools/Checkpoint.F90`	checkpointing of simulations
`femtools/Field_derivatives.F90`	evaluation of the derivatives of fields
`femtools/Interpolation.F90`	interface to the consistent interpolation of fields
`femtools/Intersection_finder.F90`	intersection finder for the supermesh
`femtools/Petsc_Tools.F90`	tools for the PETSc solvers
`femtools/Pickers.F90`	wrapper to data point picking used in consistent interpolation
`femtools/Polynomials.F90`	polynomial types used to represent shape functions
`femtools/qsortd.F90`	sorting algorithm
`femtools/Quadrature.F90`	quadrature library
`femtools/Read_Triangle.F90`	input of meshes
`femtools/Shape_Functions.F90`	creation of shape functions based on quadrature and element information
`femtools/Solvers.F90`	access to PETSc solvers
`femtools/Sparse_Tools.F90`	tools for manipulating sparse matrices
`femtools/Sparse_Tools_Petsc.F90`	tools for manipulating sparse matrices in PETSc format
`femtools/Sparsity_Patterns.F90`	definition of matrix sparsity patterns
`femtools/Supermesh.F90`	supermesh assembly
`femtools/Tetrahedron_intersection.F90`	supermesh intersections for tetrahedral meshes
`femtools/Unify_meshes.F90`	create a supermesh
`femtools/Vector_Tools.F90`	tools for handling and diagnosing vectors

Automated regression tests of major developments:

`tests/mms_burgers_cg_steady`	MMS convergence test for Burgers equation on P1 and P2 unstructured mesh
`tests/mms_burgers_cg_structured_steady`	MMS convergence test for Burgers equation on P1 and P2 structured mesh
`tests/mms_burgers_dg_steady`	MMS convergence test for Burgers equation on a $P1_{DG}$ unstructured mesh
`tests/mms_burgers_p0_steady`	MMS convergence test for Burgers equation on a P0 unstructured mesh
`tests/mms_ns_cg_steady_full`	MMS convergence test for Navier-Stokes equations on a P2P1 unstructured mesh
`tests/mms_ns_cg_steady`	MMS convergence test for Navier-Stokes equations on a $P2_LP1$ unstructured mesh
`tests/mms_ns_dg_steady`	MMS convergence test for Navier-Stokes equations on a $P1_{DG}P2$ unstructured mesh
`tests/mms_ns_dg_steady_parallel`	MMS convergence test for Navier-Stokes equations on a $P1_{DG}P2$ unstructured mesh in parallel
`tests/mms_ns_p0p1cv_steady_periodic`	MMS convergence test for Navier-Stokes equations on a $P0P1_{CV}$ periodic mesh
`tests/mms_ns_p0p1_steady_periodic`	MMS convergence test for Navier-Stokes equations on a P0P1 periodic mesh
`tests/mms_ns_p1dgp2_steady_periodic`	MMS convergence test for Navier-Stokes equations on a $P1_{DG}P2$ periodic mesh
`tests/mms_ns_p1p1stabilised_steady`	MMS convergence test for Navier-Stokes equations on a $P1_LP1$ unstructured mesh with stabilisation

tests/mms_tracer_cv_advdiff_eg_steady	MMS convergence test for the advection diffusion equation on an unstructured control volume mesh, using the parent basis functions to construct the diffusion operator
tests/mms_tracer_cv_advdiff_steady	MMS convergence test for the advection diffusion equation on an unstructured control volume mesh, using Bassi & Rebay (1997) to construct the diffusion operator
tests/mms_tracer_cv_diff_eg_steady	MMS convergence test for the diffusion equation on an unstructured control volume mesh, using the parent basis functions to construct the diffusion operator
tests/mms_tracer_cv_diff_steady	MMS convergence test for the diffusion equation on an unstructured control volume mesh, using Bassi & Rebay (1997) to construct the diffusion operator
tests/mms_tracer_cv_diff_steady_structured	MMS convergence test for the advection diffusion equation on an unstructured control volume mesh, using Bassi & Rebay (1997) and a P0 mesh for the diffusivity to construct the diffusion operator
tests/mms_tracer_p0_adv_steady_conservative	MMS convergence test for the conservative advection equation on a P0 unstructured mesh
tests/mms_tracer_p0_adv_steady	MMS convergence test for the advection equation on a P0 unstructured mesh
tests/explicit-hyperc-shear-adapt	shearing test case using HyperC on an adaptive mesh
tests/explicit-hyperc-shear	shearing test case using HyperC on a fixed mesh

