The Integrated Space-Time Finite Volume Method

By

Philip J. Zwart

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Mechanical Engineering

Waterloo, Ontario, Canada, 1999

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

The integrated space-time finite volume method for predicting time-dependent fluid flow problems is developed. By enforcing discrete conservation over space-time control volumes which fill the space-time domain, this method satisfies global conservation in space-time. Unlike traditional finite volume methods, there is no need to incorporate the Leibnitz Rule or the geometrical conservation law into the discretization. The method is validated using a variety of two-dimensional problems featuring both prescribed and free boundary motion. Advances in other aspects of cell-centered finite volume discretization — most notably in the modelling of diffusion terms and free surface flows — are also described.

Acknowledgements

I am deeply grateful to my supervisor, Dr. George Raithby, whose insights and encouragement have contributed so much to this thesis. I am also thankful for the many fruitful discussions I have had with my co-supervisor, Dr. Gord Stubley, and with Dr. Michael Raw, who first suggested the topic.

To my colleages, I am thankful for friendship and assistance in various matters. In particular, thanks to Dewey Yin for typesetting help, to Mahmud Ashrafizaadeh and Stephen Reuss for computer support, to Marc Leblanc for the chess and tennis, and to Ali Ashrafizaadeh for enjoyable conversations.

This research has been supported financially by the Natural Science and Engineering Research Council of Canada, in the form of a graduate scholarship and a research assistantship, and by AEA Technology Engineering Software Ltd. Their support is gratefully acknowledged.

To my wife Evelyn, and to my family, I wish to express my deep appreciation for love and support throughout my studies.

Soli Deo Gloria.

Contents

Figures

Tables

Nomenclature

Latin Letters

a	coefficient in discrete conservation equation
b	right-hand side of discrete conservation equation
d	pressure coefficient in steady continuity equation
f	pressure coefficient in unsteady continuity equation
$\mathbf{g}$, g_i	gravitational vector
h	free surface elevation
F^a	advective flux
F^d	diffusive flux
F^p	pressure force
F^v	viscous force
J	mass flow
K	adjustable parameter used in limiter calculation
L	distance from interior vertex to moving boundary
$\overline{L}$	average edge length on mesh
l	length of a free surface edge
m	mass
$\mathbf{m}$, m_j	vector used for anisotropic diffusion term
$\hat{\mathbf{n}}$, n_i	unit normal to face
$\tilde{p}$, p	true pressure, modified pressure
$\mathbf{q}$, q_i	diffusive flux vector
r	residual
$\mathbf{r}$, r_i	position vector,
$\mathbf{r_c}$	vector from midpoint of adjacent cell centroids to face centroid
S	area
$\dot{S}$	volumetric generation rate of a conserved scalar
s	displacement of free surface vertex
$\hat{\mathbf{s}}$, s_i	vector joining spatial cell centroids;
	vector from cell centroid to face centroid

T	period
t	time
$\hat{\mathbf{t}}$, t_i	unit tangent vector
$\mathbf{u}$, u_i	spatial velocity vector
w	weighting factor
x_i	space coordinate (tensor notation)
x, y	space coordinates

Greek Letters

α	scaling factor in diffusion discretization
$\beta_{\mathbf{min}}, \beta_{\mathbf{max}}$	parameters used to determine minimum and maximum allowable distances
Γ	diffusion coefficient
γ_{ij}	space-time metric tensor
η	the set of neighbours of a cell
λ	relaxation parameter for diffusion discretization
μ	dynamic viscosity
ξ	the set of free faces which touch a free vertex
Φ	limiter
ϕ	conserved scalar
ρ	density
τ	stress
ψ	the set of free vertices which touch a free face
Ω	volume
ω	relaxation parameter for gradient calculation

Subscripts

bnd	boundary value
f	face centroid value
n	normal component
P,Q	cells adjacent to face
ref	reference
t	time coordinate; tangential component
up	upstream cell value
fs	free surface
nb	neighbour

Superscripts

$\overline{(\,)}$	average of adjacent cell values
$'$	space-time quantity
$*$	dimensionless
o	old or lagged value

Chapter 1

Introduction

1.1 Motivation

In the past few decades, the ability to predict complex flows of industrial significance using computational fluid dynamics (CFD) has advanced tremendously. CFD has become an indispensable tool in many diverse fields, including aerodynamics, turbomachinery, combustion, and others.

The basic idea of CFD is to replace the differential equations which describe fluid flow with algebraic equations which can be solved by computers. One of the most popular techniques for doing so is provided by the finite volume method. According to this method, the solution domain is filled with a mesh, which is used to define storage locations for each variable. Finite control volumes are constructed around each storage location, and the governing equations integrated over each control volume. The volume integrals are converted to surface integrals by means of Gauss' divergence theorem, and the surface integrals are approximated in terms of variables defined at the adjacent storage locations. By this process, the differential equations are replaced by algebraic equations: one for each conservation equation for each control volume.

The finite volume method is strictly conservative in the sense that global conservation is satisfied by the discrete equations. This follows provided the discrete transport through each internal face has the same magnitude but opposite sign for the two control volumes which touch the face. Consequently, if the algebraic equations for the two control volumes are added together, the terms arising from the surface integral for the face they share must cancel.

For time-dependent problems, the finite volume principle has traditionally been used to discretize the spatial dimensions only. Time has been discretized using a finite difference procedure, such as the Euler or Runge-Kutta methods. If the mesh undergoes motion these methods require the use of the Leibnitz Rule to account for mesh motion. Global conservation is satisfied provided the geometrical conservation law (GCL) [15, 67, 75] is satisfied. If, however, the mesh topology changes with time

(for instance by vertex insertion or removal), these methods are not conservative.

The motivation for the current work is the observation that conservation principles apply to both space and time. Consequently, it makes sense to extend the finite volume principle also to the time dimension. The assertion that "if indeed finite elements have advantages in space, they should also have advantages in space-time" [35] is equally true of finite volumes. We call the resulting technique the *integrated space-time* (IST) finite volume method. With this method, the space-time solution domain is filled with a space-time mesh, which is used to construct space-time control volumes. The governing equations are integrated over each space-time control volume, and the volume integrals are converted to surface integrals using Gauss' divergence theorem. The IST finite volume method is conservative in space-time provided the discrete transport through each internal space-time face has the same magnitude but opposite sign for the two control volumes which touch the face. Consequently, there is no need to consider the Leibnitz Rule or the GCL.

Potential application of the IST finite volume method occurs wherever conservation in time is important. In many cases, existing finite volume discretizations, in particular those with fixed grids or with moving grids which satisfy the geometrical conservation law, already satisfy this property. However, extension of these methods to flows where remeshing is required is problematic. One may identify two classes of problems where this may occur: moving boundary problems where the boundary motion is severe, and time-accurate mesh adaptation requiring the insertion and removal of points.

In this thesis, only moving boundary problems are considered. One particularly interesting type of moving boundary problem involves free surface flow, in which the boundary motion itself is an outcome of the solution.

1.2 Objectives

The primary objective of the thesis is to demonstrate that the IST concept does lead to a viable algorithm for moving boundary problems. This will be performed by considering several smaller objectives:

1. The work is built around the extension of the finite volume method to space-time. The first objective is to develop and test a robust, second-order accurate finite volume method for steady-flow problems on unstructured meshes. The unstructured capability is deemed important because the types of unsteady motions to be considered are quite general.
2. Just as spatial finite volume methods require a spatial mesh to fill the spatial domain, so also the IST finite volume method requires a space-time mesh to fill the space-time domain. A second objective of this work is the development of a space-time meshing strategy for moving boundary problems.
3. Another objective is to develop a robust, second-order accurate IST finite volume solver for unsteady problems involving prescribed boundary motion.
4. The final objective is to apply the IST method to free surface flow.

In order to demonstrate the method most effectively, our attention will be restricted to the incompressible flow of constant-property fluids in one and two spatial dimen-

sions. This is done for convenience only. In developing the method, we have attempted to formulate the method in such a way that its extension to more complex flows is relatively straightforward. The only exception is the space-time meshing algorithm, which as it stands would require significant effort to extend to three dimensions.

1.3 Outline

The thesis is divided into seven chapters, of which this is the first. The second chapter will discuss the differential equations describing fluid flow together with a review of the relevant literature. In the third chapter, a two-dimensional finite volume solver for steady flows will be developed and validated.

The IST finite volume method for unsteady flows is presented in Chapters 4-6. Chapter 4 describes the space-time meshing algorithm, Chapter 5 the development and validation of the solver, and Chapter 6 the extension to free-surface flow.

Some conclusions and recommendations for further study are given in the final chapter.

Chapter 2

Background

In order to place the IST finite volume method into context, it is important to develop a sense of existing methodologies. This chapter will provide a framework for doing so. First, the physics of fluid flow, as described by the Navier-Stokes equations and a simpler companion equation, will be reviewed. Traditional approaches to discretizing these equations will then be summarized, and existing space-time methods will be introduced. Following that, some issues related to mesh motion will be discussed, and the chapter will conclude with a discussion of the physics and modelling of free surface flow.

2.1 Mathematical Basis

Fluid dynamics is described by a coupled set of equations representing conservation of mass, momentum, and energy. Historically, the equations representing conservation of momentum together with the assumption of a linear stress-strain rate relationship was labelled the *Navier-Stokes equations*; however, in some circles this label has also come to include the continuity and energy equations. This convention is followed here. Only the simplified case of incompressible flow with constant property fields is considered, which renders the energy equation redundant. Laminar flow is also assumed. Then, using Cartesian tensor notation, the continuity equation is

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{2.1}$$

and the momentum equation is

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_j u_i)}{\partial x_j} = -\frac{\partial \tilde{p}}{\partial x_i} + \rho g_i + \frac{\partial \tau_{ji}}{\partial x_j}. \tag{2.2}$$

where u_i is the velocity in the x_i-direction, t is time, ρ is the fluid density, $\bar{p}$ is the pressure, and g_i is the gravitational acceleration vector. The deviatoric stress tensor τ_{ji} is obtained from the following constitutive relationship:

$$\tau_{ji} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \tag{2.3}$$

where μ is the dynamic viscosity.

For many flows, the pressure gradient term of the momentum equation nearly balances the gravitational force term, and in the hydrostatic limit they balance exactly. For this reason the terms are typically combined by defining a modified pressure p from

$$-\frac{\partial p}{\partial x_i} = -\frac{\partial \bar{p}}{\partial x_i} + \rho g_i, \tag{2.4}$$

in which case the momentum equation becomes

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial(\rho u_j u_i)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ji}}{\partial x_j}. \tag{2.5}$$

The relationship between p and $\bar{p}$ may be obtained by noting that ρg_i, being irrotational, may be written as the gradient of a potential energy function U:

$$\frac{\partial U}{\partial x_i} = -\rho g_i. \tag{2.6}$$

Solving for U yields

$$U = -\rho g_i r_i + C, \tag{2.7}$$

r_i being the position vector. Using Eqs. (2.6) and (2.7), Eq. (2.4) may be written as

$$-\frac{\partial p}{\partial x_i} = -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial(\rho g_j r_j)}{\partial x_i}. \tag{2.8}$$

By integrating this equation and choosing $p = \bar{p} = 0$ at a reference location $r_{i,\mathrm{ref}}$, we obtain

$$p = \bar{p} - \rho g_i (r_i - r_{i,\mathrm{ref}}), \tag{2.9}$$

which identifies the modified pressure as being the deviation of the true pressure from the hydrostatic pressure.

When developing discretization methods, it is useful to consider a simplified conservation equation which retains some of the mechanisms found in general fluid dynamics. This *scalar conservation equation* describes the conservation of a generic scalar ϕ in the presence of a known flow field:

$$\frac{\partial(\rho\phi)}{\partial t} + \frac{\partial(\rho u_i \phi)}{\partial x_i} + \frac{\partial q_i}{\partial x_i} = \dot{S}, \tag{2.10}$$

where $\dot{S}$ is the volumetric generation rate of ϕ. The diffusive flux vector q_i is obtained from the following constitutive relationship:

$$q_i = -\Gamma \frac{\partial \phi}{\partial x_i}, \tag{2.11}$$

where Γ is the diffusion coefficient.

Although the scalar conservation equation is incomplete for general fluid dynamics problems, it does describe many important physical processes. In particular, it expresses the time rate of change of ϕ to be a result of imbalances in the advective and diffusive fluxes and volumetric sources. The equation is mathematically rich — in the general case, it is parabolic, but at steady state, it is elliptic, and in the absence of diffusion, it is hyperbolic.

2.2 Discretization Techniques

Two numerical frameworks — the finite element and finite volume methods — have found wide use in solving scalar transport and fluid flow probiems in complex geometries. Other techniques which have been developed for restricted classes of flows and geometries will be considered only where appropriate.

Both the finite volume and finite element methods involve discretizing the governing differential equations; that is, replacing them with discrete analogues which can be solved using digital computers. The finite element method was first developed for the analysis of solid mechanics, whereas the finite volume method was first applied to fluid mechanics. Since then, both methods have been extended to other classes of problems, but they often retain their historical terminologies.

In the finite element method, the domain is covered with a computational mesh consisting of nodes and elements. The solution within each element is described by local shape functions, and the global solution field is obtained by piecing together the local element solutions. The discrete equations are obtained by integrating the governing equations over the domain with respect to a test function. Different choices for the test function yield different discretizations: the most common choice, called Galerkin's method, is to use the shape functions.

In the finite volume method, the domain is covered with a computational mesh consisting of vertices and cells, which correspond to the nodes and elements of the finite element method. The cells are used to construct control volumes, over which the governing equations are integrated. In *cell-centered* methods, the volumes are the cells themselves, whereas in *vertex-centered* methods (also called *cell-vertex* [9], *element-based finite volume* [55], and *control volume finite-element* [7, 61] methods), the control volume is constructed from a dual mesh. Within each volume or element, a solution profile is assumed and used to obtain discrete approximations for the fluxes between adjacent volumes.

In certain cases, the finite element and vertex-based finite volume methodologies lead to similar or even identical discrete equations [9]. Thus the finite volume method sometimes has an equivalent finite element interpretation and *vice versa*. For these cases, the advantages of one method (such as conservation or ease of mathematical analysis) are shared by the other.

In this review, a finite volume perspective is adopted except in those cases where other methods do not have a corresponding finite volume interpretation. The discretization of the scalar conservation equation will be discussed first, followed by the Navier-Stokes equations.

2.3 The Scalar Conservation Equation

In the finite volume method, the scalar transport equation is integrated over each control volume Ω to obtain

$$\int_\Omega \frac{\partial(\rho\phi)}{\partial t}\,d\Omega + \int_\Omega \frac{\partial(\rho u_i\phi)}{\partial x_i}\,d\Omega + \int_\Omega \frac{\partial q_i}{\partial x_i}\,d\Omega = \int_\Omega \dot{S}\,d\Omega. \tag{2.12}$$

The advection and diffusion terms may be converted to surface integrals using Gauss' divergence theorem,

$$\int_\Omega \frac{\partial v_i}{\partial x_i}\,d\Omega = \int_S v_i n_i\,dS, \tag{2.13}$$

where S is the boundary of Ω and n_i is its outward-directed normal. Eq. (2.12) then simplifies to

$$\int_\Omega \frac{\partial(\rho\phi)}{\partial t}\,d\Omega + \int_S \rho u_i n_i \phi\,dS + \int_S q_i n_i\,dS = \int_\Omega \dot{S}\,d\Omega. \tag{2.14}$$

Further simplification can be achieved by using the Leibnitz Rule, which accounts for control volume motion. In particular, if S moves with velocity w_i, the Leibnitz Rule for any volumetric quantity f is

$$\frac{d}{dt}\int_\Omega f\,d\Omega = \int_\Omega \frac{\partial f}{\partial t}\,d\Omega + \int_S f w_j n_j\,dS. \tag{2.15}$$

Substitution of this equation into Eq. (2.14) yields the final integral form of the scalar conservation equation:

$$\frac{d}{dt}\int_\Omega \rho\phi\,d\Omega = \int_S \rho\phi(w_i - u_i)n_i\,dS - \int_S q_i n_i\,dS + \int_\Omega \dot{S}\,d\Omega. \tag{2.16}$$

Possibilities for the discretization of the various terms are now considered.

2.3.1 Advective Flux

The advective flux is typically discretized independently of the diffusive flux, such that it is treated in the same way as if the conservation equation were hyperbolic. Mathematical and numerical issues related to hyperbolic conservation laws are discussed by LeVeque [37]. An important point is that the advective flux is closely related to the transient term through the characteristic equations. In particular,

signals are propagated along the characteristics from the upstream direction, leading to a class of upwind differencing schemes. An effective way of doing so was found by Godunov [27]. He assumed a piecewise constant solution for each time step, and then exactly solved the resulting local Riemann problems at the control volume interfaces to yield the solution at the next time step.

The main source of inaccuracy in Godunov's method is the assumption of a piecewise constant solution, which leads to significant numerical diffusion. By assuming a piecewise linear solution, van Leer [69] obtained a second order scheme known as MUSCL. Like all higher-order advection discretizations, MUSCL requires that special measures be taken in order to avoid spurious overshoots and undershoots at extrema. This can be done by using a flux or slope limiter, which reduces the scheme to first order at extrema. Sweby [64] later showed that van Leer's limiter is a special case of a class of high-resolution oscillation-free schemes known as *total variation diminishing* (TVD) methods [32].

Godunov's method and TVD concepts were derived for one-dimensional conservation equations. For many years, they were applied in a one-dimensional fashion to multi-dimensional problems. Barth and Jesperson [10] took a major step forward by developing a multi-dimensional linear reconstruction algorithm and flux limiter which can be applied to unstructured meshes. The reconstruction algorithm relies on the computation of cell gradients using an appropriate form of the Green-Gauss theorem [10] or a least-squares procedure [8]. Alternative limiters have also been developed [2, 71, 76].

Another approach to discretizing the advective flux is to allow the computational mesh to move with the flow, leading to a Lagrangian formulation. Lagrangian methods have the advantage that the advective flux calculation is free of numerical diffusion; however, remeshing and projection steps may be required to avoid mesh distortion. The *arbitrary Lagrangian-Eulerian* (ALE) method [33] avoids excessive distortion by allowing the mesh to move with an arbitrary velocity.

2.3.2 Diffusive Flux

The diffusive term has an elliptical character; that is, it has no preferential direction of influence. It is therefore appropriate to use a centered discretization for it. In the finite element method, Galerkin's method is used almost exclusively, and vertex-centered finite volume methods often yield the same discretization as Galerkin's method [9, 42, 62]. Cell-centered methods also use a centered discretization; but, because they do not use shape functions, they require gradient information from adjacent cells [20, 41], vertices [23], or auxiliary cells [3, 48].

An important consideration in diffusive flux calculations is the maximum principle. If a pure diffusion problem is considered, it may be shown analytically that the maximum value of ϕ must lie on the boundary. A discrete solution should satisfy an analogous discrete maximum principle; but, as discussed in Chapter 3, this is not always achievable.

2.3.3 Source Terms

When volumetric terms are integrated over the volume, the mass matrix appears. For cell-centered methods, the mass matrix is diagonal. For vertex-centered methods, the mass-matrix has off-diagonal entries, but is often replaced with an diagonal approximation through a process called *mass lumping* [72].

2.3.4 Transient Term

After discretizing the advective flux, diffusive flux, and source terms, an ordinary differential equation in time results. The discretization of this operator gives rise to the same mass matrix as in the source term [72]. If the mass matrix is diagonal, the solution may be explicitly evolved in time using a forward Euler approximation. Stability restrictions restrict the time step size for explicit methods based on the Courant-Friedrichs-Levy (CFL) number and the diffusion number. These restrictions imply that many iterations may be required to achieve steady-state solutions, but the solution may be accelerated using multigrid algorithms [72].

The solution may also be evolved in time using implicit methods, such as the backward Euler approximation. These methods involve the solution of a system of equations at each time step. As a result, they require more computational effort than explicit schemes, but also have much better stability characteristics.

Both the forward and backward Euler methods are first-order accurate in time. Methods such as Runge-Kutta schemes may be used to achieve higher-order accuracy.

2.4 The Navier-Stokes Equations

As with the scalar equation, the finite volume discretization of the Navier-Stokes equations starts by integrating them over each control volume, again using the divergence theorem and the Leibnitz Rule to simplify the volume integrals. The integral form of the continuity equation thus becomes

$$\frac{d}{dt}\int_\Omega \rho\, d\Omega = \int_S \rho(w_i - u_i)n_i\, dS, \tag{2.17}$$

and the momentum equation becomes

$$\frac{d}{dt}\int_\Omega \rho u_i\, d\Omega = \int_S \rho u_i(w_j - u_j)n_j\, dS - \int_S p n_i\, dS + \int_S \tau_{ji} n_j\, dS. \tag{2.18}$$

When discretizing the Navier-Stokes equations, the viscous, transient, and source terms are typically handled in the same manner as their scalar counterparts, without significant complications. The manner in which the remaining terms (advection and pressure) are discretized depends on the nature of the flow.

For compressible flow, the first-order terms form a hyperbolic set and are discretized together with the energy equation using characteristic-based upwinding

methods. These methods include flux vector splitting [5, 63, 70], flux difference splitting [60], and fluctuation splitting [14].

For incompressible flow, the equations have a mixed hyperbolic/elliptic character, and the standard compressible methods are inappropriate. One approach is to restore the hyperbolic character of the equations by adding a time derivative of pressure to the continuity equations; this approach is called the artificial compressibility method [13]. It may be interpreted as preconditioning the equations to scale to the advective velocity rather than the acoustic velocity [73].

Other techniques for incompressible flow start by acknowledging its special character up front. The elliptic character can be traced to the action of pressure, for which a Poisson-type equation may be derived [20]. For this reason, pressure must have a centered discretization. Unfortunately, when this is done, a decoupled "checkerboard" solution may develop [50]. For many years the pressure checkerboard problem was overcome using staggered grids [31, 50]. More recently, the need to solve flow on nonorthogonal meshes in arbitrary geometries has driven the development of colocated methods, the most common of which is due to Rhie and Chow [59], and developed further by others [39, 53]. These colocated methods suppress the pressure decoupling mode by introducing a fourth-order pressure-smoothing term in the continuity equation, so that the interpolated cell-face velocity is sensitive to the local pressure gradient. This operation may be interpreted as numerically distinguishing between two velocities: that which advects conserved quantities and that which is itself advected.

An alternative approach, called the continuity constraint method (CCM) has recently been proposed by Williams and Baker [74]. They do not attempt to remove the decoupling, but do prevent it from contaminating the solution by calculating a smooth pressure field from the pressure Poisson equation.