`tests/lituya_bay_parallel_debug`	fixed mesh Lituya Bay simulation in parallel
`tests/lituya_bay_pseudo2dadapt`	pseudo-2d adaptive mesh Lituya Bay simulation
`tests/explicit-hyperc-hexes`	advection of a step function using HyperC on a piecewise trilinear hexahedral parent mesh
`tests/explicit-hyperc-quads`	advection of a step function using HyperC on a piecewise bilinear quadrilateral parent mesh
`tests/explicit-hyperc-tets`	advection of a step function using HyperC on a piecewise linear tetrahedral parent mesh
`tests/explicit-hyperc-tris`	advection of a step function using HyperC on a piecewise linear triangular parent mesh
`tests/explicit-hyperc-tris-p2`	advection of a step function using HyperC on a piecewise quadratic triangular parent mesh
`tests/explicit-hyperc-tris-periodic-local`	advection of a step function using HyperC on a periodic piecewise linear triangular parent mesh using a local upwind value
`tests/explicit-hyperc-tris-periodic-projgd`	advection of a step function using HyperC on a periodic piecewise linear triangular parent mesh using an upwind value projected based on the nodal gradient
`tests/explicit-hyperc-tris-periodic-projpt`	advection of a step function using HyperC on a periodic piecewise linear triangular parent mesh using an upwind value projected based on the element gradient

`tests/3material-droplet`	three material simulation testing conservation and boundedness on a fixed mesh with HyperC and the coupled limiter
`tests/3material-droplet-adapt`	three material simulation testing conservation and boundedness on an adaptive mesh with HyperC and the coupled limiter
`tests/3material-oneblobs`	one dimensional test of HyperC and the coupled limiter
`tests/3material-oneblobs-reversed`	one dimensional test of HyperC and the coupled limiter
`tests/3material-twoblobs`	one dimensional test of HyperC and the coupled limiter
`tests/3material-twoblobs-reversed`	one dimensional test of HyperC and the coupled limiter
`tests/5material-droplet-adapt`	five material simulation testing conservation and boundedness on an adaptive mesh with HyperC and the coupled limiter
`tests/5material-droplet-adapt-lumped`	five material simulation testing conservation and boundedness on an adaptive mesh with HyperC, the coupled limiter and a lumped projection between meshes
`tests/5material-droplet-parallel`	five material simulation testing conservation and boundedness on a fixed mesh with HyperC and the coupled limiter in parallel
`tests/5material-droplet-serial`	five material simulation testing conservation and boundedness on a fixed mesh with HyperC and the coupled limiter
`tests/water_collapse_2d_algencan`	water column collapse on an adaptive mesh using a nonlinear optimisation library to bound the volume fraction following mesh optimisation

`tests/water_collapse_2d`	water column collapse on an adaptive mesh
`tests/water_collapse_pseudo2dadapt`	three-dimensional water column collapse on an adaptive mesh
`tests/mmat-balancepressure`	separate the balance pressure from the main pressure during a $\mathrm{P0P1_{CV}}$ multi-material simulation
`tests/mmat-gravity-col`	gravity-pressure gradient force balance test using $\mathrm{P1_LP1_{CV}}$
`tests/mmat-gravity-col-p0p1cv-weak`	gravity-pressure gradient force balance test using $\mathrm{P0P1_{CV}}$
`tests/mmat-gravity-col-p0p1-weak`	gravity-pressure gradient force balance test using P0P1
`tests/mmat-gravity-col-p2lumpedp1`	gravity-pressure gradient force balance test using $\mathrm{P2_LP1}$
`tests/mmat-gravity-col-p2p1`	gravity-pressure gradient force balance test using $\mathrm{P2_LP1}$
`tests/mmat-interpolation-lumped`	interpolation of three materials using the lumped projection
`tests/mmat-interpolation`	interpolation of three materials using the bounded projection
`tests/mmat-interpolation-pseudo2d-parallel`	interpolation of three materials using the bounded projection in parallel
`tests/mmat-interpolation-represcribe`	interpolation of three materials using the bounded projection and reinitialising prescribed fields after mesh optimisation
`tests/viscosity_2d_p0_parallel`	P0 viscosity in parallel
`tests/viscosity_2d_p0_serial`	P0 viscosity
`tests/diffusion_2d_p0_parallel`	P0 diffusion in parallel
`tests/diffusion_2d_p0_serial`	P0 diffusion

Appendix B

Advection Tests

This appendix provides some additional results from the advection tests of Chapters 3 and 5, varying key parameters like the face value scheme, the Courant number or the adaptive mesh settings.

B.1 Two Material Solid Body Rotation

First-Order Upwind		Sweby Limited Trapezoidal		Sweby Limited Interpolant	
1^{st} Rotation	2^{nd} Rotation	1^{st} Rotation	2^{nd} Rotation	1^{st} Rotation	2^{nd} Rotation
9.35×10^{-1}	1.19×10^{0}	2.97×10^{-1}	3.44×10^{-1}	3.04×10^{-1}	3.55×10^{-1}

Table B.1: Errors for the Zalesak slot problem discussed in Section 3.3.2 after one and two full rotations using first-order upwind, Sweby limited trapezoidal and Sweby limited interpolated face values on the mesh shown in Figure 3.26(b).

B.2 Two Material Shearing Flow

Table B.2: *See next page.* Errors for the shearing flow test case discussed in Section 3.3.2 after N timesteps of forward and backward advection using first-order upwind, Sweby limited trapezoidal and Sweby limited interpolated face values. The average edge length, h, of the parent mesh and the maximum Courant number ignoring the smaller boundary cells, $\max(c_i)$, are varied relative to the case shown in Figures 3.30 and 3.33. The Courant number definition from Equation 3.59 is used.