Various strategies exist for solving the equation set. Most incompressible solvers are implicit, because the time-step restriction of explicit methods is prohibitively small. Implicit solvers can be either coupled or segregated. Coupled solvers, such as coupled algebraic multigrid [58], are ideal for rapid convergence. Segregated solvers, on the other hand, require less memory to store the coefficient matrix; the most common algorithms are SIMPLE [50] and its variants, which guess values for the velocity and pressure fields and then iteratively solve the momentum and pressure equations to correct the guesses.

2.5 Space-Time Discretizations

The discretization techniques discussed above use one type of discretization for the spatial terms and another for the time derivative. *Streamline diffusion* (SD) finite element methods are unique in coupling the discretization of space and time. These methods subdivide the space-time domain into slabs, each of which is composed of space-time elements. The elements can be oriented according to an arbitrary velocity, leading to a relationship with the ALE method [33]. If the space-time elements are aligned along the characteristics of the associated hyperbolic problem, the SD method is referred to as a *characteristic streamline diffusion* (CSD) method [29, 30]. SD methods are typically coupled with the discontinuous Galerkin method in order to retain a time-marching algorithm. They also modify the stan-

dard Galerkin discretization by adding stabilizing terms which vanish as the exact solution is reached [11, 35]. Like ALE methods, SD methods may require remeshing steps to avoid highly distorted meshes. The remeshing may be made conservative by using simplex space-time elements [24].

A new space-time discretization, called the *space-time conservation element method*, has recently been proposed by Chang [12]. In this method, the space-time domain is divided into conservation elements, over which flux balances are enforced; and solution elements, which separate the conservation elements. The solution is assumed to vary smoothly within the solution elements, leading to simple expressions for the fluxes between the conservation elements. Both the unknowns and their spatial derivatives are treated as independent variables.

2.6 Moving Meshes

There are several motivations for being able to solve flows on time-dependent meshes. For instance, there are many cases in which the domain boundary moves according to some prescribed, free, or compliant condition. In other cases, a transient solution-adaptive solver may be desired, requiring vertex insertion, movement, or deletion. In this section, some of the issues and possibilities related to discretizations for moving meshes are explored. This includes some geometrical constraints which must be satisfied and a review of methods for free-surface flows.

Conservative discretizations on moving grids must satisfy a geometrical constraint known as the *geometrical conservation law* (GCL) [67]. It may be derived from the Leibnitz Rule (Eq. (2.15)) by choosing $f = 1$:

$$\frac{d}{dt}\int_\Omega d\Omega = \int_S w_j n_j \, dS. \tag{2.19}$$

An interpretation of this equation is that the change in the domain volume during a time interval must equal the volumetric changes along the domain boundaries. If face velocities which violate the GCL are used, spurious mass sources and convergence difficulties may be experienced [15, 75].

2.7 Free Surface Flows

A special class of problems involving moving meshes involves *free-surface flow*, in which the boundary location is not known *a priori* but must rather be determined as part of the solution procedure. A common class of free surface flow involves liquid-vapour interfaces with no heat or mass transfer between the phases.

Methods for free surface flows may be grouped into two categories. *Interface-tracking* methods treat the surface as a true discontinuity, and can therefore apply appropriate boundary conditions at the interface. *Interface-capturing* methods treat the two phases together without explicitly considering the interface; as a consequence, the computed interface is smeared. There is, however, some ambiguity in this classification. For instance, the original volume-of-fluid [34] method treats the interface as a true discontinuity and is therefore be classified as an interface-tracking

method [21, 68]. However, the same method can bypass the interface reconstruction step, in which case it can be classified as interface-capturing [46].

In this review only interface-tracking methods are considered. Two types of boundary conditions must be considered: the *kinematic condition* and the *dynamic conditions*.

2.7.1 Kinematic Condition

The free surface is a material surface. It is this *kinematic condition* which permits the interface location to be tracked. One interpretation of the condition is that no mass may flow through the surface:

$$u_i n_i = u_{i,\mathrm{fs}} n_i, \tag{2.20}$$

where $u_{i,\mathrm{fs}}$ is the surface velocity. Alternatively, a particle which lies on the surface must remain attached to it:

$$v = \frac{Dh}{Dt} = \frac{\partial h}{\partial t} + u\frac{\partial h}{\partial x}, \tag{2.21}$$

where u and v are the components of the velocity vector and $h = h(x, y, t)$ is the surface elevation. Both of these forms of the kinematic condition are commonly used in free surface calculations. Although they are mathematically equivalent (provided the surface elevation is single-valued), they may lead to different numerical results.

There have been several approaches to enforcing the kinematic condition. The earliest were the fixed-mesh Eulerian methods, where a sharp interface is reconstructed using information obtained by solving additional equations which satisfy Eq. (2.21) in a domain encompassing the interface. The two most common methods of this type are the *marker-and-cell* (MAC) method [31], where massless particles in the fluid phase are tracked; and the *volume-of-fluid* (VOF) method [34], where a transport equation for liquid volume fraction is solved. A strength of these methods is their applicability to flows involving large surface motions, such as breaking waves [43].

Another approach is obtained using a Lagrangian formulation, where the kinematic condition is satisfied simply by having the mesh points move with their local velocity. A related class of methods is obtained using the space-time finite element concept, which has been applied to free surface flow by Hansbo [30] and Tezduyar *et al.* [65]. The drawback of these methods is the nonconservative projection step which may be required to maintain mesh quality.

A third approach to tracking the interface is offered by adaptive-Eulerian schemes, where the mesh conforms to the free surface and the interior mesh responds to the boundary motion without requiring remeshing. Some of these methods determine the interface location by explicitly integrating Eq. (2.21) using velocities at the free surface [4, 19, 44]. This form of the kinematic condition, however, may not guarantee overall mass conservation [49]. Other adaptive-Eulerian methods [45, 55, 66] enforce the conservative form of the kinematic condition, Eq. (2.20), by forcing the mass flow rate through the free surface to zero. Some methods enforce

the kinematic condition together with the continuity equation, so that the continuity equation for cells next to the boundary yields the free surface position [55,66]. In other studies, the kinematic condition and continuity equation are solved in a segregated manner [45]. Adaptive-Eulerian methods must also devise ways to have the interior mesh points respond to the free surface motion. A common approach is to have them slide along predefined spines [36].

2.7.2 Dynamic Conditions

The *dynamic conditions* arise from the condition of force equilibrium at the interface. If the liquid and vapour phases are denoted by the superscripts l and g, the dynamic conditions may be expressed as [38]

$$(\bar{p}^l - \bar{p}^g + \sigma\kappa)n_i = (\tau^l_{ji} - \tau^g_{ji})n_j - \frac{\partial\sigma}{\partial x_i}, \tag{2.22}$$

where σ is the surface tension coefficient, n_i points toward the vapour phase, and κ is the mean surface curvature (positive if the surface is concave in the direction of n_i). The ideal free surface is obtained if the density differences between the liquid and vapour are large, in which case the only effect of the gas phase is to exert a pressure on the surface. Further simplification results if capillary effects arising from the surface tension and normal stresses at the interface can be neglected, in which case the dynamic conditions simplify to

$$\tilde{p} = \bar{p}_{\text{fs}} \tag{2.23}$$

for the normal direction and slip conditions for the tangential directions.

Chapter 3

Discretization for Steady Flows

In this chapter the computation of steady two-dimensional incompressible flow is described. Several interconnected tasks must be addressed: mesh generation, control volume definition, equation discretization, and linear equation solution. Each of these components will be considered in turn. In addition, some test cases which verify and validate the method will be presented.

3.1 Mesh Generation

Mesh generation involves filling the spatial domain with nonoverlapping *cells*. The cell boundaries are called *faces*, even though geometrically they are edges, to emphasize the role that they play in transferring discrete fluxes into and out of control volumes.

There are two types of meshes. Structured meshes arose first, since they are described by simple data structures. In the past decade, however, unstructured mesh technologies have become common, because of their convenience for complex geometries. In anticipation of extending our method to complex boundary motion, we have considered unstructured meshes to be important. Triangular meshes are generated using a publicly-available package called Easymesh [47], which uses an incremental insertion algorithm together with Laplacian smoothing. It writes data files containing several useful data structures, but its speed degrades significantly when the mesh has more than about 10,000 vertices.

3.2 Choice of Control Volumes

After generating the mesh, the control volumes must be constructed. In Chapter 2, two common approaches for doing so were described: cell-centered methods, where the volumes are the cells themselves; and vertex-centered (or cell-vertex) methods, where the volumes are associated with vertices. Each method has advantages and

disadvantages. In this section, the methods are compared based on accuracy/cost ratio, monotonicity, special requirements, and personal preference.

An ideal discretization method should have a high accuracy/cost ratio, where cost involves both CPU time and storage. Comprehensive comparisons of this ratio between cell-centered and vertex-centered methods are not available, but there is some indication that on tetrahedral meshes, cell-centered methods have both a higher cost and a higher accuracy [72].

Monotonicity is also an important consideration: if the physical process being modelled possesses a maximum principle, then so should the discrete solution. This consideration favours cell-centered methods in some cases but vertex-centered methods in other cases. When solving a pure diffusion problem on a rectangular mesh, vertex-centered methods are guaranteed to be monotone only if the aspect ratio is close to unity, whereas cell-centered methods are always monotone. With triangular meshes, on the other hand, cell-centered methods possess no known discrete maximum principle, while vertex-centered methods are monotone provided the mesh is Delaunay [9]. This analysis does not extend easily to three-dimensional tetrahedral meshes, where monotonicity is more difficult to attain with vertex-centered methods. Additional issues arise when the mass matrix appears, for off-diagonals in the mass matrix may introduce nonphysical extrema. Cell-centered methods have a diagonal mass matrix by default, whereas its diagonalization with vertex-centered methods requires the mass-lumping approximation.

In some cases, special factors which must be considered may affect the choice. For example, material discontinuities are more easily handled by cell-centered methods, in which the discontinuity lies along control volume boundaries, than with vertex-centered methods, in which the discontinuity lies internal to control volumes. Space-time discretizations pose a similar complication for vertex-centered methods, as discussed in Chapter 5. On the other hand, it will be seen in Chapter 6 that free-surface flow modelling presents additional difficulties for cell-centered methods.

The final issue involved in the choice has to do with personal preference. For some, cell-centered methods are the most intuitive translation of the finite volume principle into a numerical algorithm. Others prefer vertex-centered methods because their mathematical properties are more easily analyzed [22]. It is perhaps not surprising, then, that cell-centered methods are particularly popular in the engineering community, while vertex-based methods are more common in the mathematics community.

In the present work, a cell-centered approach is chosen. The choice is based on convenience, for the space-time extension more easily allows for a time-marching algorithm.

3.3 The Scalar Conservation Equation

Having chosen the control volumes, we now seek to form discrete conservation equations for each. The discretization procedure will be described for the scalar conservation equation in this section and for the Navier-Stokes equations in the next.

The complete integral form of the scalar conservation equation was given earlier in Eq. (2.16). Here some simplifications are made by considering only steady source-

free transport. The simplified equation is

$$\int_S \rho\phi u_i n_i \, dS + \int_S q_i n_i \, dS = 0, \tag{3.1}$$

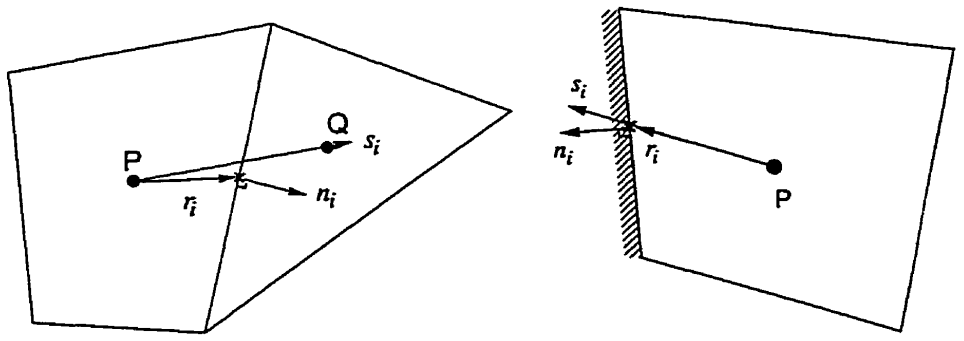
or, in vector notation,

$$\int_S \rho\phi \mathbf{u} \cdot \hat{\mathbf{n}} \, dS + \int_S \mathbf{q} \cdot \hat{\mathbf{n}} \, dS = 0. \tag{3.2}$$

The discretization process begins by applying the *midpoint rule*, wherein the surface integrals are approximated at the face midpoints surrounding each control volume. The resulting discrete control volume equations have the form

$$\sum_f (F_f^a + F_f^d) = 0, \tag{3.3}$$

where F_f^a and F_f^d respectively represent the numerical advective and diffusive transport through each face. Typical faces, together with some associated geometrical entities, are illustrated in Figure 3.1. Important vectors shown on the diagram include n_i (or $\hat{\mathbf{n}}$), the unit outward-directed normal; s_i (or $\hat{\mathbf{s}}$), the unit vector joining cell centroids; and r_i (or $\mathbf{r}$), the vector joining a cell centroid to the face midpoint.



(a) An internal face (b) A boundary face

Figure 3.1: Typical control volume faces and geometrical nomenclature.

At internal faces, conservative numerical fluxes are required; that is, they must have equal magnitude and opposite sign for the two cells adjacent to the face. At boundary faces, appropriate boundary conditions must be applied. Expressions for the numerical fluxes in terms of the cell centroid values of ϕ will be given below. First, however, an algorithm for calculating cell gradients of ϕ, which are used in the flux approximations, will be presented.

3.3.1 Cell Gradient Vectors

The numerical flux approximations require estimates for cell gradient vectors $\nabla\phi$. There are two common techniques for evaluating the cell gradients: a Green-Gauss theorem [10] and a least-squares approach [8]. We choose the least-squares approach for two reasons. First, it is exact for linear profiles, whereas some effort is needed to make the Green-Gauss method linearly exact. Second, it extends more naturally to space-time, which is useful for the IST algorithm.

To understand the least-squares algorithm, consider a particular cell P and its set of immediate neighbours η_P, as illustrated in Figure 3.2. (If P is adjacent to a boundary, the boundary face must also be considered a neighbour.) The change in centroid values between neighbour j and P is given by $\phi_j - \phi_P$, $j \in \eta_P$. If the cell gradient $\nabla\phi|_P$ is exact, then this difference is also

$$\phi_j - \phi_P = \nabla\phi|_P \cdot (\mathbf{r}_j - \mathbf{r}_P), \tag{3.4}$$

where $\mathbf{r}_j - \mathbf{r}_P$ is the vector from cell P to cell j But unless the solution is linear, the cell gradient cannot be exact, for cell P has more neighbours than the gradient vector has components. The least-squares gradient is that which minimizes

$$\sum_{j \in \eta_P} w_j \left[\nabla\phi|_P \cdot (\mathbf{r}_j - \mathbf{r}_P) - (\phi_j - \phi_P)\right]^2,$$

where w_j is a weighting factor. We choose $w_j = 1/|\mathbf{r}_j - \mathbf{r}_P|^2$, which tends to favour each neighbour in the stencil equally.

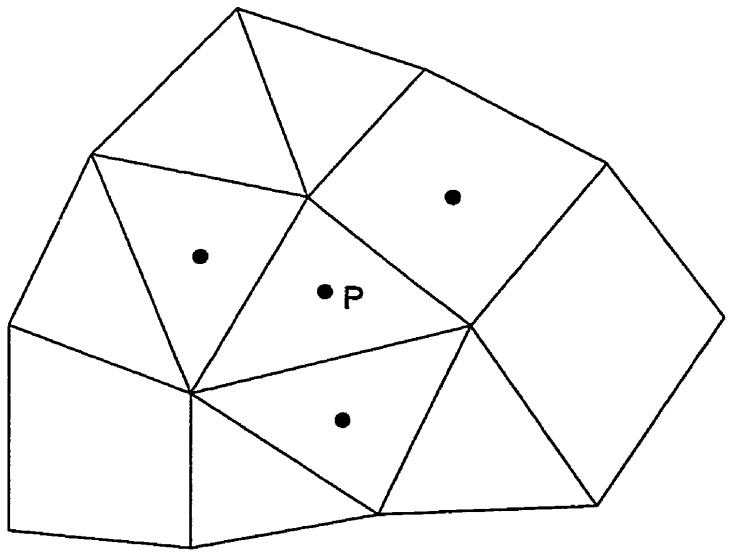


Figure 3.2: Least-squares stencil used in calculating cell gradient vectors.

The solution to this least-squares problem requires solution of the following matrix equation:

$$\begin{bmatrix} \sum w_j \Delta x_j \Delta x_j & \sum w_j \Delta x_j \Delta y_j \\ \sum w_j \Delta x_j \Delta y_j & \sum w_j \Delta y_j \Delta y_j \end{bmatrix} \begin{pmatrix} \phi_x \\ \phi_y \end{pmatrix} = \begin{pmatrix} \sum w_j \Delta x_j \Delta \phi_j \\ \sum w_j \Delta y_j \Delta \phi_j \end{pmatrix}, \tag{3.5}$$

where

$$\begin{aligned} \Delta x_j &= x_j - x_P, \\ \Delta y_j &= y_j - y_P, \\ \Delta \phi_j &= \phi_j - \phi_P, \end{aligned}$$

and ϕ_x and ϕ_y are the components of $\nabla\phi|_P$.

When cell gradients appear in the discrete fluxes, they are lagged, which in some cases may hamper convergence. It has been found empirically that convergence may be significantly improved by underrelaxing the gradient calculations as follows:

$$\nabla\phi|_P = \omega\, \nabla\phi|_P^{\text{new}} + (1-\omega)\, \nabla\phi|_P^{\text{old}}. \tag{3.6}$$

This form of underrelaxation is more effective than underrelaxing the entire solution field, for it hones in on the actual cause of convergence difficulties. Typically $\omega = 0.8$.

3.3.2 Advection Term

The advective transport is given by

$$F_f^a = J_f \phi_f, \tag{3.7}$$

where

$$J_f = \rho \mathbf{u} \cdot \hat{\mathbf{n}} S_f \tag{3.8}$$

is the mass flow through the face. An upwind-biased discretization is used for ϕ_f:

$$\phi_f = \phi_{\text{up}} + \Phi \nabla\phi \cdot \mathbf{r}|_{\text{up}}. \tag{3.9}$$

$\Phi = 0$ assumes a piecewise constant solution and yields a first-order upwind method. $\Phi = 1$ assumes a piecewise linear solution, and applies a second-order correction using the upwind cell gradient. Like any second-order advection discretization, this may lead to spurious overshoots and undershoots, which can be reduced or eliminated by limiting Φ near extrema. The limiter of Barth and Jesperson [10] was the first one developed for unstructured meshes. It is based on the principle that, when a cell gradient is used to reconstruct ϕ at the face midpoints, the reconstructed values must be bounded by the values at the cell and its neighbours. Unfortunately, this limiter has a slope discontinuity, which may produce convergence difficulties. Venkatakrishnan [71] has proposed a modification which improves convergence at

the expense of strict monotonicity. The modified limiter has an adjustable parameter K which controls the magnitude of the allowable overshoots and undershoots. Both limiters have been tested in the current work.

The advective flow must also be evaluated at boundary faces. At wall boundaries, where there is no flow through the face, $J_f = 0$. At inflow boundaries, ϕ_f must be specified, and at outflow boundaries, ϕ_f is obtained using the same expression as at internal faces.

In setting up the matrix equation for the advective flux, the first-order upwind term is made active, while the second-order correction is lagged by putting it into the right-hand side of the matrix equation.

In order to validate the advection scheme and illustrate some of its properties, consider the advection of ϕ in a square geometry. The mesh is shown in Figure 3.3. The rotational velocity field is defined by $\mathbf{u} = y\hat{\mathbf{i}} - x\hat{\mathbf{j}}$. A square-wave profile for ϕ is specified along the inflow boundary, which is placed along a cut-line to the left of the origin. This square-wave profile should be maintained over the course of a rotation. Four solutions are shown in Figure 3.4. As expected, the first-order upwind is very diffusive, while the pure second-order method introduces new extrema near the discontinuities. Two solutions with Venkatakrishnan's limiter are also provided: with $K = 1$, the overshoots and undershoots are very small, while with $K = 5$ they are more significant. Convergence could not be achieved with Barth and Jesperson's limiter.

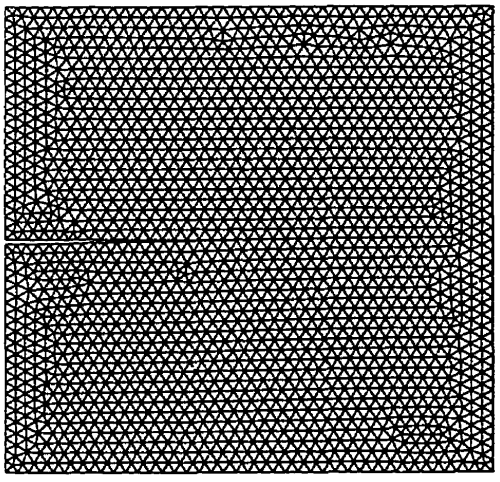
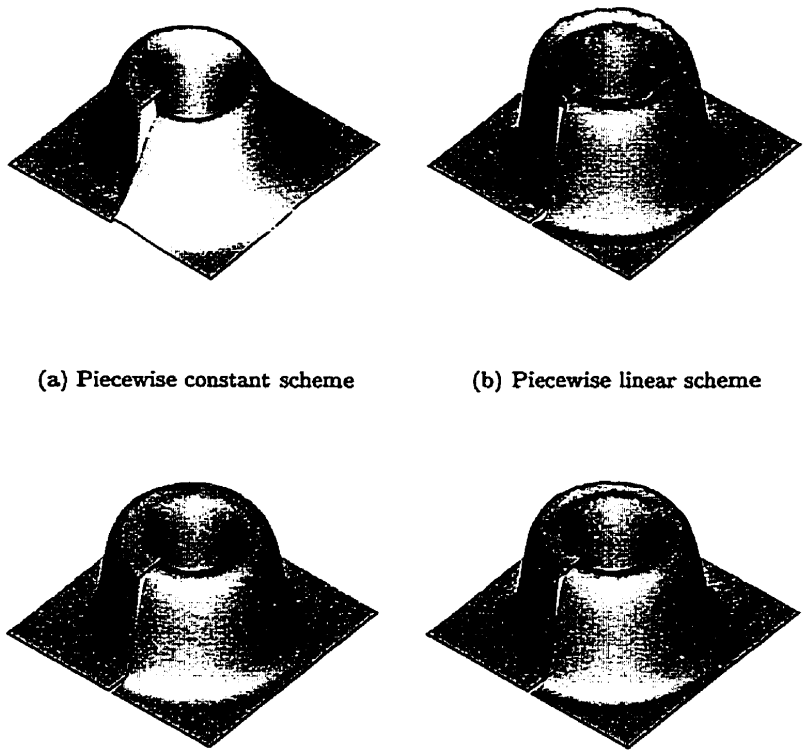


Figure 3.3: Mesh used for circular advection test case.