N	h	$\max(c_i)$	First-Order Upwind	Sweby Limited Trapezoidal	Sweby Limited Interpolant
250	$\frac{\pi}{100}$	$\frac{1}{4}$	4.48×10^{-1}	1.38×10^{-1}	1.41×10^{-1}
250	$\frac{\pi}{50}$	$\frac{1}{4}$	6.15×10^{-1}	2.26×10^{-1}	2.31×10^{-1}
250	$\frac{\pi}{200}$	$\frac{1}{4}$	3.19×10^{-1}	8.09×10^{-2}	8.36×10^{-2}
250	$\frac{\pi}{100}$	$\frac{1}{10}$	4.56×10^{-1}	1.50×10^{-1}	1.53×10^{-1}
250	$\frac{\pi}{50}$	$\frac{1}{10}$	6.27×10^{-1}	2.44×10^{-1}	2.49×10^{-1}
250	$\frac{\pi}{200}$	$\frac{1}{10}$	3.24×10^{-1}	9.00×10^{-2}	9.27×10^{-2}
500	$\frac{\pi}{100}$	$\frac{1}{4}$	6.50×10^{-1}	1.73×10^{-1}	1.80×10^{-1}
500	$\frac{\pi}{50}$	$\frac{1}{4}$	8.76×10^{-1}	2.87×10^{-1}	2.96×10^{-1}
500	$\frac{\pi}{200}$	$\frac{1}{4}$	4.67×10^{-1}	1.02×10^{-1}	1.06×10^{-1}
500	$\frac{\pi}{100}$	$\frac{1}{10}$	6.60×10^{-1}	1.91×10^{-1}	1.98×10^{-1}
500	$\frac{\pi}{50}$	$\frac{1}{10}$	8.90×10^{-1}	3.12×10^{-1}	3.22×10^{-1}
500	$\frac{\pi}{200}$	$\frac{1}{10}$	4.74×10^{-1}	1.15×10^{-1}	1.19×10^{-1}
1,000	$\frac{\pi}{100}$	$\frac{1}{4}$	9.36×10^{-1}	2.30×10^{-1}	2.42×10^{-1}
1,000	$\frac{\pi}{50}$	$\frac{1}{4}$	1.20×10^{0}	3.91×10^{-1}	4.08×10^{-1}
1,000	$\frac{\pi}{200}$	$\frac{1}{4}$	7.03×10^{-1}	1.35×10^{-1}	1.43×10^{-1}
1,000	$\frac{\pi}{100}$	$\frac{1}{10}$	9.48×10^{-1}	2.57×10^{-1}	2.69×10^{-1}
1,000	$\frac{\pi}{50}$	$\frac{1}{10}$	1.21×10^{0}	4.28×10^{-1}	4.46×10^{-1}
1,000	$\frac{\pi}{200}$	$\frac{1}{10}$	7.11×10^{-1}	1.55×10^{-1}	1.62×10^{-1}
2,000	$\frac{\pi}{100}$	$\frac{1}{4}$	1.33×10^{0}	4.12×10^{-1}	4.37×10^{-1}
2,000	$\frac{\pi}{50}$	$\frac{1}{4}$	1.54×10^{0}	6.69×10^{-1}	7.01×10^{-1}
2,000	$\frac{\pi}{200}$	$\frac{1}{4}$	1.08×10^{0}	2.43×10^{-1}	2.62×10^{-1}
2,000	$\frac{\pi}{100}$	$\frac{1}{10}$	1.34×10^{0}	4.45×10^{-1}	4.70×10^{-1}
2,000	$\frac{\pi}{50}$	$\frac{1}{10}$	1.54×10^{0}	7.10×10^{-1}	7.41×10^{-1}
2,000	$\frac{\pi}{200}$	$\frac{1}{10}$	1.09×10^{0}	2.68×10^{-1}	2.87×10^{-1}
3,000	$\frac{\pi}{100}$	$\frac{1}{4}$	1.54×10^{0}	6.13×10^{-1}	6.49×10^{-1}
3,000	$\frac{\pi}{50}$	$\frac{1}{4}$	1.68×10^{0}	9.32×10^{-1}	9.71×10^{-1}
3,000	$\frac{\pi}{200}$	$\frac{1}{4}$	1.35×10^{0}	3.81×10^{-1}	4.10×10^{-1}
3,000	$\frac{\pi}{100}$	$\frac{1}{10}$	1.55×10^{0}	6.43×10^{-1}	6.79×10^{-1}
3,000	$\frac{\pi}{50}$	$\frac{1}{10}$	1.68×10^{0}	9.66×10^{-1}	1.00×10^{0}
3,000	$\frac{\pi}{200}$	$\frac{1}{10}$	1.35×10^{0}	4.04×10^{-1}	4.35×10^{-1}
4,000	$\frac{\pi}{100}$	$\frac{1}{4}$	1.65×10^{0}	7.95×10^{-1}	8.38×10^{-1}
4,000	$\frac{\pi}{50}$	$\frac{1}{4}$	1.72×10^{0}	1.15×10^{0}	1.19×10^{0}
4,000	$\frac{\pi}{200}$	$\frac{1}{4}$	1.51×10^{0}	5.19×10^{-1}	5.56×10^{-1}
4,000	$\frac{\pi}{100}$	$\frac{1}{10}$	1.65×10^{0}	8.23×10^{-1}	8.65×10^{-1}
4,000	$\frac{\pi}{50}$	$\frac{1}{10}$	1.72×10^{0}	1.17×10^{0}	1.21×10^{0}
4,000	$\frac{\pi}{200}$	$\frac{1}{10}$	1.51×10^{0}	5.63×10^{-1}	6.01×10^{-1}