(a) Piecewise constant scheme

(b) Piecewise linear scheme

(c) Limited scheme, $K = 1$

(d) Limited scheme, $K = 5$

Figure 3.4: Solutions to the circular advection test case.

The trade-off between convergence and accuracy with Venkatakrishnan's limiter is evident in Table 3.1, where the number of iterations required to reduce all normalized residuals below 10^{-5} is listed together with the magnitude of the largest overshoot or undershoot. In all cases the gradient underrelaxation factor is 0.7 and the linear equations are solved directly. (The iteration count remains the same for all cases other than the piecewise constant solution when the iterative solver of Raw [58] is used instead.) Interestingly, moderate choices for K actually give better convergence behaviour than the pure piecewise linear scheme.

Table 3.1: Convergence and accuracy behaviour for circular advection test case.

Advection scheme	Iterations required	Overshoot/undershoot magnitude
Piecewise constant	1	0%
Piecewise linear	34	15%
Limited, $K = 0.5$	58	0.9%
Limited, $K = 1$	29	1.8%
Limited, $K = 2$	26	2.9%
Limited, $K = 5$	29	5.5%
Limited, $K = 10$	31	9.9%

3.3.3 Diffusion Term

The diffusive transport is given by

$$F_f^d = \mathbf{q} \cdot \hat{\mathbf{n}} S_f, \tag{3.10}$$

or, by combining with the constitutive relationship $\mathbf{q} = -\Gamma \nabla \phi$,

$$F_f^d = -\Gamma \nabla \phi \cdot \hat{\mathbf{n}} S_f. \tag{3.11}$$

In this work, a new discretization for this term has been developed. It features several advantages: it is linearly exact, collapses to classical stencils on orthogonal meshes, extends to higher dimensions without modification, and can be extended to anisotropic continua such as space-time. The discretization proceeds by decomposing F_f^d as follows:

$$F_f^d = -\Gamma \left(\nabla \phi \cdot (\alpha \hat{\mathbf{s}}) + \overline{\nabla \phi} \cdot (\hat{\mathbf{n}} - \alpha \hat{\mathbf{s}}) \right) S_f, \tag{3.12}$$

where

$$\nabla \phi \cdot (\alpha \hat{\mathbf{s}}) = \alpha \frac{\phi_Q - \phi_P}{\Delta s}, \tag{3.13}$$

$\overline{\nabla \phi}$ is the average of the adjacent cell gradients, and α is a scaling factor. We demand that $\alpha = 1$ on orthogonal meshes in order for the method to collapse to

classical stencils. Methods developed for structured grids amount to $\alpha = 1/\hat{\mathbf{n}} \cdot \hat{\mathbf{s}}$ [41]. Others [20] suggest

$$F_f^d = -\Gamma \left(\nabla\phi \cdot \hat{\mathbf{s}} + \overline{\nabla\phi} \cdot (\hat{\mathbf{n}} - \hat{\mathbf{s}}) \right) S_f, \tag{3.14}$$

which is equivalent to $\alpha = 1$. The optimal choice is $\alpha = \hat{\mathbf{n}} \cdot \hat{\mathbf{s}}$, because it shifts the projection of $\hat{\mathbf{n}} - \hat{\mathbf{s}}$ onto $\hat{\mathbf{s}}$ into the first term, as illustrated in Figure 3.5. As a result, $\overline{\nabla\phi}$ is used only for the nonorthogonal contribution to F_f^d. This choice also extends unambiguously to anisotropic diffusion problems, as described in Chapter 5. The only known disadvantage of this diffusion discretization, which is shared by all other cell-centered methods, is the lack of a discrete maximum principle.

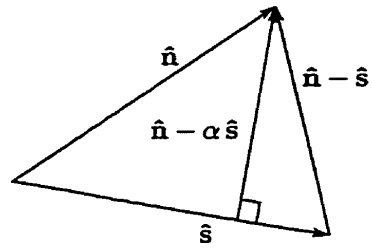


Figure 3.5: Motivation for choosing $\alpha = \hat{\mathbf{n}} \cdot \hat{\mathbf{s}}$ for the diffusion discretization.

At boundary faces, the expression for F_f^d remains the same as at internal faces, but requires one-sided approximations for some of the terms. In particular, $\hat{\mathbf{s}}$ is the vector from the cell centroid to the boundary face midpoint,

$$\nabla\phi \cdot (\alpha\hat{\mathbf{s}}) = \alpha \frac{\phi_{\text{bnd}} - \phi_P}{\Delta s}, \tag{3.15}$$

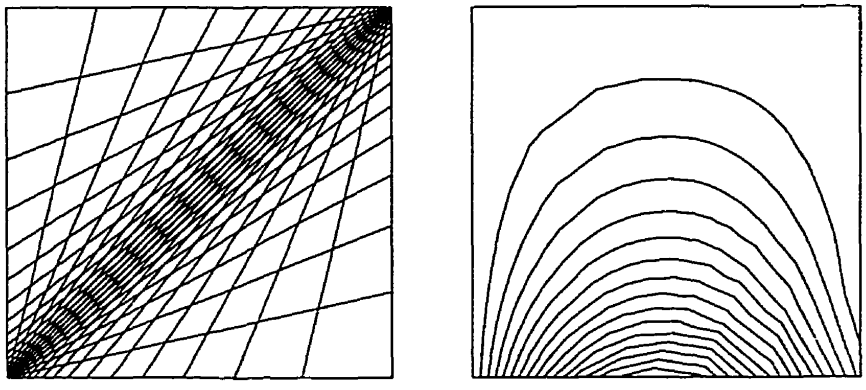
and $\overline{\nabla\phi}$ is the gradient at the cell adjacent to the boundary. ϕ_{bnd} is specified at inflow boundaries and Dirichlet walls and is extrapolated to outflow boundaries. At flux-specified wall faces, F_f^d is specified and may be used to calculate ϕ_{bnd}.

The orthogonal term of this discretization is made active, while the nonorthogonal term involving the cell gradients is lagged. On highly skewed meshes, the lagged term may become large, hamper convergence, or even cause divergence. By augmenting the active coefficient by an amount proportional to the nonorthogonality of the face, we may reduce the possibility of this occurence. In particular, the active coefficient is replaced by

$$\frac{\Gamma\alpha S_f}{\Delta s} \Rightarrow \frac{\Gamma S_f}{\Delta s} \left(\alpha + \frac{1-\alpha}{\lambda} \right).$$

λ is arbitrary, having a typical value of 0.8. This form of underrelaxation is surgical, affecting the coefficients only in potentially problematic regions. A corresponding term must be added to the right-hand side of the matrix equation so that the converged solution is unaffected by λ. It has been found that this type of stabilization, together with gradient underrelaxation, provides an effective means of achieving convergence on a wide range of problems.

As an example of the robustness of this diffusion discretization, consider the diffusion of ϕ in a box of length and width L, as shown in Figure 3.6. Along the side and top boundaries, $\phi = 0$, and along the bottom, $\phi = \sin(\pi x/L)$. The mesh quality is very poor; the smallest angle is below 10°. However, with underrelaxation factors $\omega = 0.5$ and $\lambda = 0.5$, an accurate solution has been achieved, as shown in Figure 3.6(b). 58 iterations were required to reduce the maximum normalized residual below 10^{-5}.



(a) Mesh (b) Solution contours

Figure 3.6: Diffusion test case.

Other tests have confirmed this diffusion discretization to be linearly exact and second-order for both triangular and quadrilateral meshes.

3.3.4 Solution Methodology

After calculating the fluxes through each face, and scattering the fluxes to the adjacent cells, there results one algebraic equation for each control volume. The

equation for a control volume P has the form

$$a_P\phi_P + \sum a_{\mathrm{nb}}\phi_{\mathrm{nb}} = b_P, \tag{3.16}$$

where the coefficients a_P and a_{nb} arise from the active terms and the right-hand side b_P has contributions from lagged terms and boundary conditions.

As a final step before solving the equations, the coefficients are modified in order to ensure the mass flows used in the advective fluxes are mass-conserving. This is important in problems where the velocity field is not yet converged, and is achieved by subtracting the discrete continuity equation multiplied by ϕ_P from Eq. (3.16). This modification has no effect on the converged solution, but does enhance iterative robustness [50, 54].

As the algebraic equations for neighbouring cells are linked, the new solution field $\{\phi\}$ is defined by a system of equations of the form

$$[A]\{\phi\} = \{b\}. \tag{3.17}$$

The coefficients which have been made active guarantee that the coefficient matrix $[A]$ is diagonally dominant, which ensures efficient convergence by iterative solvers.

The system of equations is actually solved in delta (or update) mode by calculating a residual field $\{r\}$ for the old solution field $\{\phi^o\}$:

$$\{r\} = \{b\} - [A]\{\phi^o\}. \tag{3.18}$$

Then, defining

$$\{\delta\phi\} = \{\phi\} - \{\phi^o\}, \tag{3.19}$$

Eq. (3.17) may be written as

$$[A]\{\delta\phi\} = \{r\}. \tag{3.20}$$

This system of equations is solved iteratively using an algebraic multigrid solver [58]. The multigrid algorithm uses adaptive coarsening rules, based on the relative coefficient strengths, so that the errors in all directions are reduced effectively.

After solving the system of equations for $\{\delta\phi\}$, the new solution field is updated using Eq. (3.19). The boundary values are also updated. Then new cell gradients are calculated, and new fluxes assembled, leading to a new system of algebraic equations. This process is repeated until the maximum normalized residual falls below a predefined tolerance. The residual for a cell P is normalized by the central coefficient a_P and the maximum difference in solution field:

$$\hat{r}_P = \left| \frac{r_P}{a_P(\phi_{\max} - \phi_{\min})} \right|. \tag{3.21}$$

A solution field calculated in this manner is stored at the cell centroids. For visualization in a post-processor, however, it is more convenient to obtain values at the mesh vertices. This is accomplished using a second-order reconstruction from the surrounding cell values.

3.4 The Navier-Stokes Equations

In this section, the discretization method developed for the scalar conservation equation will be extended to the Navier-Stokes equations. The equations were presented in integral form in Eqs. (2.17) and (2.18). Under the assumption of steady flow, they simplify to

$$\int_S \rho u_i n_i \, dS = 0, \tag{3.22}$$

representing conservation of mass, and

$$\int_S \rho u_j n_j u_i \, dS + \int_S p n_i \, dS - \int_S \tau_{ji} n_j \, dS = 0, \tag{3.23}$$

representing conservation of the i^{th} component of momentum. By approximating each surface integral at the face midpoints, the discrete form of the equations results:

$$\sum_f J_f = 0, \tag{3.24}$$

and

$$\sum_f (F^a_{f,i} + F^p_{f,i} + F^v_{f,i}) = 0, \tag{3.25}$$

where J_f, $F^a_{f,i}$, $F^p_{f,i}$, and $F^v_{f,i}$ respectively represent the mass flow, advective momentum transport, pressure force, and viscous force at each face.

In the sections below, the numerical approximations used for the advection term, viscous term, pressure term, and mass flows will be presented.

3.4.1 Advection Term

The advective transport of the i^{th} momentum component through a face is given by

$$F^a_{f,i} = J_f u_{f,i}. \tag{3.26}$$

The quantity $u_{f,i}$ is the *advected velocity*, distinct from the *advecting velocity* which appears in the mass flow J_f. The discretization used for $u_{f,i}$ is the same as that used for the advective transport of a scalar (Eq. (3.9)). Unlike the scalar case, however, there does not appear to be any motivation for using a nonlinear limiter. The difference may be traced to the mathematical nature of the equations. The scalar equation may be hyperbolic, and therefore permit discontinuities in the solution field. These discontinuities are responsible for triggerring wiggles, and the role of the limiter is to dampen the wiggles by increasing numerical diffusion near the discontinuities. The incompressible Navier-Stokes equations, on the other hand, have an elliptic character. The solutions are therefore smooth, affording no

opportunities for discontinuity-induced wiggles to arise. Wiggles may still occur, but they are caused by the pressure-velocity coupling rather than the advection term. Empirical studies have also shown the limiter to have little or no effect on the solution. For these reasons, unless otherwise stated, the Navier-Stokes solutions shown in this thesis do not use a nonlinear limiter.

Another feature of the advection term in momentum which is absent in the scalar case is its nonlinearity. In particular, velocity appears both in $u_{f,i}$ and in J_f. The discretization for J_f will be presented shortly. $F^a_{f,i}$ is linearized using *Picard iteration* [20], where J_f is based on values from the previous iteration.

3.4.2 Viscous Terms

The viscous force at a face is given by

$$F^v_{f,i} = -\mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) n_j S_f. \tag{3.27}$$

The first term in this expression is identical to F^d_f in the scalar conservation equation, and is approximated using the same new cell-centered discretization. The second term is lagged using known cell gradients.

At boundary faces, the viscous term treatment becomes more complex, for several boundary types require $F^v_{f,i}$ to be expressed using a local tangential-normal coordinate system:

$$F^v_{f,i} = F^{\text{tang}}_{f,i} + F^{\text{norm}}_{f,i}. \tag{3.28}$$

Let the subscripts n and t represent the normal and tangential directions. Then the tangential force is

$$F^{\text{tang}}_{f,i} = -\tau_{nt} t_i S_f \tag{3.29}$$

$$= -\mu \left(\frac{\partial u_t}{\partial x_n} + \frac{\partial u_n}{\partial x_t} \right) t_i S_f, \tag{3.30}$$

where τ_{nt} is the shear stress and t_i are the components of the unit tangent vector to the boundary face. The normal force is

$$F^{\text{norm}}_{f,i} = -\tau_{nn} n_i S_f \tag{3.31}$$

$$= -\mu \left(\frac{\partial u_n}{\partial x_n} + \frac{\partial u_n}{\partial x_n} \right) n_i S_f, \tag{3.32}$$

where τ_{nn} is the normal viscous stress.

The tangential force is discretized in a similar manner as at interior faces. The first term is

$$-\mu \frac{\partial u_t}{\partial x_n} S_f = -\mu \left(\alpha \frac{u_{t,\text{bnd}} - u_{t,P}}{\Delta s} + \overline{\nabla u_t} \cdot (\hat{\mathbf{n}} - \alpha \hat{\mathbf{s}}) \right). \tag{3.33}$$

The second term vanishes at wall boundaries and is calculated using cell gradients at other boundaries. Because the tangential force is calculated using tangential velocities, rather than Cartesian velocities, it is lagged. In order to prevent convergence from stalling near the boundary, the following approximation is used to generate active coefficients:

$$F_{\mathrm{f},i}^{\mathrm{tang}} \approx -\alpha \frac{u_{i,\mathrm{bnd}} - u_{i,P}}{\Delta s} S_{\mathrm{f}}, \tag{3.34}$$

with an equivalent amount added to the right-hand side of the matrix equation so that the correct result is obtained at convergence.

The normal force is discretized in a similar manner. The first term is

$$-\mu \frac{\partial u_n}{\partial x_n} S_{\mathrm{f}} = -\mu \left(\alpha \frac{u_{n,\mathrm{bnd}} - u_{n,P}}{\Delta s} + \overline{\nabla u_n} \cdot (\hat{\mathbf{n}} - \alpha \hat{\mathbf{s}}) \right) \tag{3.35}$$

and the second term is treated using cell gradients. The following approximation is used to generate coefficients:

$$F_{\mathrm{f},i}^{\mathrm{norm}} \approx -2\alpha \frac{u_{i,\mathrm{bnd}} - u_{i,P}}{\Delta s} |n_i| S_{\mathrm{f}}. \tag{3.36}$$

At inflow boundaries, both components of velocity are known and both the normal and tangential forces are calculated. At outflow boundaries, both components of velocity are extrapolated, the tangential force is important, and the normal force is neglected in order to enhance convergence. This approximation is not serious because the resulting error is immediately advected out of the domain. At all walls, the normal velocity is zero. At no-slip walls, the normal force vanishes, the tangential velocity is specified, and the tangential force is calculated using Eq. (3.30). At symmetry and slip walls, the tangential force is zero, the tangential velocity is calculated from Eq. (3.30), and the normal force is calculated using Eq. (3.32).

3.4.3 Pressure Term

The pressure force at a face is

$$F_{\mathrm{f},i}^{p} = p_{\mathrm{f}} n_i S_{\mathrm{f}}. \tag{3.37}$$

Since pressure has no preferential direction of action in incompressible flow, p_{f} is most appropriately approximated by a centered discretization:

$$p_{\mathrm{f}} = \frac{1}{2}(p_P + p_Q) + \overline{\nabla p} \cdot \mathbf{r}_{\mathbf{c}}, \tag{3.38}$$

where $\mathbf{r}_{\mathbf{c}}$ is the vector from the midpoint between centroids P and Q to the face midpoint, and $\overline{\nabla p}$ is the average of the adjacent cell pressure gradients. The terms involving cell pressures are made active. The pressure gradient term is lagged. It has

apparently not been used before, but is required to make the discretization linearly exact. It converges more quickly than an alternative linearly-exact discretization,

$$p_f = \frac{1}{2}\left(p_P + \nabla p \cdot \mathbf{r}|_P + p_Q + \nabla p \cdot \mathbf{r}|_Q\right). \tag{3.39}$$

At outflow boundaries, p_f is specified. At inflows, no-slip walls, and slip walls, p_f is extrapolated as follows:

$$p_f = p_P + \nabla p \cdot \mathbf{r}|_P. \tag{3.40}$$

At symmetry boundaries, the normal derivative of pressure is set to zero:

$$\frac{\partial p}{\partial n} = \alpha \frac{p_{\text{bnd}} - p_P}{\Delta s} + \overline{\nabla p} \cdot (\hat{\mathbf{n}} - \alpha \hat{\mathbf{s}}) = 0. \tag{3.41}$$

3.4.4 Mass Flows

The mass flow through a face is

$$J_f = \rho u_{f,n} S_f, \tag{3.42}$$

where $u_{f,n} = \mathbf{u}_f \cdot \hat{\mathbf{n}}$ is the advecting velocity. It is important to discretize the advecting velocity in a special manner in order to avoid pressure-decoupling [50]. Standard colocated approaches follow the lead of Rhie and Chow [59] in introducing a pressure gradient dependence into $u_{f,n}$. In effect, this modification introduces a pressure dissipation term into the continuity equation. The particular form used here is similar to the expression derived for a cell-vertex method in [1]:

$$u_{f,n} = \overline{u}_{f,n} + \alpha d_f \left(\frac{p_Q - p_P}{\Delta s} - \overline{\nabla p} \cdot \hat{\mathbf{s}}\right), \tag{3.43}$$

where $\overline{u}_{f,n}$ is obtained in the same way as p_f in momentum,

$$\overline{u}_{f,n} = \frac{1}{2}(u_{P,n} + u_{Q,n}) + \overline{\nabla u_n} \cdot \mathbf{r}_c, \tag{3.44}$$

and $\overline{\nabla p}$ is the average of the adjacent cell pressure gradients. The pressure dissipation coefficient at a face is

$$d_f = -\frac{1}{2}\left(\frac{\Omega_P}{a_P} + \frac{\Omega_Q}{a_Q}\right), \tag{3.45}$$

where Ω is the cell volume and a is the average central coefficient for the discrete momentum equations. The term involving $(p_Q - p_P)/\Delta s$ and the term involving the average of the cell velocities are made active and all other terms are lagged.

This discretization is similar to existing methods [20, 41]. The new features in this discretization are the correction to $u_{f,n}$ in Eq. (3.44) to make it linearly exact and the choice of $\alpha = \hat{\mathbf{n}} \cdot \hat{\mathbf{s}}$ in the pressure dissipation term.

The mass flows must also be discretized appropriately at boundary faces. At outflow boundaries, pressure is specified and a one-sided expression is used to evaluate $u_{f,n}$:

$$u_{f,n} = \overline{u}_{f,n} + \alpha d_f \left(\frac{p_{bnd} - p_P}{\Delta s} - \overline{\nabla p} \cdot \hat{s} \right). \tag{3.46}$$

At other boundaries, J_f is known from other boundary condition information, but Eq. (3.46) is useful in calculating the pressure at certain boundary faces. Numerical experiments have shown that this type of pressure extrapolation to the boundary is very useful in calculating cell pressure gradients, especially on triangular meshes where the least-squares stencil could otherwise be singular.

As with the scalar conservation equation, good convergence behaviour has been achieved through underrelaxing the gradient calculation and augmenting the coefficients based on the mesh nonorthogonality rather than by time marching. If time marching is used, however, it is important to modify the expression for $u_{f,n}$ so that its value at steady state does not depend on the time step Δt [1, 39]. This is done by defining

$$c = \frac{\rho}{\Delta t} \tag{3.47}$$

and

$$f_f = \frac{d_f}{1 - c d_f}. \tag{3.48}$$

Then the advecting velocity is calculated from

$$u_{f,n} = \overline{u}_{f,n} + \alpha f_f \left(\frac{p_Q - p_P}{\Delta s} - \overline{\nabla p} \cdot \hat{s} \right) - c f_f (u^o_{f,n} - \overline{u}^o_{f,n}), \tag{3.49}$$

where the superscript o denotes values from the previous iteration.

3.4.5 Solution Methodology

The solution methodology for the Navier-Stokes equations is similar to that described for the scalar conservation equation. The major difference is that at each control volume, there are three algebraic equations: one for continuity and two for momentum. Thus at each control volume there is a system of equations having the form

$$[a]_P \{\Phi\}_P + \sum [a]_{nb} \{\Phi_{nb}\} = \{b\}_P, \tag{3.50}$$

where the point coefficient matrix $[a]$ has the form

$$[a] = \begin{bmatrix} a_{pp} & a_{pu} & a_{pv} \\ a_{up} & a_{uu} & a_{uv} \\ a_{vp} & a_{vu} & a_{vv} \end{bmatrix} \tag{3.51}$$

and the point solution vector $\{\Phi\}$ has the form

$$\{\Phi\} = \begin{pmatrix} p \\ u \\ v \end{pmatrix}. \tag{3.52}$$

In these equations p, u, and v represent pressure, u-velocity, and v-velocity, respectively. Non-zeros may exist for all coefficients except a_{uv} and a_{vu}.

The resulting matrix equation for the entire solution field has a block structure. The same algebraic multigrid solver used for the scalar system [58] is used for solving this system. After solving the system of equations and updating the cell pressures and velocities, the boundary face values and mass flows are updated. Then new cell gradients are calculated and a new system of equations is assembled. This process is repeated until the maximum normalized residual for all variables, defined in the same manner as for the scalar conservation equation, falls below a predefined tolerance.

3.5 Validation

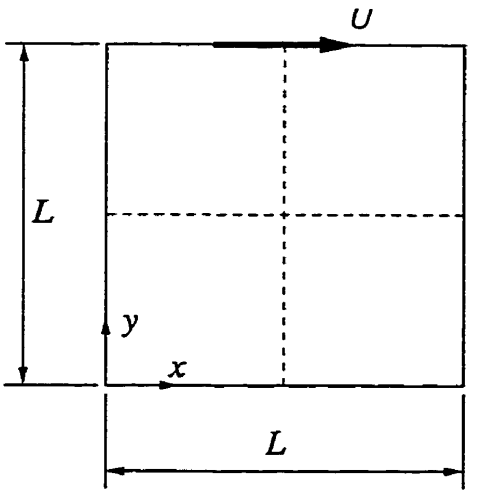
The discretization procedure described in this chapter has been tested for consistency and coding errors using some very simple test cases. These test cases are not described here. Instead, three validation tests for which benchmark solutions are available are presented: shear-driven flow in a square cavity, shear-driven flow in a skewed cavity, and flow over a backward-facing step.

3.5.1 Shear-Driven Square Cavity Flow

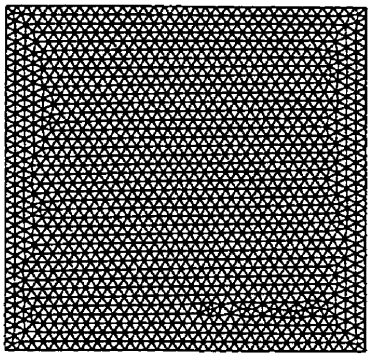
A standard benchmark solution which occurs in the literature is the shear-driven cavity. Consider a square cavity of dimension L, as illustrated in Figure 3.7(a). The cavity lid moves to the right with a velocity U, driving a large vortex in the cavity and possibly some smaller ones in the corners. Benchmark solutions for a variety of Reynolds numbers have been published by Ghia *et al.* [26]. We have considered an intermediate Reynolds number of 1000.

The problem has been solved using two meshes. The coarse mesh, shown in Figure 3.7(b), has 2688 cells. The fine mesh has 10,440 cells. In both cases all normalized residuals have been driven below 10^{-5}, which required 27 iterations for the coarse mesh and 20 iterations for the fine mesh.

The computed solutions may be compared with the benchmark solution by comparing velocity profiles along the two dashed lines shown in Figure 3.7(a). The normalized u-velocity is plotted along the vertical line in Figure 3.8 and the normalized v-velocity is plotted along the horizontal line in Figure 3.9. The velocities are normalized by U and the positions by L. The data points for the computed solutions are obtained from cells whose centroid lies within a threshold distance from the line. The coarse and fine mesh solutions are nearly the same, showing that the solution is essentially mesh-independent, and feature excellent agreement with the benchmark solution.



(a) Geometry of cavity. Velocity profiles will be plotted along the dashed lines.



(b) Coarse mesh.

Figure 3.7: Shear-driven square cavity test case.

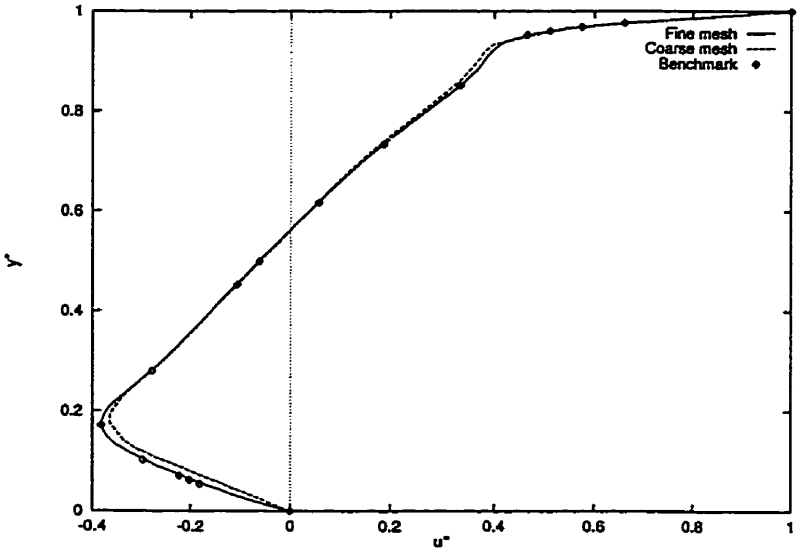


Figure 3.8: Normalized u-velocity along a vertical line through the centre of the cavity for the shear-driven square cavity test case.

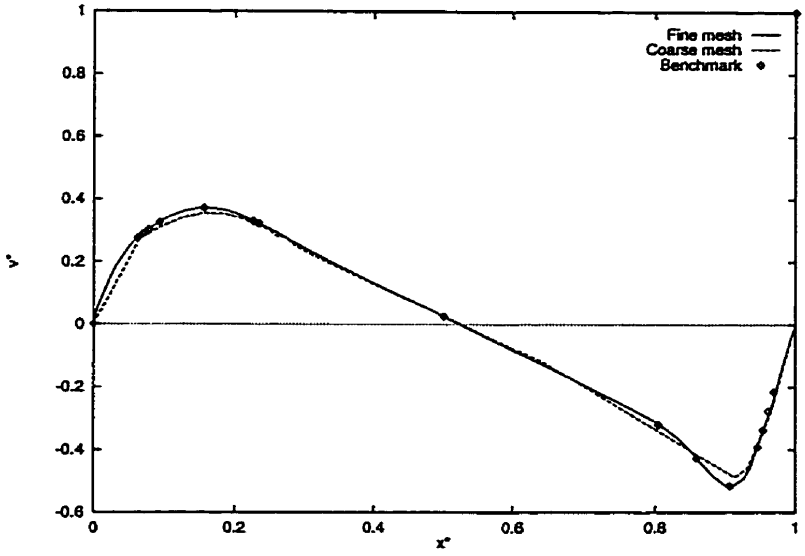


Figure 3.9: Normalized v-velocity along a horizontal line through the centre of the cavity for the shear-driven square cavity test case.

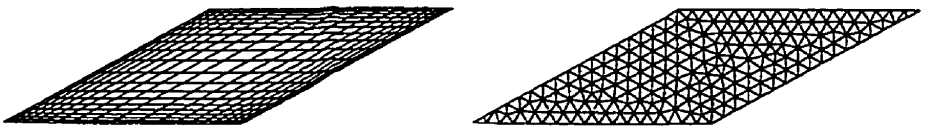
3.5.2 Shear-Driven Skewed Cavity Flow

Solutions have also been obtained to a skewed shear-driven cavity problem. The problem definition is identical to the square cavity flow except that the cavity is skewed to the right by 60°. Benchmark solutions to this case have been published by Demirdžić *et al.* [17]. Mathur and Murthy [41] performed a similar study also using an unstructured finite-volume solver. We have obtained solutions using both triangular and quadrilateral meshes, of which the coarsest are shown in Figure 3.10. Four quadrilateral meshes were used, having sizes of 19×19, 39×39, 79×79, and 159×159, which respectively required 42, 59, 46, and 36 iterations to reduce all normalized residuals below 10^{-5}. Four triangular meshes were also used, having 384, 1512, 6020, and 23,700 cells, which correspond roughly to the quadrilateral meshes and which respectively required 33, 48, 42, and 37 iterations to reach convergence.

The normalized velocities for the solutions are plotted along a skewed vertical line through the centre of the cavity in Figure 3.11 and along a horizontal line through the centre of the cavity in Figure 3.12.

3.5.3 Flow over a Backward-Facing Step

The final test case involves laminar flow over a backward-facing step. The geometry, shown in Figure 3.13, is identical to that given in the benchmark paper by Gartling [25]. The domain length is $L = 30$, the step height is $b = 0.5$, and the origin is placed



(a) Quadrilateral mesh (19×19 cells). (b) Triangular mesh (384 cells).

Figure 3.10: Coarse meshes used for shear-driven skewed cavity test case.

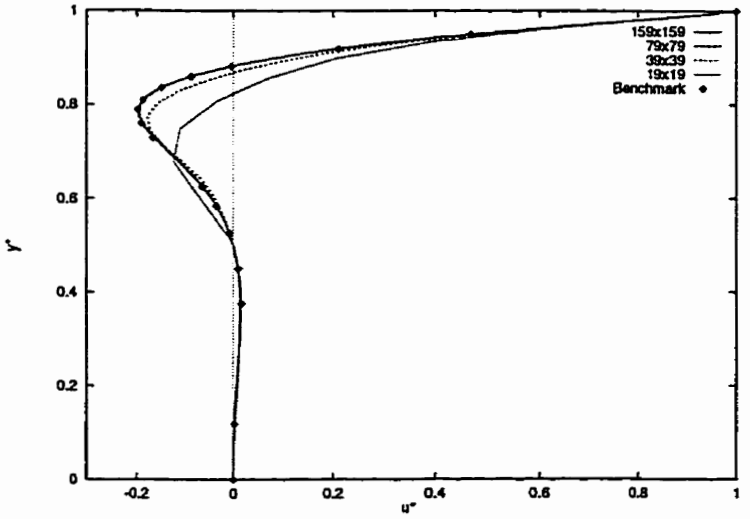
at the top of the step. Fully developed flow is assumed at the inflow boundary:

$$u(y) = 24y(0.5 - y), \tag{3.53}$$

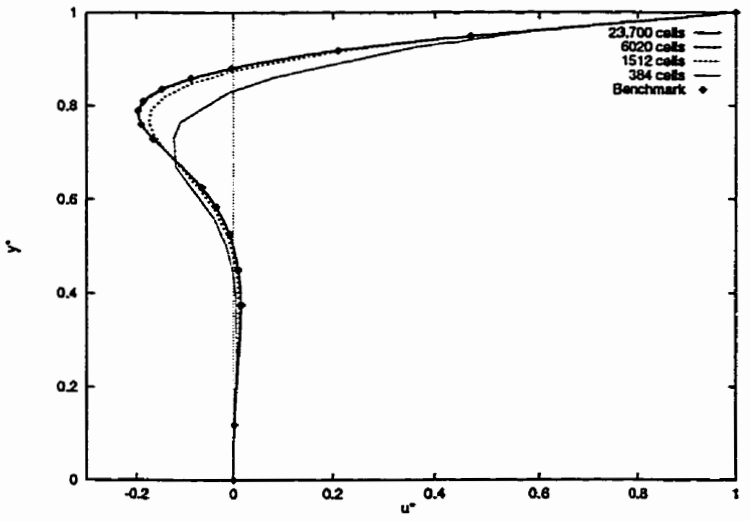
which gives an average inflow velocity of $\overline{u} = 1$. As in the benchmark paper, a Reynolds number of $\mathrm{Re} = \rho\overline{u}(2b)/\mu = 800$ is used. Although this flow is not experimentally realizable (three-dimensional effects appear at about $\mathrm{Re} = 400$ [6]), it is still a popular test case because of its difficulty [28]. The difficulty stems from the separation zones along the lower and upper walls. The two primary zones are indicated on the figure: the flow behind the step reattaches at $x = L_l$, while the first bubble on the upper surface detaches at $x = L_{u1}$ and reattaches at $x = L_{u2}$. The solution throughout the domain is sensitive to the positions of the bubbles, leading to slow convergence and a strong sensitivity to discretization error.

This problem has been solved on coarse, medium, and fine triangular meshes, respectively having 2241 cells, 8947 cells, and 37,513 cells. These meshes have nonuniform densities, with one grading factor for $0 < x < 10$ and another for $10 < x < 30$. The gradings are such that the cells at $x = 10$ are 1.5 times larger than at $x = 0$ and the cells at $x = 30$ are four times larger than at $x = 0$. The problem has also been solved on coarse, medium, and fine quadrilateral meshes, having 1600, 6400, and 15,600 cells. The quadrilateral meshes have uniform spacings in the $y-$ direction. In the $x-$direction, 75% of the nodes are distributed uniformly in the region $0 < x < 15$ and the remainder are distributed nonuniformly in the remainder. The cell aspect ratio in the upstream portion of the duct is five. In all cases, convergence was relatively slow – typically 100-150 iterations were required to reduce the normalized residuals below 10^{-5}.

The calculated detachment and reattachment points are tabulated in Table 3.2, together with the benchmark values. They are determined as the points where the wall shear stress changes sign. The results indicate that a mesh-independent result has not yet been reached with the finest meshes, but the solutions feature the correct trend toward the benchmark solution. The results also suggest that the quadrilateral meshes give somewhat better results than the triangular meshes. This appears to be related to the aspect ratios of the quadrilateral cells: for a given mesh size, they resolved the flow features in the transverse direction better than the triangular cells.

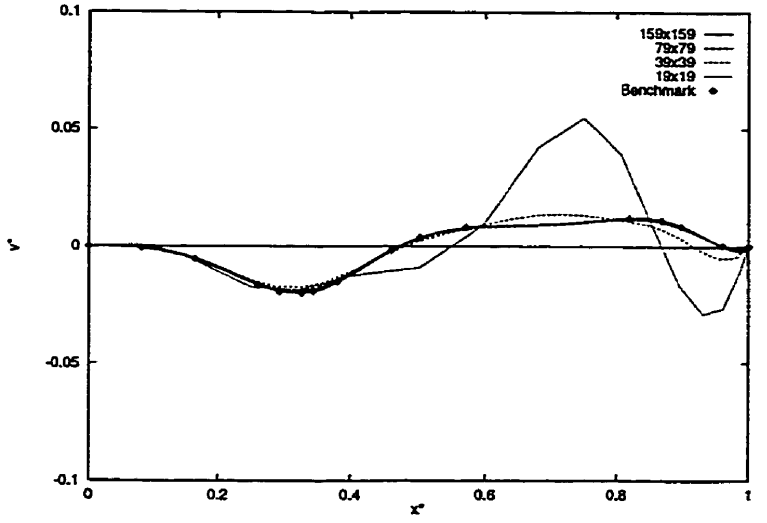


(a) Quadrilateral meshes.

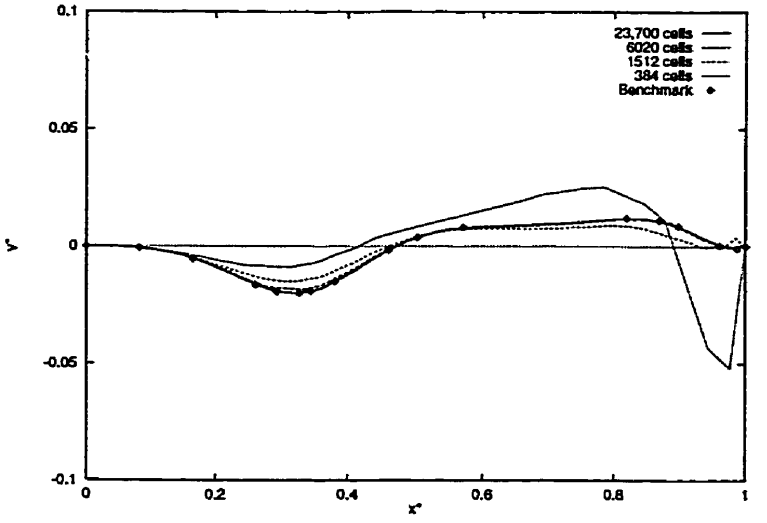


(b) Triangular meshes.

Figure 3.11: Normalized u-velocity along a skewed vertical line through the centre of the cavity for the shear-driven skewed cavity test case.



(a) Quadrilateral meshes.



(b) Triangular meshes.

Figure 3.12: Normalized v-velocity along a horizontal line through the centre of the cavity for the shear-driven skewed cavity test case.

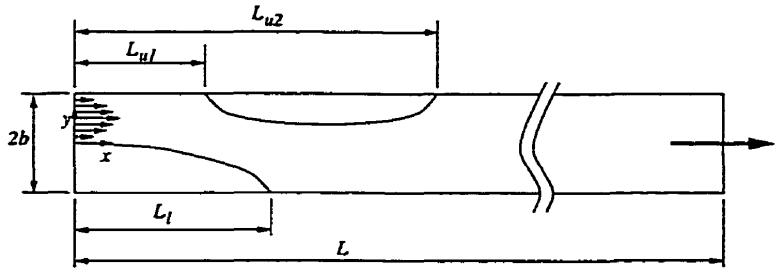


Figure 3.13: Geometry of backward step test case, with $b = 1$ and $L = 30$.

Table 3.2: Separation points for the backward step test case.

Mesh	L_l	L_{u1}	L_{u2}
Coarse triangular (2241 cells)	3.85	2.50	6.75
Medium triangular (8947 cells)	5.64	4.45	9.98
Fine triangular (37,513 cells)	6.00	4.76	10.40
Coarse quad (80 × 20 cells)	5.30	4.10	10.10
Medium quad (160 × 40 cells)	5.86	4.62	10.42
Fine quad (320 × 80 cells)	6.01	4.78	10.46
Benchmark	6.10	4.85	10.48

Chapter 4

Unsteady Flows I: Space-Time Mesh Generation

4.1 Introduction

In the previous chapter a finite volume methodology for solving steady-state conservation equations was presented. By extending this methodology to the time dimension, the next two chapters will develop the IST finite volume method for unsteady flows. This chapter is devoted to space-time meshing for moving boundary problems, and the next to the IST discretization procedure.

Just as conventional discretization methods fill the spatial domain with a spatial mesh, so also space-time methods fill the space-time domain with a space-time mesh. Thus the dimension of the mesh is increased by one: for two-dimensional problems, the mesh has two spatial dimensions plus a time dimension.

Space-time methods usually split the space-time domain into *time slabs*, in order to decouple the solution at a particular time from those at later times. There have been various approaches to meshing individual time slabs. In many of them, the space-time elements follow the flow, leading to a Lagrangian method [29, 30]. These methods typically require periodic global remeshes and solution projections in order to avoid mesh distortion and tangling. Another approach has been to tessellate every time slab with simplex space-time elements [24], which seems expensive for higher-dimensional problems.

The space-time meshing algorithm presented here avoids global remeshing entirely, instead using only local mesh modifications near the moving boundaries. The algorithm will first be outlined for one-dimensional problems, followed by a more involved explanation for two-dimensional problems.

4.2 One-dimensional problems

This section describes the space-time meshing algorithm for one-dimensional moving boundary problems. The space-time domain is first subdivided into time slabs, as shown in Figure 4.1, and each time slab is filled with nonoverlapping cells. The time slabs are bounded by *time planes* t^n and t^{n+1}. Each time plane is covered with a spatial mesh, which forms a boundary for the space-time cells above and below the time plane.

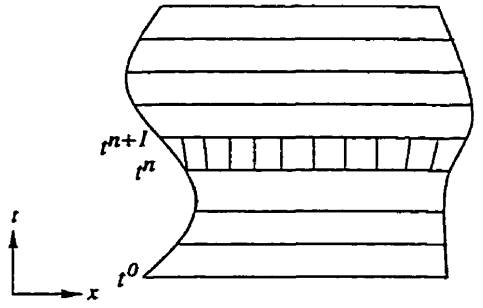


Figure 4.1: Division of space-time domain into time slabs.

In principle, the mesh within a time slab may be quite general. For simplicity, all cells are required to span the distance between times t^n and t^{n+1}. The cells are two-dimensional and bounded by two different types of faces: *time faces*, which lie on a time plane, and *space-time faces*, which span the distance between time planes.

When tessellating a time slab, the existing spatial mesh on the lower time plane t^n is used to simultaneously generate the spatial mesh on t^{n+1} and the space-time mesh between the time planes. For the first time slab, the spatial mesh at time t^0 must be given. In one dimension, this initial mesh consists simply of a predefined number of vertices evenly spaced over the length of the spatial domain, as well as the time faces of length $\overline{L}$ which which join the vertices.

The time slab meshing algorithm uses a four step-process, as illustrated in Figure 4.2. In step (a), the lower spatial mesh is extruded in time to generate a space-time face from each vertex and a quadrilateral cell from each time face. In step (b), the boundary vertices on the new time plane are moved to their new prescribed locations. This step may produce time faces which are too long, too short, or even tangled. Step (c) therefore modifies the mesh topology next to the boundary by adding or removing vertices. A vertex is added next to the boundary if the time face length is too long,

$$L > \beta_{\max}\overline{L}, \tag{4.1}$$

and removed if it is too short,

$$L < \beta_{\min}\overline{L}. \tag{4.2}$$

Adding a vertex generates a triangular cell having a vertex on the lower time plane and an edge on the upper time plane (right side of figure). Removing a vertex generates a triangular cell having an edge on the lower time plane and a vertex on the upper time plane (left side). $\beta_{\max}$ and $\beta_{\min}$ are defined parameters. Finally, in step (d) the vertex locations on the new time plane are smoothed.

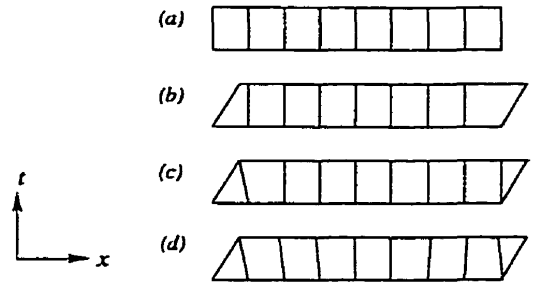


Figure 4.2: Generating a mesh for a time slab: (*a*) extrude in time; (*b*) move boundaries; (*c*) add/remove vertices; (*d*) smooth.

The algorithm as described places a restriction on the allowable time step: no more than one vertex may be added or removed next to a boundary in a time step. However, for one-dimensional problems, it is not hard to modify the algorithm to allow additional vertices to be added or removed. An example of a space-time mesh generated with this algorithm is shown in Figure 4.3, where a domain has its left boundary fixed at $x = 0$ and its right boundary varies according to $x = .5\sin(\pi t)$. The space-time mesh is shown for $0 < t < 3$ using time steps of $\Delta t = 0.1$ and ten cells on the initial time plane.

4.3 Two-dimensional problems

4.3.1 Overview

Space-time meshing for two-dimensional problems follows the same general approach as with one dimension, but the steps are more involved. As in the one-dimensional case, time slabs are used, with space-time cells spanning the distance between the time planes. Generating the space-time mesh requires tracking two types of topologies: the two-dimensional spatial mesh on each time plane and the space-time mesh which joins the time planes. A good space-time mesh requires having quality meshes for both of these. The initial spatial mesh for the t^0 time plane is generated by EasyMesh [47]. For simplicity, only triangular time faces are considered.