Table B.2: *cont.*

N	h	$\max(c_i)$	HyperC	UltraC	Ultrac-S	UltraC-P
250	$\frac{\pi}{100}$	$\frac{1}{4}$	2.46×10^{-2}	3.95×10^{-2}	3.65×10^{-2}	4.04×10^{-2}
250	$\frac{\pi}{50}$	$\frac{1}{4}$	4.74×10^{-2}	6.85×10^{-2}	6.38×10^{-2}	7.43×10^{-2}
250	$\frac{\pi}{200}$	$\frac{1}{4}$	1.39×10^{-2}	2.68×10^{-2}	2.32×10^{-2}	2.07×10^{-2}
250	$\frac{\pi}{100}$	$\frac{1}{10}$	2.19×10^{-2}	4.23×10^{-2}	3.78×10^{-2}	4.08×10^{-2}
250	$\frac{\pi}{50}$	$\frac{1}{10}$	4.19×10^{-2}	7.34×10^{-2}	6.89×10^{-2}	7.72×10^{-2}
250	$\frac{\pi}{200}$	$\frac{1}{10}$	1.38×10^{-2}	2.83×10^{-2}	2.55×10^{-2}	2.65×10^{-2}
500	$\frac{\pi}{100}$	$\frac{1}{4}$	2.85×10^{-2}	4.62×10^{-2}	4.30×10^{-2}	4.36×10^{-2}
500	$\frac{\pi}{50}$	$\frac{1}{4}$	5.19×10^{-2}	8.23×10^{-2}	6.78×10^{-2}	8.55×10^{-2}
500	$\frac{\pi}{200}$	$\frac{1}{4}$	1.72×10^{-2}	2.96×10^{-2}	2.56×10^{-2}	2.65×10^{-2}
500	$\frac{\pi}{100}$	$\frac{1}{10}$	2.81×10^{-2}	5.33×10^{-2}	4.91×10^{-2}	4.73×10^{-2}
500	$\frac{\pi}{50}$	$\frac{1}{10}$	5.24×10^{-2}	8.58×10^{-2}	7.49×10^{-2}	9.11×10^{-2}
500	$\frac{\pi}{200}$	$\frac{1}{10}$	1.73×10^{-2}	3.47×10^{-2}	2.65×10^{-2}	2.97×10^{-2}
1,000	$\frac{\pi}{100}$	$\frac{1}{4}$	3.45×10^{-2}	5.81×10^{-2}	5.68×10^{-2}	5.42×10^{-2}
1,000	$\frac{\pi}{50}$	$\frac{1}{4}$	6.24×10^{-2}	1.06×10^{-1}	7.89×10^{-2}	9.14×10^{-2}
1,000	$\frac{\pi}{200}$	$\frac{1}{4}$	2.22×10^{-2}	3.61×10^{-2}	3.15×10^{-2}	3.22×10^{-2}
1,000	$\frac{\pi}{100}$	$\frac{1}{10}$	3.56×10^{-2}	7.67×10^{-2}	6.65×10^{-2}	6.15×10^{-2}
1,000	$\frac{\pi}{50}$	$\frac{1}{10}$	6.08×10^{-2}	1.31×10^{-1}	9.36×10^{-2}	1.05×10^{-1}
1,000	$\frac{\pi}{200}$	$\frac{1}{10}$	2.22×10^{-2}	4.90×10^{-2}	4.12×10^{-2}	4.44×10^{-2}
2,000	$\frac{\pi}{100}$	$\frac{1}{4}$	5.42×10^{-2}	7.40×10^{-2}	9.85×10^{-2}	8.95×10^{-2}
2,000	$\frac{\pi}{50}$	$\frac{1}{4}$	8.77×10^{-2}	1.52×10^{-1}	1.12×10^{-1}	1.35×10^{-1}
2,000	$\frac{\pi}{200}$	$\frac{1}{4}$	3.79×10^{-2}	5.09×10^{-2}	4.86×10^{-2}	4.48×10^{-2}
2,000	$\frac{\pi}{100}$	$\frac{1}{10}$	6.34×10^{-2}	1.21×10^{-1}	9.84×10^{-2}	1.12×10^{-1}
2,000	$\frac{\pi}{50}$	$\frac{1}{10}$	9.69×10^{-2}	1.98×10^{-1}	1.30×10^{-1}	1.77×10^{-1}
2,000	$\frac{\pi}{200}$	$\frac{1}{10}$	3.27×10^{-2}	9.61×10^{-2}	6.90×10^{-2}	6.52×10^{-2}
3,000	$\frac{\pi}{100}$	$\frac{1}{4}$	7.36×10^{-2}	1.06×10^{-1}	1.22×10^{-1}	1.04×10^{-1}
3,000	$\frac{\pi}{50}$	$\frac{1}{4}$	1.65×10^{-1}	1.77×10^{-1}	2.07×10^{-1}	2.13×10^{-1}
3,000	$\frac{\pi}{200}$	$\frac{1}{4}$	5.32×10^{-2}	6.19×10^{-2}	6.38×10^{-2}	5.00×10^{-2}
3,000	$\frac{\pi}{100}$	$\frac{1}{10}$	8.07×10^{-2}	1.62×10^{-1}	1.38×10^{-1}	1.40×10^{-1}
3,000	$\frac{\pi}{50}$	$\frac{1}{10}$	1.41×10^{-1}	2.48×10^{-1}	1.74×10^{-1}	1.93×10^{-1}
3,000	$\frac{\pi}{200}$	$\frac{1}{10}$	4.09×10^{-2}	1.29×10^{-1}	1.03×10^{-1}	1.03×10^{-1}
4,000	$\frac{\pi}{100}$	$\frac{1}{4}$	1.26×10^{-1}	1.72×10^{-1}	1.72×10^{-1}	1.39×10^{-1}
4,000	$\frac{\pi}{50}$	$\frac{1}{4}$	2.12×10^{-1}	4.01×10^{-1}	3.18×10^{-1}	3.01×10^{-1}
4,000	$\frac{\pi}{200}$	$\frac{1}{4}$	6.36×10^{-2}	8.06×10^{-2}	8.37×10^{-2}	7.63×10^{-2}
4,000	$\frac{\pi}{100}$	$\frac{1}{10}$	1.11×10^{-1}	2.52×10^{-1}	1.87×10^{-1}	1.70×10^{-1}
4,000	$\frac{\pi}{50}$	$\frac{1}{10}$	1.79×10^{-1}	3.42×10^{-1}	2.48×10^{-1}	2.95×10^{-1}
4,000	$\frac{\pi}{200}$	$\frac{1}{10}$	5.35×10^{-2}	1.59×10^{-1}	2.06×10^{-1}	2.00×10^{-1}