The first step in the meshing algorithm involves extruding the spatial mesh which lies on the lower time plane. Edges generate quadrilateral space-time faces and time faces generate triangular prisms. Next, vertices which lie on moving

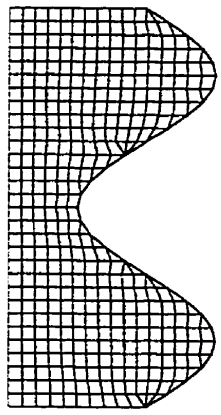


Figure 4.3: Space-time mesh for a one-dimensional problem having a fixed left boundary and oscillating right boundary.

boundaries are moved to their new locations. Third, modifications are made to the mesh topology near the boundary to maintain mesh quality. This step is the most involved, and is discussed in further detail below. Finally, one or two layers of vertices next to the boundary are smoothed.

Modifying the mesh topology introduces new types of space-time faces and cells. The faces have one of three possible shapes, shown in Figure 4.4. The unmodified topology is a quadrilateral, having an edge on both the lower and upper time planes. Two triangular topologies may also be encountered. The face types are labelled according to how they appear on the lower and upper time planes; *e.g.*, a VERT-EDGE face has a vertex on the lower time plane and an edge on the upper.

The cells have one of six possible topologies, shown in Figure 4.5. The unmodified topology is a triangular prism, having a triangular time face on both the lower and upper time planes. Two pyramid shapes, having a triangular time face on one time plane and an edge on the other, may also be encountered. Three tetrahedra are also possible: two have a triangular time face on one time plane and a vertex on the other, and the third has an edge on both time planes.

Topology changes near the moving boundary are grouped into three categories: adding and removing vertices adjacent to the moving boundary, adding and removing vertices on the boundary itself, and diagonal swapping. Permitting all of these on any given time slab leads to a host of interactions between the different operations which have to be explicitly considered. The number of interactions is reduced by permitting only one category on a particular time slab. Each type of modification is now considered.

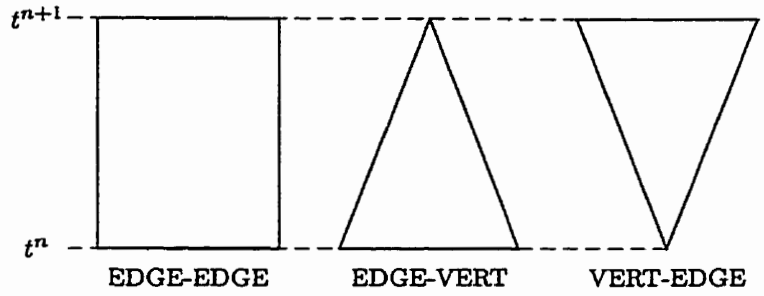
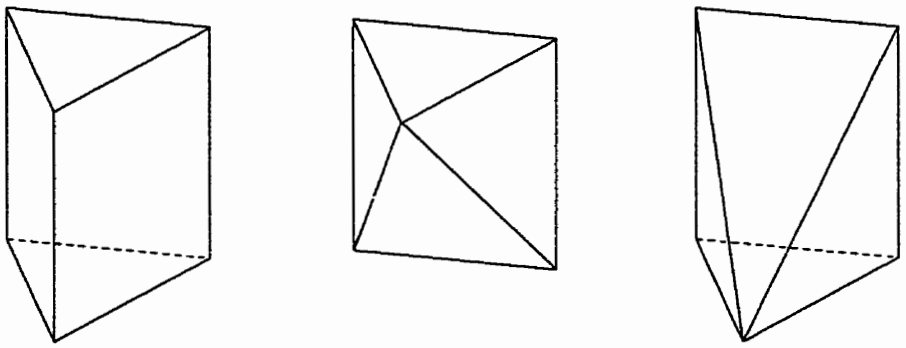
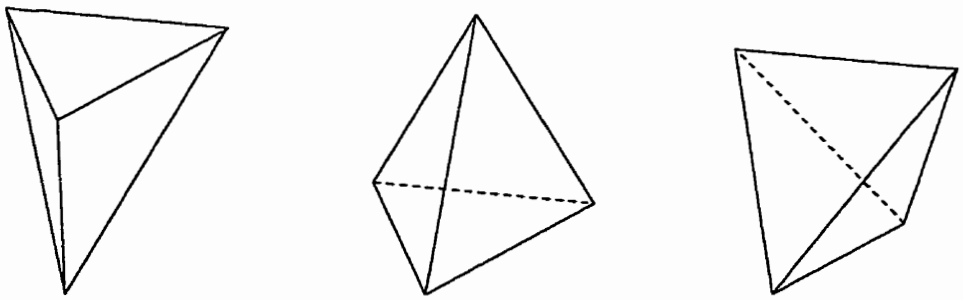


Figure 4.4: Space-time face topologies for two-dimensional problems.



(a) TRI-TRI (b) EDGE-TRI (c) TRI-EDGE



(d) VERT-TRI (e) TRI-VERT (f) EDGE-EDGE

Figure 4.5: Cell topologies for two-dimensional problems.

4.3.2 Adding and Removing Vertices Adjacent to the Boundary

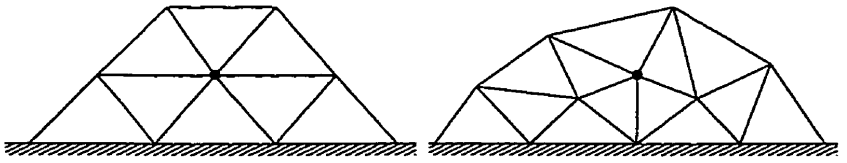
Deciding where to add and remove vertices adjacent to the moving boundary in two dimensions follows the criteria for one-dimensional problems. Define $\overline{L}$ to be the average edge length on the initial time plane, and L to be the distance of a vertex to a moving boundary. A vertex must be added if the distance is too long,

$$L > \beta_{\max}\overline{L}, \tag{4.3}$$

and removed if it is too short,

$$L < \beta_{\min}\overline{L}. \tag{4.4}$$

In order to maintain mesh quality when these operations are carried out, it is essential that the spatial mesh on the new time plane retain its original structure. Figure 4.6 distinguishes between two types of vertices on a time plane. The *face vertex* is connected to the boundary by a face, whereas the *edge vertex* is connected by a single edge. Removing these vertices results in different topological changes, as does adding new vertices when they are too far from the boundary.

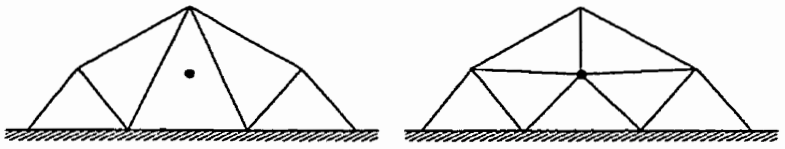


(a) Face vertex (b) Edge vertex

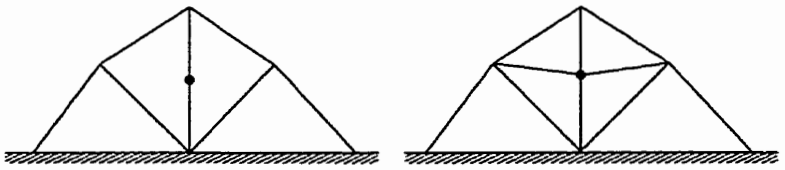
Figure 4.6: Types of vertices which may be connected to a moving boundary. The mesh lies on a time plane and the boundary is indicated by the hash marks.

Adding a vertex

When a vertex is added, the topologies of both the spatial mesh on the new time plane and the space-time mesh are modified. The change to the new time plane is relatively straightforward, as illustrated in Figure 4.7. The resulting topology of the time slab, shown in Figure 4.8, is more complex. When adding an edge vertex, the topological changes can be separated into a left and a right section, each consisting of a TRI-VERT cell and two EDGE-TRI cells. When adding a face vertex, the same two sections appear, but now sandwich a central piece consisting of a TRI-VERT cell and an EDGE-TRI cell.

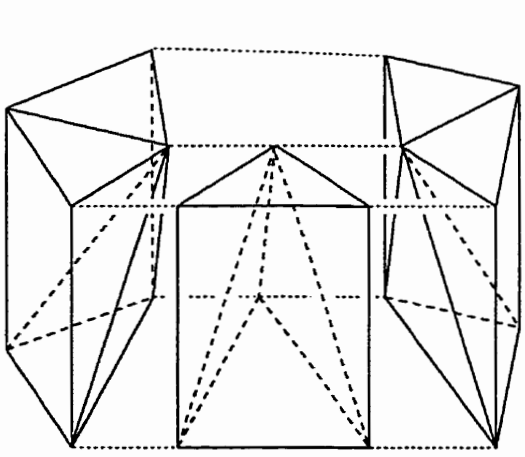


(a) Adding a face vertex

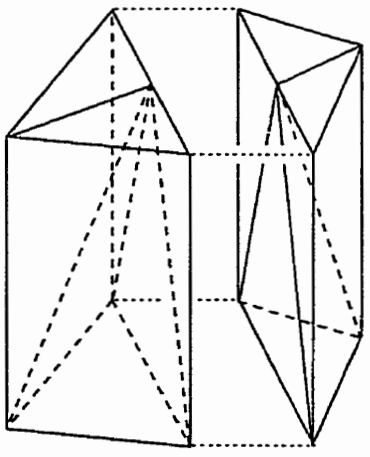


(b) Adding an edge vertex

Figure 4.7: Topological changes to time plane when a vertex is added.



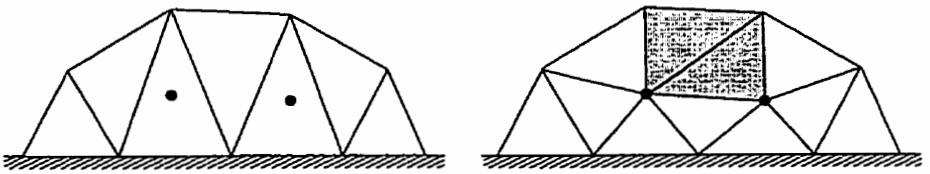
(a) Adding a face vertex



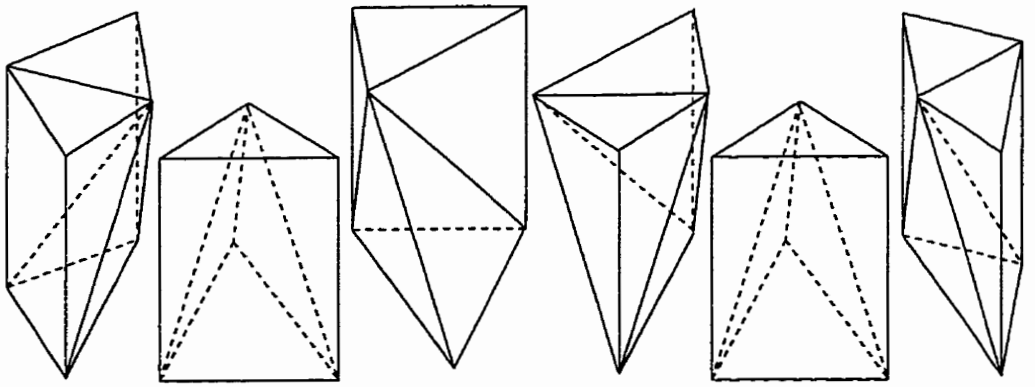
(b) Adding an edge vertex

Figure 4.8: Topological changes to time slab when a vertex is added. The dotted lines illustrate how the pieces glue together.

These operations have assumed that only a single vertex is added in isolation. When adjacent vertices are added, the interaction between them must also be considered. For example, Figure 4.9 illustrates vertices added to adjacent time faces. The new time plane features a quadrilateral face which must be triangulated. The resulting mesh has a squarish appearance. The space-time mesh for this operation has been split into six sections. The left and right sections of Figure 4.8 reappear, and the central section reappears twice. In the very center, two additional pieces appear, which arise from the interaction between the two vertices. The centre-left piece is composed of a TRI-VERT and an EDGE-TRI cell (which also form two-thirds of the right section of Figure 4.8(a)). The centre-right piece has an EDGE-EDGE cell sandwiched by two VERT-TRI cells.



(a) Changes to time plane. The shaded quadrilateral illustrates how the mesh takes on a squarish look.

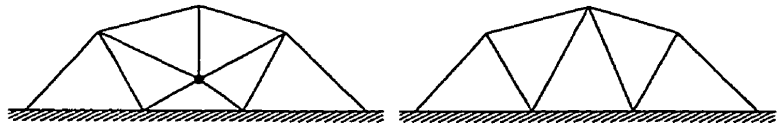


(b) Structure of time slab

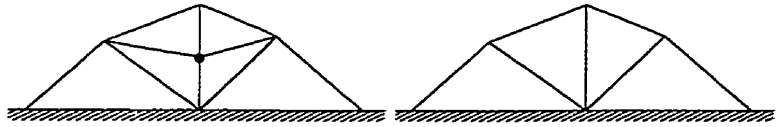
Figure 4.9: Topological changes when two vertices are added.

Removing a vertex

Removing a vertex is roughly the reverse procedure of adding a vertex. The topological changes to the time plane are shown in Figure 4.10. The operation consists of removing all edges on the time plane which touch the vertex and retriangulating the resulting polygon. The examples shown are unambiguous, but in other cases, where the vertex to be removed has more neighbours, the retriangulation must be done in such a way that the mesh quality does not deteriorate.



(a) Removing a face vertex



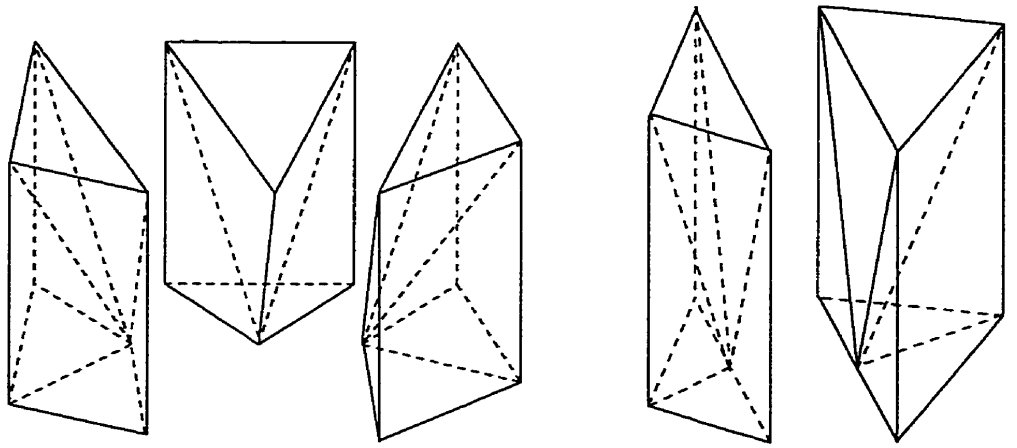
(b) Removing an edge vertex

Figure 4.10: Topological changes to time plane when a vertex is removed.

The topological effects of removing a vertex are illustrated in Figure 4.11. The space-time cells are inverted versions of those which appear when a vertex is added (Figure 4.8). For every time face which shares the vertex on the old time plane, a TRI-EDGE cell appears which causes the time face to collapse into the edge opposite the vertex. In addition, for each new time face which appears on the new time plane, a VERT-TRI cell appears. When the vertex being removed has more neighbours than the examples shown here, the central section becomes more complex. Removing adjacent vertices is also more involved than removing a single vertex; however, the situation is quite similar to adding adjacent vertices.

4.3.3 Adding and Removing Vertices on the Boundary

Adding and removing vertices on the moving boundary itself is also based on distances. If a boundary edge is too long a vertex is added at its midpoint. If the boundary edges adjacent to a vertex are too short, the vertex is removed. The resulting topological changes to the time plane and time slab are one-sided relatives of the corresponding changes when edge vertices are added or removed adjacent to the boundary (half of Figures 4.8(b) and 4.11(b)).



(a) Removing a face vertex

(b) Removing an edge vertex

Figure 4.11: Topological changes to time slab when a vertex is removed.

4.3.4 Diagonal swapping

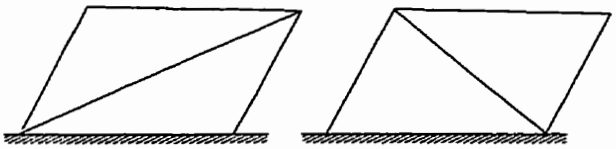
In some cases diagonal swapping is also required to maintain mesh quality. This process is illustrated in Figure 4.12. The space-time cell, having two time faces on each of the lower and upper time planes, is more complex than the primitive shapes presented in Figure 4.5. The cell could be decomposed into tetrahedra if only a single swap is performed. If two adjacent diagonals are swapped, however, a decomposition is impossible. For this reason, the cell is kept as a composite.

4.3.5 Geometry Calculations

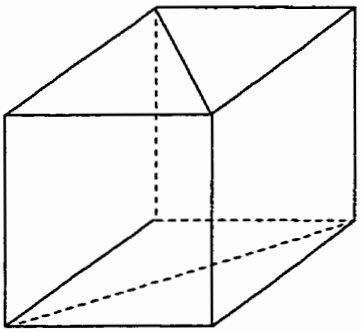
The cell-centered flow solver requires the calculation of several geometrical entities. Areas and centroids are required for time faces; areas, centroids, and normal vectors for space-time faces; and volumes and centroids for cells.

The faces and cells through most of the time slab are extruded from the lower time plane, and their geometries are easily calculated. Only near moving boundaries, where vertices have been added, removed, or moved, are more extensive calculations required. These calculations are explained below.

The geometries of time, EDGE-VERT, and VERT-EDGE faces are straightforward, as they are triangular. The geometries of EDGE-EDGE faces are calculated by introducing an auxiliary point as the average of the four bounding vertices. The auxiliary point is used to decompose the face into four triangles, whose areas are summed to obtain the face area and whose centroids are weighted by area and averaged to obtain the face centroid.



(a) Changes to time plane



(b) Structure of space-time cell

Figure 4.12: Swapping a diagonal.

A similar process is used for the cell calculations. The geometries of VERT-TRI, TRI-VERT, and EDGE-EDGE cells are easily calculated, as they are tetrahedra. For the remaining cell types, an auxiliary point is used to decompose the cell into tetrahedra (14 for TRI-TRI cells and 8 for EDGE-TRI and TRI-EDGE cells). Cell volumes are obtained by summing the tetrahedral volumes, and cell centroids are obtained from the volume-weighted average of the tetrahedral centroids.

It should be noted that, although a Green-Gauss theorem could be used to calculate cell volumes, it is still necessary to decompose the cell into tetrahedra for the centroid calculation. This is because the Green-Gauss surface integral for the centroid position has quadratic terms, which cannot be solved exactly using a single-point quadrature.

4.3.6 Test Cases

To illustrate how the algorithm works in practice, consider a square cavity having a dimension L. The right boundary of the square moves back and forth in a sinusoidal manner. The amplitude of the oscillation is $0.75L$ and the period of oscillation is T. The initial geometry is meshed using 670 triangles and 80 time slabs are used per period. Figure 4.13 shows the resulting mesh on time planes $0 < t < 1.125T$ in intervals of $0.125T$. The results clearly show that good mesh quality is maintained throughout the oscillation.

Other cases involving more complex boundary motion will be presented later in the thesis.

4.3.7 Discussion

The results for this test case and others indicate that the method works well on a variety of problems. There are some limitations, however. First, unlike the one-dimensional case, there is a CFL-type restriction (based on the boundary vertex speed) on the time slab thickness. This is due to the excessive complexity which would be required to allow multiple layers of vertices to be added or removed in a time slab. A further limitation is the assumption of an isotropic spatial mesh on the time planes; if necessary, it should be possible to extend the algorithm to other cases as well. In addition to these limitations, the algorithm is relatively complex to code: many special cases, particularly near corners, must be considered.

Despite these limitations, the method has some significant advantages. The main advantage is its computational efficiency. By using a four-step procedure of extruding the mesh on the old time plane, moving the boundary vertices, adding and removing vertices near the boundary, and smoothing on the new time plane, mesh quality is maintained while avoiding global remeshes.

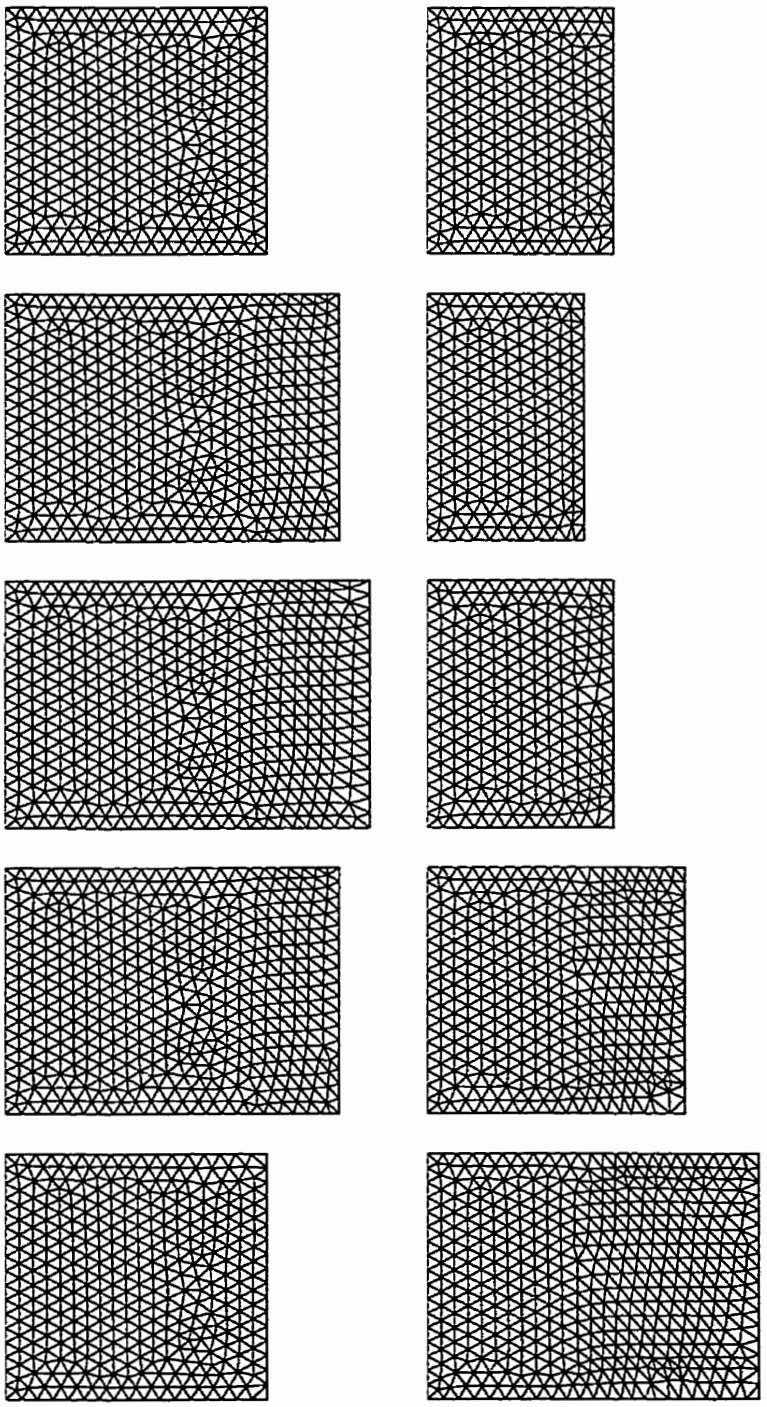


Figure 4.13: Spatial mesh for cavity test case on time planes $0 < t < 1.125T$ in intervals of $0.125T$. The sequence runs by column.

Chapter 5

Unsteady Flows II: The Integrated Space-Time Finite Volume Method

In this chapter, the IST discretization will be described. It will begin by rewriting the governing equations in a form which unifies the terms involving space and time. Next, the space-time control volumes will be chosen. The discretization procedures for the unsteady scalar conservation equation and the Navier-Stokes equations will then be described, and the chapter will conclude with a validation test case.

5.1 Mathematical Formulation

5.1.1 Differential Forms

Underlying the IST concept is the premise that, just as conservation principles apply to both space and time, so should the discrete finite volume principle. In order to translate this premise into a numerical algorithm, it is helpful to rewrite the governing equations in a form which unifies space and time. This will be done first for the scalar equation, presented earlier as Eq. (2.10):

$$\frac{\partial(\rho\phi)}{\partial t} + \frac{\partial(\rho u_i \phi)}{\partial x_i} + \frac{\partial q_i}{\partial x_i} = \dot{S}. \tag{5.1}$$

The IST formulation requires the use of space-time vectors, which are distinguished from purely spatial vectors by a prime. As space-time vectors have an increased span, the subscript t is defined to be one more than the number of spatial dimensions. Thus the time coordinate is x'_t.

By defining a "time velocity" $u'_t = 1$, the transient and advection terms of

Eq. (5.1) are combined as follows:

$$\frac{\partial(\rho\phi)}{\partial t} + \frac{\partial(\rho u_i \phi)}{\partial x_i} = \frac{\partial(\rho u_i' \phi)}{\partial x_i'}. \tag{5.2}$$

We must also ensure that the other terms maintain the special nature of time. For instance, diffusion occurs in space but not in time. The anisotropy of the space-time continuum is reflected by the definition of a space-time metric tensor γ_{ij}':

$$\gamma_{ij}' = \begin{cases} 1 & \text{if } i = j \text{ and } i, j < t, \\ 0 & \text{otherwise.} \end{cases} \tag{5.3}$$

For instance, if there are two spatial dimensions, then γ_{ij}' expands to

$$\gamma_{ij}' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{5.4}$$

Then Eq. (5.1) may be rewritten as

$$\frac{\partial(\rho u_i' \phi)}{\partial x_i'} + \frac{\partial q_i'}{\partial x_i'} = \dot{S}, \tag{5.5}$$

where

$$q_i' = -\gamma_{ji}' \Gamma \frac{\partial \phi}{\partial x_j'}. \tag{5.6}$$

In the same manner, the continuity equation (Eq. (2.1)) may be written in IST form as

$$\frac{\partial u_i'}{\partial x_i'} = 0, \tag{5.7}$$

and the momentum equation (Eq. (2.5)) as

$$\frac{\partial(\rho u_j' u_i')}{\partial x_j'} = -\gamma_{ji}' \frac{\partial p}{\partial x_j'} + \frac{\partial \tau_{ji}'}{\partial x_j'}, \tag{5.8}$$

where

$$\tau_{ji}' = \gamma_{ki}' \gamma_{lj}' \mu \left(\frac{\partial u_k'}{\partial x_l'} + \frac{\partial u_l'}{\partial x_k'} \right). \tag{5.9}$$

In the momentum equation, the free index i varies over the spatial dimensions. It is fascinating to observe, however, that if the time component is considered, the continuity equation is recovered. It is therefore possible to express the mass and momentum system as a single "space-time momentum equation". For our purposes, however, there does not seem to be any advantage in doing so, and the continuity and momentum equations will be considered separately.

5.1.2 Integral Forms

When applying the finite volume principle to steady problems, the differential equations are integrated over a control volume Ω. In the same manner, the IST finite volume principle requires the integration of the differential equations over a space-time control volume Ω'. Doing so for the scalar equation, Eq. (5.5), gives

$$\int_{\Omega'} \frac{\partial(\rho u_i' \phi)}{\partial x_i'} \, d\Omega' + \int_{\Omega'} \frac{\partial q_i'}{\partial x_i'} \, d\Omega' = \int_{\Omega'} \dot{S} \, d\Omega'. \tag{5.10}$$

Using Gauss' divergence theorem, the volume integrals are converted to surface integrals:

$$\int_{S'} \rho u_i' n_i' \phi \, dS' + \int_{S'} q_i' n_i' \, dS' = \int_{\Omega'} \dot{S} \, d\Omega', \tag{5.11}$$

where S' is the space-time surface bounding Ω' and n_i' is the outward-directed space-time normal to S'.

The only volume integral in this equation arises from the volumetric source term. All other terms, including the transient term now combined with the advection term, involve surface integrals. This occurs because all corresponding terms in Eq. (5.5) are in divergence form, illustrating that conservation principles apply both to space and time dimensions. So an interpretation of Eq. (5.11) is: the net outflow of ϕ from Ω' is balanced by internal generation. In contrast, the conventional form (Eq. (2.14)) is interpreted as: the rate of change of ϕ in Ω is balanced by the net inflow rate through the surface and the internal generation rate. This conceptual simplification also yields algorithmic simplifications, for the Leibnitz rule and GCL do not need to be considered. New complexity is introduced, however, from the time dimension of the control volume.

Another advantage of the IST formulation follows from the combination of transient and advection terms. As a result, the same discretization may be used for both. This will be demonstrated later.

Following the same procedure, the integral form of the continuity equation is

$$\int_{S'} \rho u_i' n_i' \, dS' = 0 \tag{5.12}$$

and of the momentum equation is

$$\int_{S'} \rho u_i' u_j' n_j' \, dS' - \int_{S'} \gamma_{ji}' p n_j' \, dS' - \int_{S'} \tau_{ji}' n_j' \, dS' = 0. \tag{5.13}$$

5.2 Choice of Control Volumes

In Section 3.2, different issues affecting the choice of control volumes for steady flows were discussed. With IST, the same choice between cell-centered and vertex-centered methods is faced. One additional issue must also be addressed: the need for a time-marching algorithm. Time-marching is possible provided there are time

planes across which the solution field is discontinuous. This is ensured if time slabs are used together with a cell-centered method, so long as the fluxes through time faces are discretized using "upwind" (from the previous time slab) information. Vertex-centered methods, on the other hand, explicitly require the solution to be made discontinuous using the discontinuous Galerkin method [35]. A cell-centered method is chosen in this study.

5.3 The Scalar Conservation Equation

In this section, the IST algorithm for the scalar conservation equation is described. The integral form of the equation was given earlier as Eq. (5.11). Dropping the source term, we obtain

$$\int_{S'} \rho u_i' n_i' \phi \, dS' + \int_{S'} q_i' n_i' \, dS' = 0, \tag{5.14}$$

or, in vector notation,

$$\int_S \rho \phi \mathbf{u}' \cdot \hat{\mathbf{n}}' \, dS + \int_S \mathbf{q}' \cdot \hat{\mathbf{n}}' \, dS = 0. \tag{5.15}$$

The surface integrals are approximated using the midpoint rule. The resulting discrete form of the equation for each control volume is

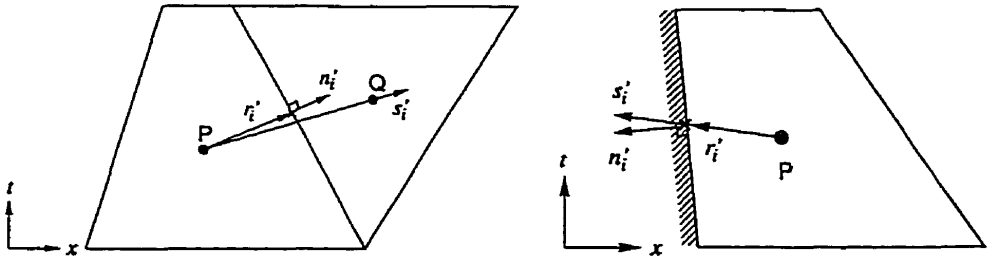
$$\sum_{\mathrm{f}} (F_{\mathrm{f}}^a + F_{\mathrm{f}}^d) = 0, \tag{5.16}$$

where, as in the steady algorithm, F_{f}^a and F_{f}^d respectively represent the numerical advective and diffusive transport through each face. Transient effects are included in the advection term. Typical faces which bound space-time control volumes for one spatial dimension are illustrated in Figure 5.1. Important vectors shown on the diagram are n_i' (or $\hat{\mathbf{n}}'$), s_i' (or $\hat{\mathbf{s}}'$), and r_i' (or $\mathbf{r}'$), which are space-time extensions of the vectors shown in Figure 3.1 for the steady algorithm.

Before deriving expressions for the numerical fluxes in terms of the cell centroid values of ϕ, the method for calculating space-time cell gradients of ϕ will be presented.

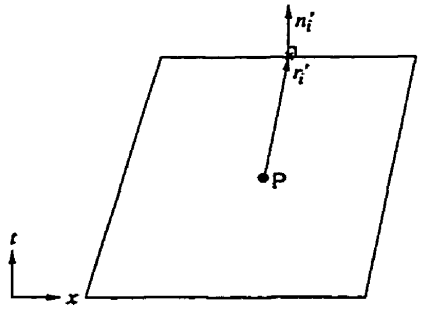
5.3.1 Cell Gradient Vectors

A procedure for calculating cell gradient vectors for steady problems was described in Section 3.3.1. In space-time, the gradient vector also includes a time component and is therefore denoted by $\nabla'\phi$. The extension of the least-squares method to space-time is straightforward. Consider all space-time neighbours of cell P, as illustrated for one spatial dimension in Figure 5.2(a). Note that the space-time neighbours from the next time slab cannot be included, for they are not yet known;



(a) An internal space-time face

(b) A boundary space-time face



(c) A time face

Figure 5.1: Typical control volume faces in space-time.

this leads to a one-sided stencil in time. The components of the gradient vector are determined from the matrix equation

$$\begin{bmatrix} \sum w_j \Delta x_j \Delta x_j & \sum w_j \Delta x_j \Delta t_j \\ \sum w_j \Delta x_j \Delta t_j & \sum w_j \Delta t_j \Delta t_j \end{bmatrix} \begin{pmatrix} \phi_x \\ \phi_t \end{pmatrix} = \begin{pmatrix} \sum w_j \Delta x_j \Delta \phi_j \\ \sum w_j \Delta t_j \Delta \phi_j \end{pmatrix}. \tag{5.17}$$

For some degenerate space-time cells, such as the triangular cell shown in Figure 5.2(b), this matrix may be poorly conditioned and the stencil must be modified to include cells from the previous time slab.

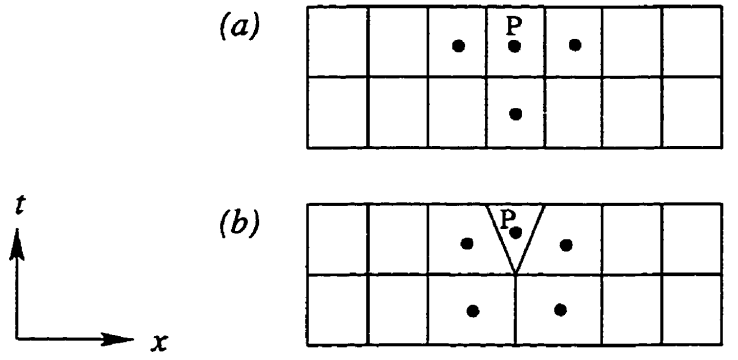


Figure 5.2: Least-squares stencil used in calculating cell gradient vectors.

For two spatial dimensions, the cell gradient vector includes three components, and is calculated from the following matrix equation:

$$\begin{bmatrix} \sum w_j \Delta x_j \Delta x_j & \sum w_j \Delta x_j \Delta y_j & \sum w_j \Delta x_j \Delta t_j \\ \sum w_j \Delta x_j \Delta y_j & \sum w_j \Delta y_j \Delta y_j & \sum w_j \Delta y_j \Delta t_j \\ \sum w_j \Delta x_j \Delta t_j & \sum w_j \Delta y_j \Delta t_j & \sum w_j \Delta t_j \Delta t_j \end{bmatrix} \begin{pmatrix} \phi_x \\ \phi_y \\ \phi_t \end{pmatrix} = \begin{pmatrix} \sum w_j \Delta x_j \Delta \phi_j \\ \sum w_j \Delta y_j \Delta \phi_j \\ \sum w_j \Delta t_j \Delta \phi_j \end{pmatrix}. \tag{5.18}$$

The stencil for cells which are degenerate on the lower time plane (*i.e.*, VERT-TRI, EDGE-TRI, and EDGE-EDGE cells) must be modified to include cells from the previous time slab. The gradient vectors are underrelaxed in the same manner as described in Section 3.3.1.

For the steady solver, the weighting factors were chosen to be the squared inverse of the distance between the cell centroids. For one case having a significantly nonuniform mesh (Section 6.3.4), this proved to be unstable. Consequently, the inverse distance, which favours more distant neighbours, has been used for all test cases.

5.3.2 Advection Term

The advective transport through a face is

$$F_f^a = J_f \phi_f, \tag{5.19}$$

where

$$J_f = \rho \mathbf{u}' \cdot \hat{\mathbf{n}}' S_f \tag{5.20}$$

is the mass transport through the face. J_f has distinctly different interpretations at space-time and time faces. At space-time faces, it represents the quantity of mass which crosses the face during the time slab, whereas at time faces it represents the quantity of mass at that time level. At both faces, ϕ_f is obtained from the same upwind-biased approximation:

$$\phi_f = \phi_{up} + \Phi \nabla' \phi \cdot \mathbf{r}'|_{up} . \tag{5.21}$$

It is interesting to note that choosing $\Phi = 0$ on orthogonal space-time gives a discretization equivalent to the backward Euler approximation for the transient term. Similarly, "downwinding" in time would be equivalent to the forward Euler approximation, and $\Phi = 1$ is similar to a three-level second-order backward-difference scheme. The current framework, however, is capable of maintaining second-order accuracy also on general space-time meshes.

The value of ϕ_f must be specified at all inflows into the space-time domain. This includes not only traditional inflow boundaries but also the initial time plane t^0, where $\mathbf{u}' \cdot \hat{\mathbf{n}}' = -1$. The values of ϕ_f specified at these time faces are precisely the same as initial conditions specified in traditional methods.

Nonlinear expressions for Φ may also be used. For instance, the limiter of Barth and Jesperson [10] may be extended to space-time by requiring that all ϕ_f around a cell be bounded by the space-time neighbours of the cell. The space-time neighbours from the next time slab must be excluded, however, for they are not yet known. As a result, accuracy is reduced to first order not only near extrema but also wherever temporal gradients are large compared with spatial gradients.

As a test case illustrating aspects of this advection scheme, consider the one-dimensional advection of a scalar in a spatial domain $0 < x < 2$ for $0 < t < 1$. Figure 5.3 illustrates the space-time domain. The velocity is two units to the right, and the initial and boundary conditions are $\phi(x, 0) = 0$ and $\phi(0, t) = 1$. The problem is solved on an orthogonal mesh having 20 cells in the x-direction and 10 cells in the t-direction. Solutions obtained using several discretizations are shown in Figure 5.4. The exact solution, which is to maintain the discontinuity between the initial and boundary condition along the characteristic line (aligned with the space-time velocity vector) is also included. The solutions obtained by solving all slabs simultaneously, such that all space-time neighbours are included in the gradient and limiter calculations, are also shown.

Most of the solutions bahave as expected. The piecewise constant scheme (b) is very diffusive. When all slabs are solved together, the piecewise linear scheme (e) generates overshoots and undershoots, which are eliminated by the limiter (f). Two

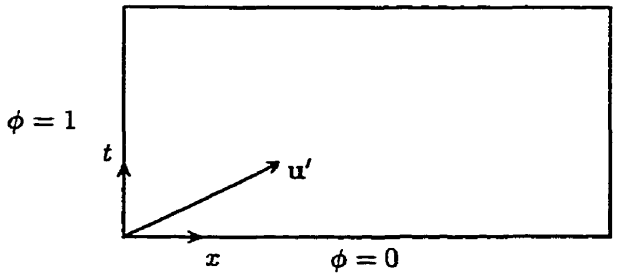


Figure 5.3: Space-time geometry for advection test case.

noteworthy features arise from the use of time slabs. First, the piecewise limiter scheme (c) has a larger overshoot than when the slabs are solved together, but no undershoot. This arises from the gradient stencil which, being one-sided in time, skews the discontinuity to one side. Second, the limited scheme (d) is more diffusive when time slabs are used; this results from the absence of neighbours from the next time slab in the limiter calculation.

5.3.3 Diffusion Term

The diffusive transport is

$$F_{\mathrm{f}}^{d} = \mathbf{q}' \cdot \hat{\mathbf{n}}' \, S_{\mathrm{f}}, \tag{5.22}$$

where $\mathbf{q}'$ is a vector having components

$$q_i' = -\gamma_{ji}' \Gamma \frac{\partial \phi}{\partial x_j'}, \tag{5.23}$$

Unlike the diffusion term considered in Section 3.3.3, this expression involves an anisotropic medium. However, the discretization developed in that section may be extended to anisotropic situations in an unambiguous manner by recognizing that the diffusive flux is driven by the component of the gradient vector in the direction of the spatial component of $\hat{\mathbf{n}}'$. By defining a vector $\mathbf{m}'$ having components

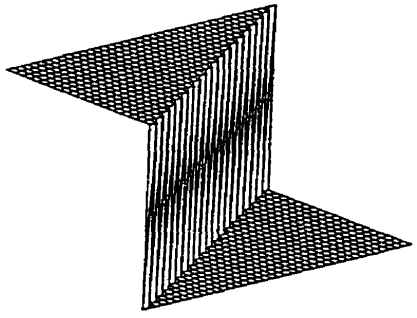
$$m_i' = \gamma_{ji}' n_j', \tag{5.24}$$

the diffusive transport becomes

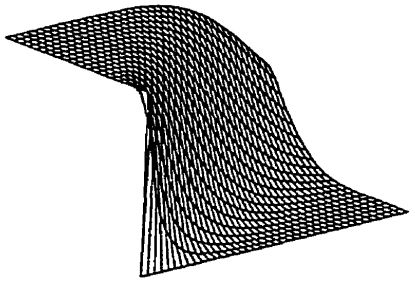
$$F_{\mathrm{f}}^{d} = -\Gamma \nabla' \phi \cdot \mathbf{m}' \, S_{\mathrm{f}}. \tag{5.25}$$

This expression has the same form as Eq. (3.11). It may be decomposed into orthogonal and nonorthogonal contributions as follows:

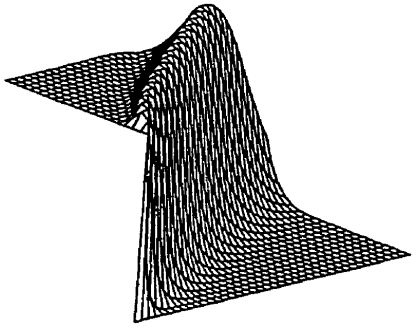
$$F_{\mathrm{f}}^{d} = -\Gamma \left(\nabla' \phi \cdot (\alpha \hat{\mathbf{s}}') + \overline{\nabla' \phi} \cdot (\mathbf{m}' - \alpha \hat{\mathbf{s}}') \right) S_{\mathrm{f}}, \tag{5.26}$$



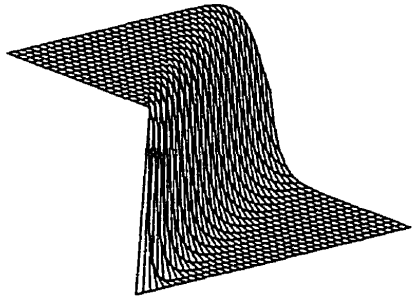
(a) Exact solution



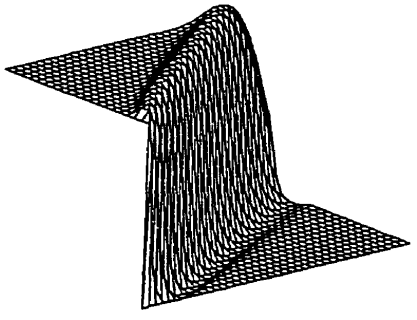
(b) Piecewise constant scheme



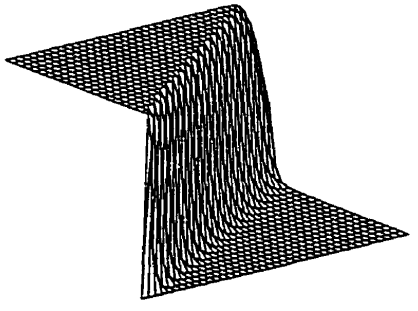
(c) Piecewise linear scheme



(d) Limited scheme



(e) Piecewise linear scheme, slabs solved together



(f) Limited scheme, slabs solved together

Figure 5.4: Space-time solutions for advection test case.

where

$$\nabla'\phi \cdot (\alpha \hat{\mathbf{s}}') = \alpha \frac{\phi_Q - \phi_P}{\Delta s'} \tag{5.27}$$

and $\overline{\nabla'\phi}$ is the average of the adjacent cell gradients. In the anisotropic case, the natural choice for the scaling factor α is $\alpha = \mathbf{m}' \cdot \hat{\mathbf{s}}'$. But at some highly nonorthogonal space-time faces, this choice may produce negative α, leading to convergence difficulties. Instead we choose

$$\alpha = \text{Max}(\mathbf{m}' \cdot \hat{\mathbf{s}}', 0). \tag{5.28}$$

This diffusion discretization is valid not only for space-time, but also for general anisotropic diffusivities. Because of the special nature of γ'_{ji} in space-time, however, F_f^d is zero at time faces.