Table B.3: *See next page.*

Table B.3: *cont.* Errors for the shearing flow test case discussed in Section 3.3.2 after N timesteps of forward and backward advection using HyperC, UltraC, UltraC-P and UltraC-S face values. The average edge length, h, of the parent mesh and the maximum Courant number ignoring the smaller boundary cells, $\max(c_i)$, are varied relative to the case shown in Figures 3.30 and 3.33. The Courant number definition from Equation 3.59 is used.

	$\max(c_i) = 0.125$		$\max(c_i) = 0.25$		$\max(c_i) = 0.5$	
N	Error	Max. Nodes	Error	Max. Nodes	Error	Max. Nodes
250	2.18×10^{-2}	1,194	2.00×10^{-2}	1,360	1.67×10^{-2}	1,623
500	2.46×10^{-2}	1,194	1.99×10^{-2}	1,360	5.55×10^{-2}	1,623
1,000	3.47×10^{-2}	1,237	3.98×10^{-2}	1,360	4.79×10^{-2}	1,623
2,000	5.19×10^{-2}	1,833	3.99×10^{-2}	1,834	4.71×10^{-2}	2,124
3,000	1.17×10^{-1}	2,310	6.09×10^{-2}	2,371	6.68×10^{-2}	2,544
4,000	1.10×10^{-1}	2,808	1.16×10^{-1}	2,956	9.46×10^{-2}	3,278

Table B.4: Errors for the shearing flow test case on an adaptive mesh with edge length gradation and metric advection using a scaling time of $\gamma = 3$, as discussed in Section 3.5. A circle is sheared for an equivalent of N timesteps of forward advection on the fixed mesh from Section 3.3.2 before being advected back to its initial condition for the same amount of time. HyperC is used at a variety of maximum Courant numbers, $\max(c_i)$. The Courant number definition from Equation 3.59 is used.