The behaviour of this discretization has been studied by considering transient diffusion in a one-dimensional rod having unit length and translating with a velocity of 1.5. Dirichlet boundary conditions of zero are applied to the ends, and the initial condition is $\sin(\pi x)$. The analytical solution features an exponential decay of this initial condition with time. The problem is solved in the range $0 < t < 0.5$. A typical space-time mesh and solution, together with the results of a mesh refinement study, are provided in Figure 5.5. With a piecewise linear scheme for the advection term, the method is second order. With the limiter, the absence of neighbours in the next time level reduces the overall accuracy to first order, although it is still better than the piecewise constant scheme.

5.3.4 Solution Methodology

The procedure for iterating towards a converged solution within a time slab is identical to that outlined for steady flow in Section 3.3.4. After converging the solution for a time slab, the solution is reconstructed at the vertices on the upper time plane for post-processing. Then the space-time mesh for the next time slab is constructed and a new solution obtained. This process repeats until a specified stopping time is reached.

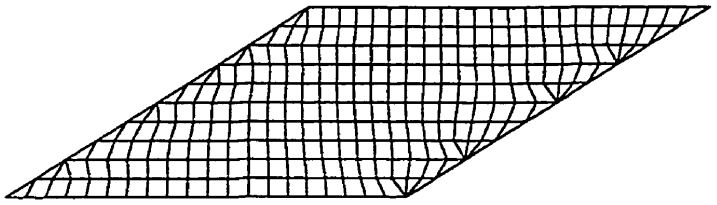
5.4 The Navier-Stokes Equations

The integral form of the continuity and momentum equations using IST notation were given as Eqs. (5.12) and (5.13). After approximating the fluxes at the faces, the discrete form of the equations are

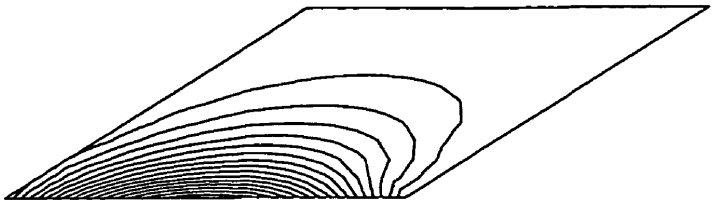
$$\sum_f J_f = 0 \tag{5.29}$$

for the continuity equation and

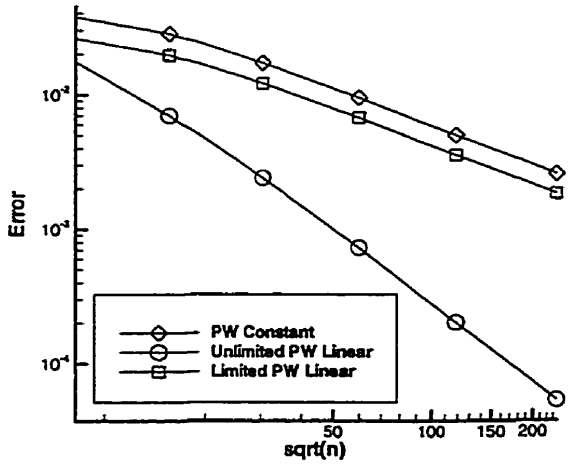
$$\sum_f (F^a_{f,i} + F^p_{f,i} + F^v_{f,i}) = 0 \tag{5.30}$$



(a) Typical space-time mesh



(b) Typical solution



(c) Mesh refinement results, with error being RMS error and n being the number of space-time cells.

Figure 5.5: Transient diffusion in a translating rod.

for the momentum equation. J_f, $F^a_{f,i}$, $F^p_{f,i}$, and $F^v_{f,i}$ respectively represent the mass flow, advective momentum transport, pressure force, and viscous force at each face. They are related to the corresponding terms for the steady case discussed in Section 3.4, but also include the transient effects. In the following sections, the discretizations developed for steady flow will be extended to space-time.

5.4.1 Advection Term

The advective transport of momentum through a face is

$$F^a_{f,i} = J_f u_{f,i}. \tag{5.31}$$

The prime is left off $u_{f,i}$ to emphasize that only the spatial components of momentum are relevant. The same upwind-biased discretization described for the scalar equation is used for $u_{f,i}$. As with the steady Navier-Stokes equations, a nonlinear limiter is not used. Instead, by choosing $\Phi = 1$, second-order accuracy is achieved everywhere. Picard iteration is used to linearize the advection term.

5.4.2 Viscous Terms

The viscous force at a face is given by

$$F^v_{f,i} = -\mu \gamma'_{kj} \left(\frac{\partial u_i}{\partial x'_k} + \frac{\partial u'_k}{\partial x_i} \right) n'_j S_f, \tag{5.32}$$

where the prime has again been left off the variables involving the index i. The first term is discretized in the same manner as the diffusive flux for the scalar equation, and the second term is lagged using known cell gradient vectors. $F^v_{f,i}$ is identically zero at time faces.

At boundary faces, $F^v_{f,i}$ must be decomposed into normal and tangential components. This process is similar to that described for the steady equations, but additional care is required to ensure that only the spatial normal and tangential vectors are considered.

5.4.3 Pressure Term

The pressure force at a face is

$$F^p_{f,i} = \gamma'_{ji} p_f n'_j S_f. \tag{5.33}$$

At interior space-time faces, a linearly-exact centered discretization is used for p_f,

$$p_f = \frac{1}{2}(p_P + p_Q) + \overline{\nabla' p} \cdot \mathbf{r}_c', \tag{5.34}$$

and boundary space-time faces are treated in a similar manner as in steady flow. At time faces, $F^p_{f,i} = 0$.

5.4.4 Mass Flows

The mass flow through a face is given by

$$J_f = \rho u'_{f,n} S_f, \tag{5.35}$$

where $u'_{f,n}$ is the space-time velocity component perpendicular to the face:

$$u'_{f,n} = \mathbf{u}'_f \cdot \hat{\mathbf{n}}'. \tag{5.36}$$

Now, defining $\mathbf{u}_f$ to be the spatial components of the space-time velocity vector, $\mathbf{n}$ and n'_t to be the spatial and time components of the space-time normal vector, and using the identity $u_t = 1$, we obtain

$$\begin{aligned} u'_{f,n} &= \mathbf{u}_f \cdot \mathbf{n} + n'_t \\ &= u_{f,n} + n'_t. \end{aligned} \tag{5.37}$$

The first term represents the spatial contribution to the mass flow; *i.e.*, the mass which exits (or enters) the control volume due to fluid flow. The second term represents the temporal contribution; *i.e.*, the mass left behind (or swallowed) as the face moves with time.

At time faces, the spatial contribution vanishes, so

$$J_f = \rho S_f. \tag{5.38}$$

At space-time faces, the spatial contribution may be discretized in a similar manner as with steady flow:

$$u_{f,n} = \overline{u}_{f,n} + \alpha f_f \left(\frac{p_Q - p_P}{\Delta s'} - \overline{\nabla' p} \cdot \hat{\mathbf{s}}' \right) - c f_f (u^o_{f,n} - \overline{u}^o_{f,n}), \tag{5.39}$$

where

$$c = \frac{\rho}{\Delta t}, \tag{5.40}$$

$$f_f = \frac{d_f}{1 - c d_f}, \tag{5.41}$$

and

$$d_f = -\frac{1}{2} \left(\frac{\Omega'_P}{a_P} + \frac{\Omega'_Q}{a_Q} \right). \tag{5.42}$$

The superscript o denotes values from the previous time slab. In cases where there is no unique space-time face from the previous time slab which corresponds to the face, an average value is used.

5.4.5 Solution Methodology

The procedure for iterating towards a converged solution within a time slab is identical to that outlined for steady flow in Section 3.4.5. The time step loop which advances the solution through time is the same as for the unsteady scalar equation.

5.5 Validation

This section describes a test case involving prescribed boundary motion. Consider a channel featuring a moving indentation in one wall. Experimental studies of this type of flow have been carried out by Pedley and Stephanoff [51], and numerical studies have been performed using a vorticity-stream function approach [56] and a finite volume method [16]. The geometry is shown in Figure 5.6. The oscillation period is T and the normalized time is $t^* = t/T$. The height of the indentation at a particular time is defined by

$$h = .19(1 - \cos(2\pi t^*)) \tag{5.43}$$

and the curved portion of the lower wall is described by

$$y = \begin{cases} 0.5h(1 - \tanh(4.14(x - 5.25))) & \text{if} \quad 4b < x < 6.5b, \\ 0.5h(1 + \tanh(4.14(x + 5.25))) & \text{if} \quad -6.5b < x < -4b. \end{cases} \tag{5.44}$$

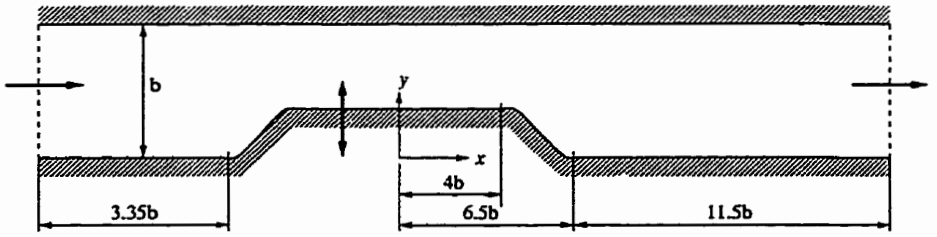


Figure 5.6: Geometry of moving indentation test case, with $b = 1$.

As with the previous numerical studies, only the first cycle of the flow is solved. Fully developed conditions are assumed at the inflow and at $t = 0$. Solutions have been obtained on a coarse mesh (having 6622 triangles on the initial time plane and 50 time slabs) and a fine mesh (having 25,896 triangles and 100 slabs). The coarse mesh required 10-25 (but usually 12-20) iterations to reduce all normalized residuals below 10^{-4} on every time slab. The fine mesh required 9-27 iterations.

The spatial meshes on various time planes around the downstream end of the indentation for the coarse run are given in Figure 5.7. The qualitative nature of the solution may be inferred from the u-velocity contours given in Figure 5.8. As the

indentation moves into the channel, several downstream separation regions appear along the lower and upper walls. Then, as the indentation recedes, these regions are pushed downstream and break up.

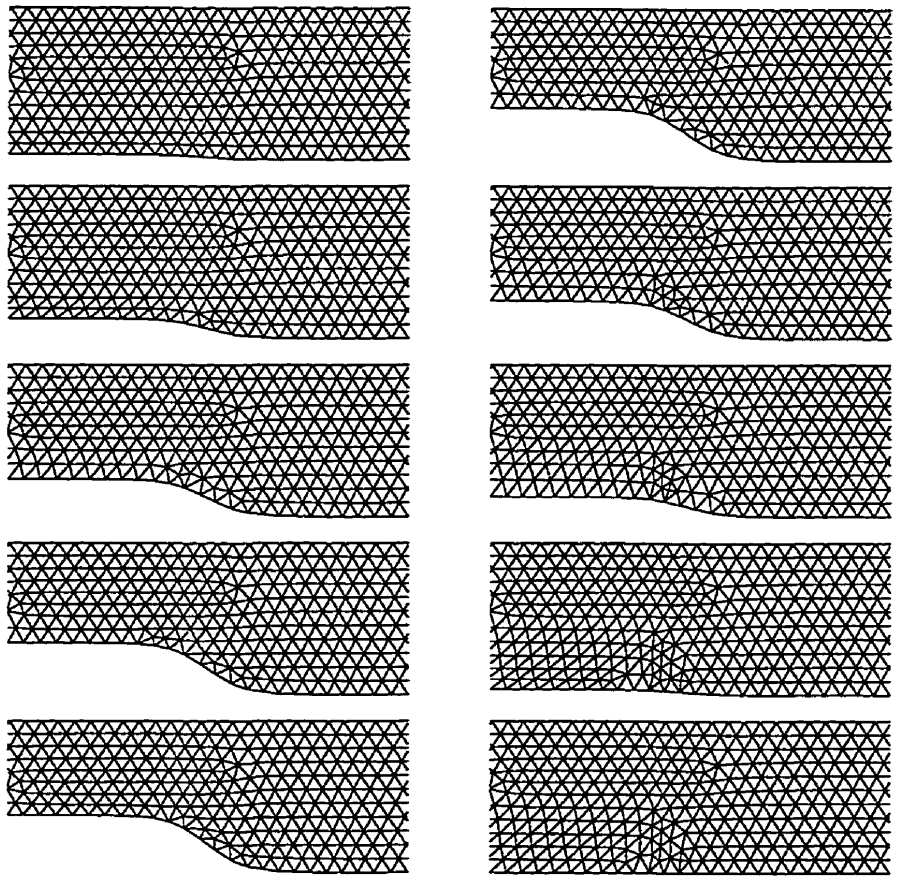


Figure 5.7: Mesh in the downstream vicinity of the indentation for the moving indentation test case on time planes $t^* = 0.1, 0.2, \ldots, 1.0$. The sequence runs by column.

More quantitative information can be obtained by plotting the nondimensional shear stress along the channel walls at various time levels. The profiles along the upper wall are given in Figure 5.9 and along the lower wall in Figure 5.10. Both coarse and fine mesh results are provided. The profiles are not entirely the same, indicating that a fully mesh-independent solution has not yet been obtained. However, they feature the same trends and are consistent with the results provided by Demirdžić and Perić [16].

A noteworthy feature of the shear stress plots is the presence of small high-frequency kinks along a few of the curves for the lower wall. These kinks appear on

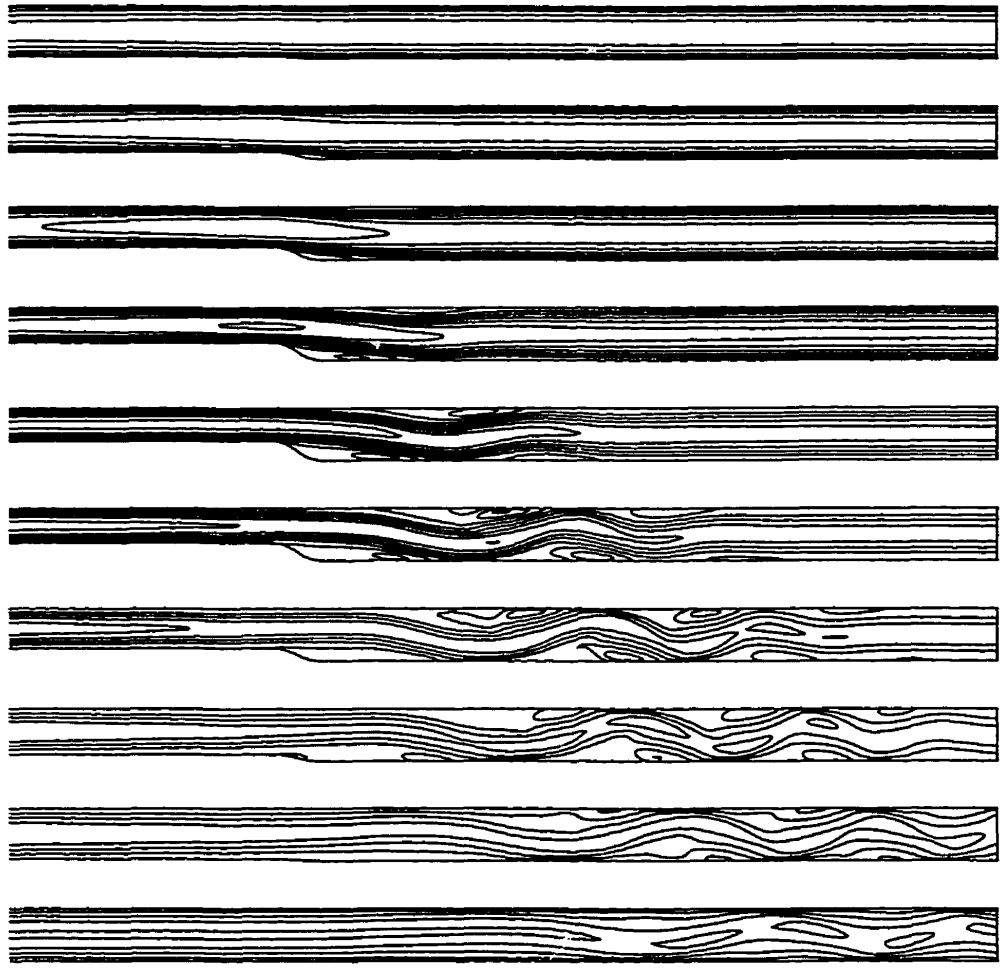
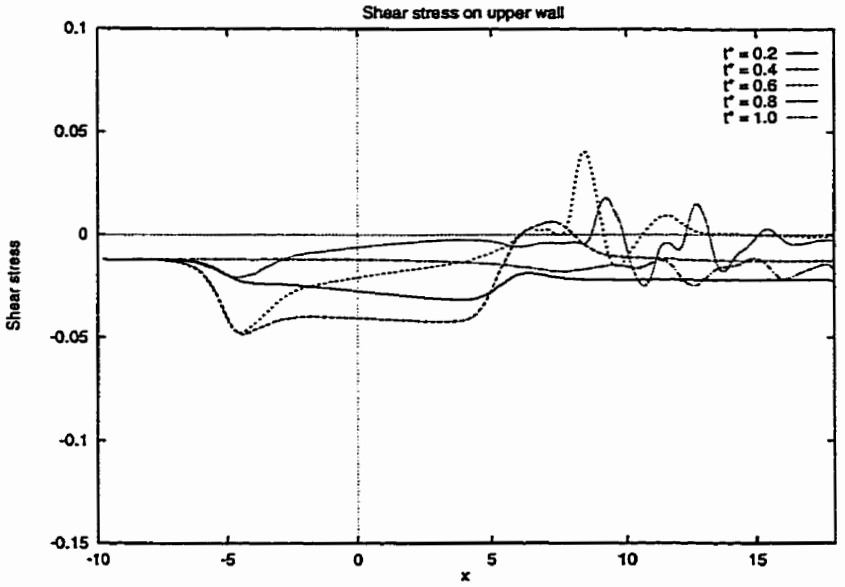
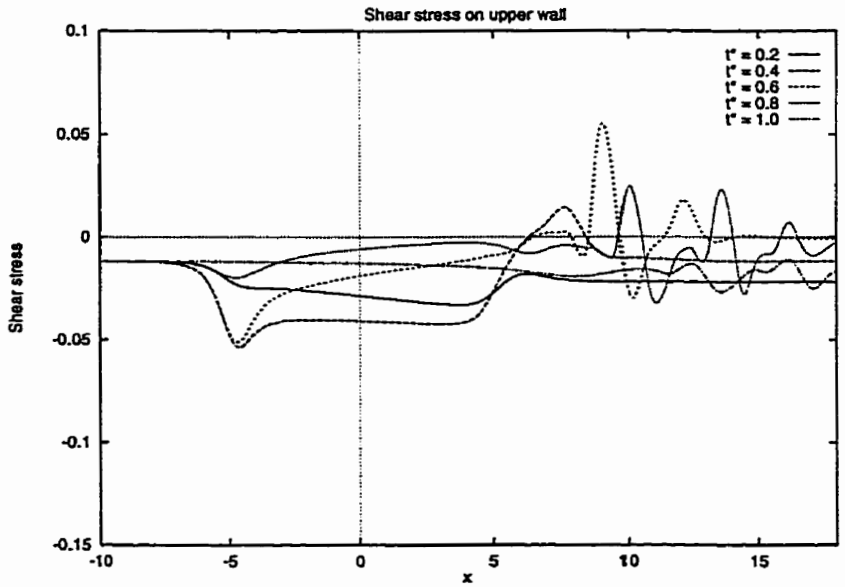


Figure 5.8: Contours of u-velocity for moving indentation test case on time planes $t^* = 0.1, 0.2, \ldots, 1.0$.

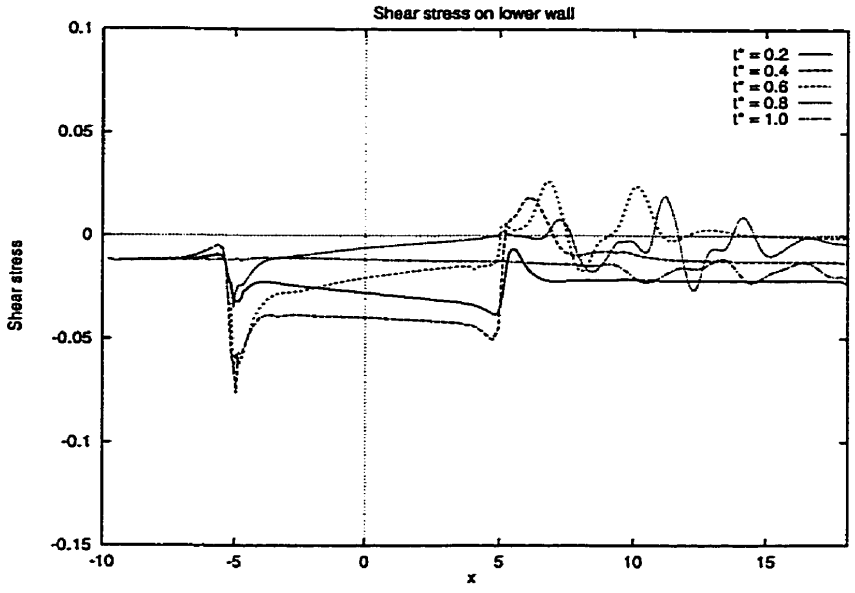


(a) Coarse mesh

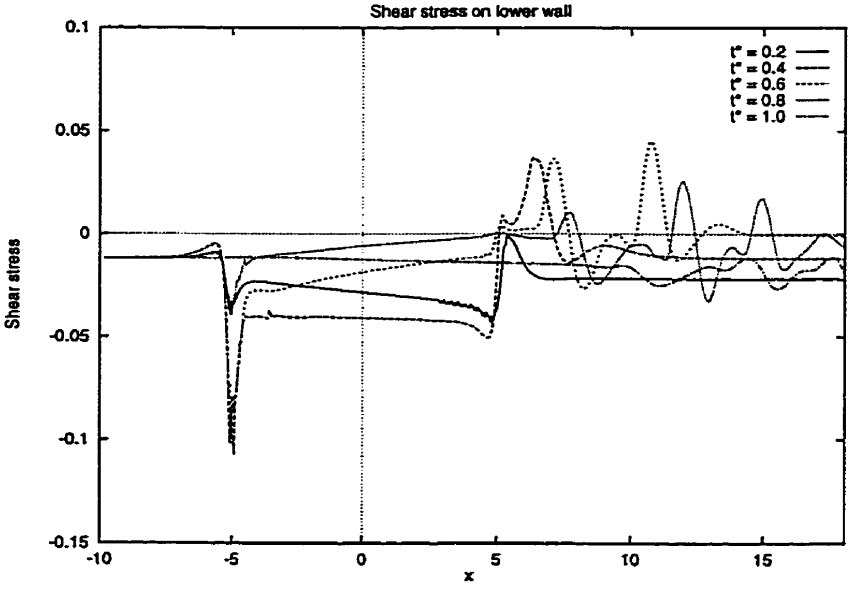


(b) Fine mesh

Figure 5.9: Shear stress profiles along the upper channel wall for the moving indentation test case.