	$\gamma = 1$		$\gamma = 2$		$\gamma = 3$	
N	Error	Max. Nodes	Error	Max. Nodes	Error	Max. Nodes
250	4.65×10^{-2}	621	4.71×10^{-2}	683	4.99×10^{-2}	822
500	1.17×10^{-1}	621	8.57×10^{-2}	683	4.64×10^{-2}	822
1,000	1.49×10^{-1}	635	1.92×10^{-1}	730	7.15×10^{-2}	822
2,000	2.35×10^{-1}	1,001	2.29×10^{-1}	1,196	1.55×10^{-1}	1,352
3,000	2.33×10^{-1}	1,363	2.16×10^{-1}	1,515	2.90×10^{-1}	1,569
4,000	2.56×10^{-1}	1,697	3.43×10^{-1}	1,778	2.16×10^{-1}	2,035

Table B.5: Errors for the shearing flow test case on an adaptive mesh with metric advection but no edge length gradation, as discussed in Section 3.5. A circle is sheared for an equivalent of N timesteps of forward advection on the fixed mesh from Section 3.3.2 before being advected back to its initial condition for the same amount of time. HyperC is used at a Courant number of 1/4 and a variety of metric advection scaling times, γ.

B.3 Five Material Shearing Flow

N	Material 1 Error	Material 2 Error	Material 3 Error	Material 4 Error
250	5.92×10^{-2}	5.51×10^{-2}	5.15×10^{-2}	6.72×10^{-2}
500	6.80×10^{-2}	6.64×10^{-2}	5.42×10^{-2}	7.10×10^{-2}
1,000	7.48×10^{-2}	9.44×10^{-2}	7.85×10^{-2}	9.30×10^{-2}
2,000	1.13×10^{-1}	1.36×10^{-1}	1.13×10^{-1}	2.05×10^{-1}
3,000	1.12×10^{-1}	3.75×10^{-1}	1.41×10^{-1}	4.11×10^{-1}
4,000	1.16×10^{-1}	4.61×10^{-1}	2.19×10^{-1}	5.29×10^{-1}

Table B.6: Errors for the five material shearing flow test case discussed in Section 5.1.1 after N timesteps of forward and backward advection at a maximum Courant number of $\max(c_i) = 0.125$ (half that of the results presented in Table 5.3). Results for each prognostic volume fraction using HyperC and the coupled limiter are shown.

	Sum Multiple Materials
N	Error
250	2.51×10^{-2}
500	3.02×10^{-2}
1,000	3.87×10^{-2}
2,000	6.78×10^{-2}
3,000	1.20×10^{-1}
4,000	1.65×10^{-1}

Table B.7: Errors for the sum of the four prognostic volume fractions from the multi-material shearing flow test case discussed in Section 5.1.1 after N timesteps of forward and backward advection at a maximum Courant number of $\max(c_i) = 0.125$ (half that of the results presented in Table 5.4).

N	Material 1 Error	Material 2 Error	Material 3 Error	Material 4 Error
250	5.15×10^{-2}	6.66×10^{-2}	6.87×10^{-2}	1.00×10^{-1}
500	6.65×10^{-2}	1.00×10^{-1}	8.27×10^{-2}	1.51×10^{-1}
1,000	9.42×10^{-2}	1.96×10^{-1}	1.28×10^{-1}	1.93×10^{-1}
2,000	1.46×10^{-1}	3.22×10^{-1}	2.11×10^{-1}	3.77×10^{-1}
3,000	2.18×10^{-1}	4.23×10^{-1}	3.52×10^{-1}	5.34×10^{-1}
4,000	2.75×10^{-1}	6.22×10^{-1}	4.77×10^{-1}	9.04×10^{-1}

Table B.8: Errors for the five material shearing flow test case on an adaptive mesh discussed in Section 5.1.1 after the equivalent of N timesteps of forward and backward advection on the fixed mesh at a maximum Courant number of $\max(c_i) = 0.125$ (half that of the results presented in Table 5.5). Results for each prognostic volume fraction using HyperC and the coupled limiter are shown.

	Sum Multiple Materials	
N	Error	Max. Nodes
250	5.33×10^{-2}	$1,945$
500	7.47×10^{-2}	$2,031$
1,000	9.90×10^{-2}	$2,454$
2,000	1.92×10^{-1}	$3,207$
3,000	2.54×10^{-1}	$3,859$
4,000	4.21×10^{-1}	$4,476$

Table B.9: Errors for the sum of the four prognostic volume fractions from the multi-material shearing flow test case discussed in Section 5.1.1 after the equivalent of N timesteps of forward and backward advection on the fixed mesh at a maximum Courant number of $\max(c_i) = 0.125$ (half that of the results presented in Table 5.6).