(a) Coarse mesh



(b) Fine mesh

Figure 5.10: Shear stress profiles along the lower channel wall for the moving indentation test case.

space-time faces which are adjacent to special space-time cell types, such as EDGE-TRI or TRI-EDGE cells. The kinks are much larger when a first-order advection scheme is used, indicating that they are discretization-induced. They are apparently a consequence of the time offset between the centroids of the boundary face and adjacent cell.

Chapter 6

Free Surface Flows

In the previous chapter, an IST algorithm for unsteady flows was developed and applied to problems having prescribed boundary motion. In this chapter a different class of moving boundary problems is considered: free surfaces flow. The conditions which hold at free surfaces — the kinematic and dynamic conditions — were presented in Chapter 2. By assuming an ideal free surface, the dynamic conditions reduce to a slip condition in the tangential direction and an imposed pressure in the normal direction:

$$\tilde{p} = \tilde{p}_{\text{fs}}. \tag{6.1}$$

Eq. (6.1) may also be expressed using the modified pressure p rather than the true pressure $\tilde{p}$:

$$p = \tilde{p}_{\text{fs}} - \rho g_i (r_i - r_{i,\text{ref}}). \tag{6.2}$$

These conditions are not difficult to apply and are not considered further.

The greater difficulty lies with the kinematic condition, which is used to determine the interface position. It may be expressed as

$$\mathbf{u} \cdot \mathbf{n} = \mathbf{u}_{\text{fs}} \cdot \mathbf{n}. \tag{6.3}$$

Chapter 2 discussed several approaches to applying this condition. This study adopts an adaptive Eulerian scheme. The conservation equations and the kinematic condition are solved in a segregated manner, since enforcing them simultaneously is an advantage primarily for steady flows where large time steps are useful. The algorithm used to link the two equations is as follows:

1. Solve the continuity and momentum equations, treating the free surface as a pressure boundary.
2. Calculate the mass flows through the free surface faces.

3. Update the free surface position to drive these mass flows to zero.
4. Repeat steps (1)-(3) until convergence.

Clearly, steps (2) and (3) are the crucial steps which determine the free surface position. Conventional finite volume methods find the mass flow rate through a free surface face as

$$\dot{m}_{\rm fs} = \rho(\mathbf{u} - \mathbf{u}_{\rm fs}) \cdot \hat{\mathbf{n}} S_{\rm f} \tag{6.4}$$

and then drive it to zero by adjusting the cell volume adjacent to the face:

$$\frac{d\Omega}{dt} = -\frac{1}{\rho}\dot{m}_{\rm fs}. \tag{6.5}$$

With the IST finite volume method, a space-time formulation of the kinematic condition must be satisfied. This condition is stated simply as: no mass crosses the space-time faces which lie on the free boundary. Thus step (2) requires finding the mass crossing the space-time faces,

$$J_{\rm fs} = \rho \mathbf{u}' \cdot \mathbf{n}' S_{\rm f}, \tag{6.6}$$

and step (3) involves adjusting the vertex positions on the upper time plane so that

$$J_{\rm fs} = 0. \tag{6.7}$$

How this is performed is described in the following sections, first for one-dimensional flows and then for two-dimensional flows. Various example problems will also be given.

6.1 One-Dimensional Flows

The implementation of the kinematic condition for one-dimensional flows is straightforward. Consider the space-time boundary face shown in Figure 6.1. After solving the continuity and momentum equations, a mass $J_{\rm fs}$ is found to pass through the space-time boundary face. To satisfy the kinematic condition, a new boundary vertex location $x_{\rm fs}$ must be found which gives $J_{\rm fs} = 0$. From the definition of $J_{\rm fs}$, this occurs when

$$\rho \mathbf{u}' \cdot \mathbf{n}' S_{\rm f} = 0, \tag{6.8}$$

or, expanding the dot product,

$$\rho(n_t + u_x n_x) S_{\rm f} = 0, \tag{6.9}$$

where u_x is the advecting velocity at the free surface face. But the components of the normal vector are

$$n_t S_{\rm f} = -(x_{\rm fs} - x_{\rm fs}^{n-1}), \tag{6.10}$$

$$n_x S_{\rm f} = \Delta t. \tag{6.11}$$

Solving for x_{fs} gives

$$x_{fs} = x_{fs}^{n-1} + u_x \Delta t. \tag{6.12}$$

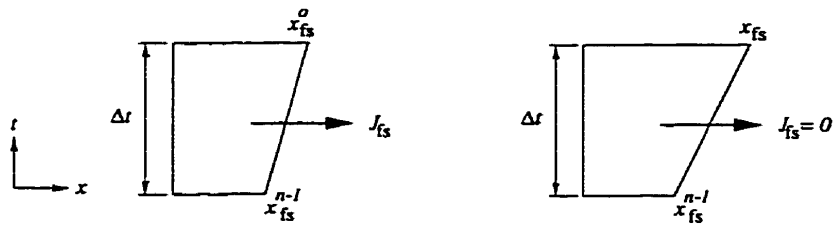


Figure 6.1: Application of kinematic condition for one-dimensional problems. *Left:* before; *right:* after.

An equivalent result may be obtained using a residual formulation. Although the equation to be solved is Eq. (6.9), at a particular iteration there remains a residual r ($r = J_{fs}$):

$$\rho(n_t^o + u_x n_x) S_f = r, \tag{6.13}$$

where n_t^o is the existing time-component of the normal vector. Subtracting Eq. (6.13) from (6.9) gives

$$\rho(n_t - n_t^o) S_f = -r. \tag{6.14}$$

But $n_t S_f = -(x_{fs} - x_{fs}^{n-1})$ and $n_t^o S_f = -(x_{fs}^o - x_{fs}^{n-1})$. Solving for x_{fs} gives

$$x_{fs} = x_{fs}^o + \frac{r}{\rho}. \tag{6.15}$$

For one-dimensional problems, either Eq. (6.12) or (6.15) may be used to update the free vertex position. In higher dimensions, however, the equations to be solved are nonlinear and the residual form is more useful.

6.2 Two-Dimensional Flows

Application of the kinematic condition for one-dimensional flows is straightforward because there is one free space-time face for each free vertex. Consequently, there is one equation which may be used to find every free vertex position. Unfortunately, the same is not true in higher dimensions. For example, consider the two-dimensional case shown in Figure 6.2, illustrating a hypothetical mesh on a time plane near a free boundary. Depending on whether the corner vertices are free or fixed, there may be three, four, or five free vertices along the free surface. But,

assuming that all the space-time faces are quadrilateral, there are always four space-time faces. This illustrates that there is in general a mismatch between the number of equations (the kinematic conditions for the space-time faces) and the number of unknowns (the free vertex positions). The presence of triangular space-time faces may also change the balance.

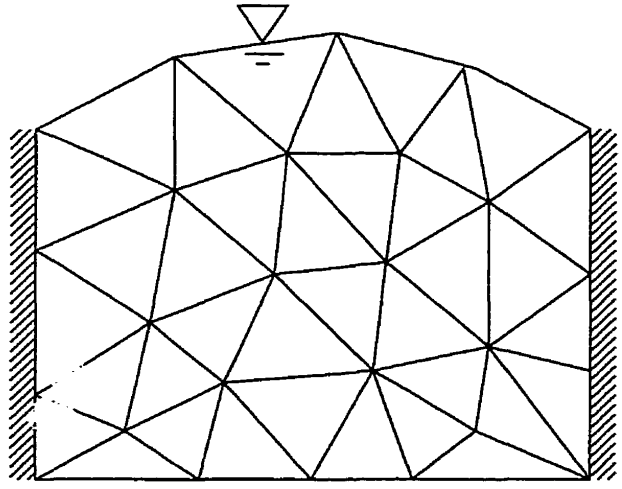


Figure 6.2: Two-dimensional free surface.

This issue has been noted in the literature, although not in the context of a space-time method. One work-around is to stagger the control volumes below the free surface, such that there is an exact correspondence between the number of vertices and control volumes [66]; however, the resulting algorithm complexity is significant. Another approach is to introduce new degrees of freedom in the form of control points on each face [45]. The kinematic conditions are used to find the control point positions, and the vertex positions are obtained by interpolating between the control point positions. The disadvantage of this method is the conceptual inconsistency in the mesh representation.

This study adopts a new approach, wherein the kinematic condition is not satisfied for each face independently, but rather for an appropriate union of faces surrounding each free vertex. Define ξ_i to be the set of free faces which touch a free vertex. Then, instead of enforcing $J_f = 0$ for all free faces, the condition

$$\sum_{j \in \xi_i} J_j w_{ji} = 0 \tag{6.16}$$

is enforced for all free vertices. The weighting factor w_{ji} determines the fraction of the mass flow through face j which is apportioned to vertex i. It is chosen as $w_{ji} = 1/n$, n being the number of free vertices which touch the face. In most cases,

this choice leads to $w_{ji} = 1/2$, but at some fixed boundaries and for some triangular space-time faces it gives $w_{ji} = 1$.

A direction of motion must also be chosen for each vertex. This study chooses the average normal of the free edges incident to the vertex.

Using this methodology, a matrix equation for the vertex displacements may be constructed. For each vertex i, the mass flow through the portion of the free surface associated with the vertex is defined as

$$F_i = \sum_{j \in \xi_i} J_j w_{ji}. \tag{6.17}$$

F_i should be zero for all free vertices, but at a particular iteration there is a residual mass flow $F_i = r_i$. By defining Δs_k to be the displacement of vertex k which will drive these residuals to zero, Newton's method may be used to obtain the displacements as follows:

$$\frac{\partial F_i}{\partial s_k} \Delta s_k = -r_i. \tag{6.18}$$

The Jacobian matrix, $\partial F_i / \partial s_k$, is tridiagonal.

On its own, this algorithm is not stable. In particular, the case of there being fewer free faces than vertices results in a singular matrix, which leads to unconstrained wiggles in the free surface. Problems also exist when there are enough free faces. An analysis of the dominant effects on the matrix indicates that a typical row has the form

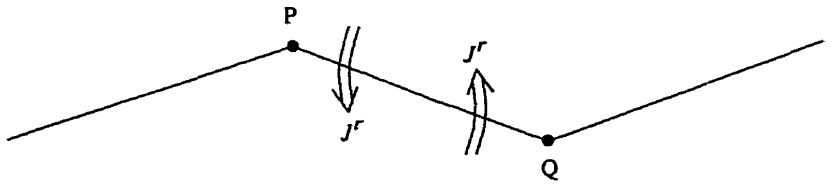
$$-\frac{\rho l}{4}(\Delta s_{i-1} + 2\Delta s_i + \Delta s_{i+1}) = -r_i, \tag{6.19}$$

where l is the length of the edges on the free surface. All coefficients have the same sign. This is physically reasonable, for shifting a vertex i in its perpendicular direction will decrease the mass flow exiting the faces associated with both itself and its neighbour vertices. However, the resulting solution is prone to wiggles. So this algorithm is susceptible to two types of wiggles: unconstrained wiggles when there are too few faces and constrained wiggles when there are enough faces.

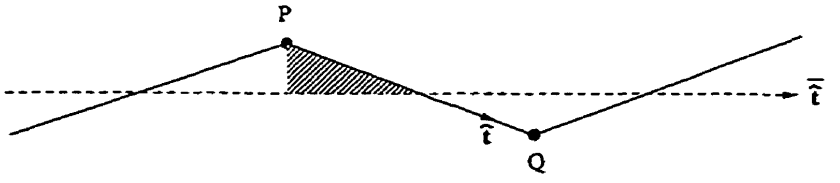
Such wiggles — whatever their origin — are eliminated using a new procedure called *mass redistribution* along the free surface. Consider the wiggly free surface shown in Figure 6.3(a). The wiggle will be damped if the mass flow F_P for vertex P is decreased by an amount J^r and the mass flow F_Q for vertex Q is increased by the same amount. In essence, mass is redistributed between vertices across free surface faces. If J^r_{ji} represents the redistributed mass to vertex i across face j, the mass flow through the portion of the free surface associated with vertex i is redefined as

$$F_i = \sum_{j \in \xi_i} (J_j w_{ji} + J^r_{ji}). \tag{6.20}$$

It remains to find an expression for the redistributed mass J^r_{ji}. From Figure 6.3(b), it is clear that a wiggle leads to a difference between the local edge tangent vector $\hat{\mathbf{t}}$ and the mean edge tangent vector $\bar{\hat{\mathbf{t}}}$. (The mean edge tangent is



(a) Free surface wiggle



(b) Calculating the redistributed mass

Figure 6.3: A mechanism for damping free surface wiggles.

determined as the average of the vertex tangents which touch the edge, where each vertex tangent is itself the average of the edge tangents which it touches.) The wiggle will be eliminated when the mass associated with the area of the cross-hatched triangle is redistributed from vertex P to vertex Q. If A is the triangle area, then

$$\begin{aligned} J^r &= \rho A \\ &\approx \rho \frac{l^2}{8}(\hat{\mathbf{t}} \times \bar{\hat{\mathbf{t}}}). \end{aligned} \tag{6.21}$$

Then $J^r_{jP} = -J^r$ and $J^r_{jQ} = J^r$. This redistribution is not performed along faces which touch walls, where it may destroy physically reasonable curvature.

All terms in the definition of the mass flows F_i associated with the free vertices (Eq. (6.20)) are now defined. Applying the condition that $F_i = 0$ for all free vertices leads to a tridiagonal Jacobian matrix, which is constructed numerically by column and solved using a tridiagonal matrix algorithm.

It is valuable to consider the conservation properties of this algorithm. Define ψ_j to be the set of free vertices which touch face j. Although the algorithm does not satisfy the kinematic condition for all faces independently, it does satisfy it both globally and in the neighbourhood of each vertex provided that, for every face,

$$\sum_{i \in \psi_j} w_{ji} = 1 \tag{6.22}$$

and

$$\sum_{i \in \psi_j} J^{\tau}_{ji} = 0. \tag{6.23}$$

This may be proven by summing the mass flows through all free surface faces:

$$\sum_j J_j = \sum_j \left\{ \left(\sum_{i \in \psi_j} w_{ji} \right) J_j + \sum_{i \in \psi_j} J^{\tau}_{ji} \right\} \tag{6.24}$$

$$= \sum_j \sum_{i \in \psi_j} \left(w_{ji} J_j + J^{\tau}_{ji} \right). \tag{6.25}$$

But if a vertex i is in ψ_j, it must also be true that those faces j are also in ξ_i. The summation may therefore be shifted from being over all free faces to being over all free vertices:

$$\sum_j J_j = \sum_i \sum_{j \in \xi_j} \left(w_{ji} J_j + J^{\tau}_{ji} \right) \tag{6.26}$$

$$= \sum_i 0, \tag{6.27}$$

because that is the equation being enforced for each vertex. Consequently the kinematic condition is satisfied both in the neighbourhood of each free vertex and for the free surface as a whole.

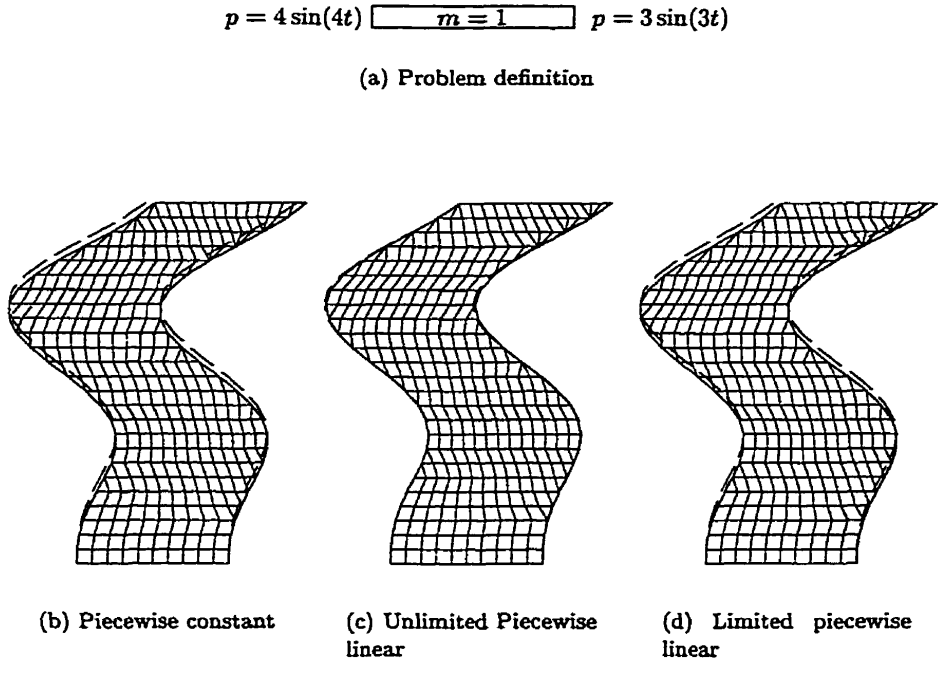
6.3 Validation

6.3.1 Motion of a One-dimensional Slug

As an example of the performance of this method for one-dimensional flows, consider a slug of fluid of unit length and density. Oscillating pressures imposed on the left and right boundaries force the slug to move, as illustrated in Figure 6.4. The boundary pressures are $4\sin(4t)$ on the left side and $3\sin(3t)$ on the right side. A mesh consisting of 10 spatial cells at $t = 0$ and a time step of 0.1 is used to solve the problem in the range $0 < t < 2.5$. The space-time meshes for three advection schemes are shown in Figure 6.4. The accuracy can be assessed by comparing the boundary of the space-time mesh with the analytical boundary positions, also shown on the plots. Even on this coarse mesh, the solutions are quite accurate.

6.3.2 Rotating Slice of Fluid

The next test case serves to verify the free surface algorithm for two-dimensional flows. A slice of fluid, having a length l and initial height h_0, rotates about the



(a) Problem definition

(b) Piecewise constant

(c) Unlimited Piecewise linear

(d) Limited piecewise linear

Figure 6.4: One-dimensional free boundary test case. The dashed line indicates the analytical boundary position.

x—axis with angular velocity ω, as illustrated in Figure 6.5(a). At steady-state, the free-surface profile established by the fluid is

$$\frac{h(x)}{h_0} = 1 - \frac{(\omega l)^2}{6gh_0}\left\{1 - 3\left(\frac{x}{l}\right)^2\right\}. \tag{6.28}$$

From this expression a dimensionless spin rate may be identified as $(\omega l)^2/6gh_0$.

This flow is modelled by including a volumetric source term $\rho\omega^2 x$ in the u—momentum equation. It is solved on a domain having $h_0 = l$ using an initial mesh of 247 cells, as shown in Figure 6.5(b). The viscosity is chosen so as to obtain the steady-state profile relatively quickly without overshoots. Solutions were obtained for dimensional spin rates of 1/6 and 1/2. The resulting final meshes are also included in the figure, together with the analytical free surface profiles. The analytical and computed solutions are nearly indistinguishable.

6.3.3 Breaking Dam

The next test case involves the collapse of a two-dimensional column of fluid. Experiments of this nature using several configurations were performed some time ago by Martin and Moyce [40]. It is also a common numerical test case [30, 34, 49, 57]. The case having an initial aspect ratio $h_0/w_0 = 2$, where h_0 is the initial column height and w_0 is the initial width, is considered. As viscous effects are negligible, the flow is modelled as inviscid. A dimensionless time is defined as

$$t^* = \sqrt{\frac{2g}{w_0}}t \tag{6.29}$$

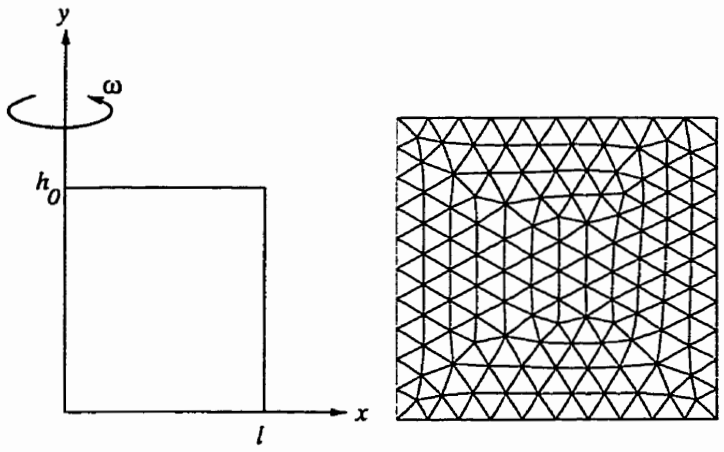
and a dimensionless front position as

$$w^* = \frac{w}{w_0}. \tag{6.30}$$

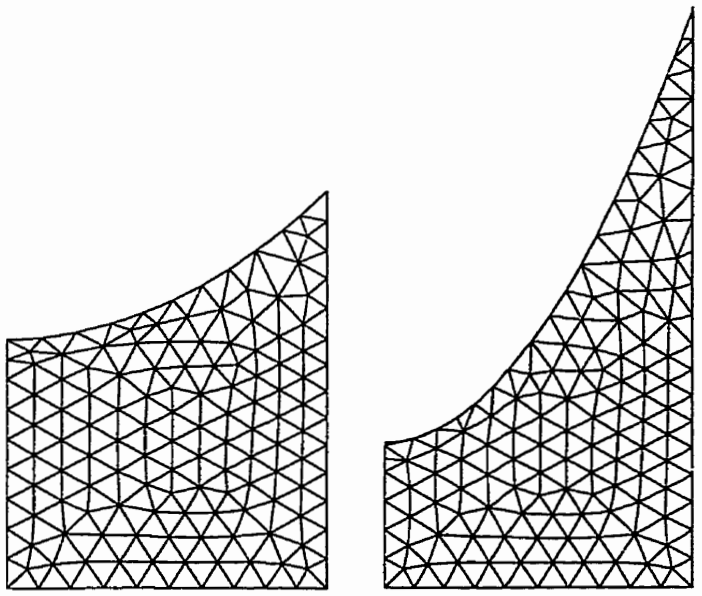
The problem is solved in the time range $0 < t^* < 5$ using a coarse mesh (having initially 130 cells and $\Delta t^* = 0.05$) and a fine mesh (having 500 cells and $\Delta t^* = 0.025$).

The spatial mesh for the fine grid results is shown for various time levels in Figure 6.6. The figure clearly demonstrates the capacity of the method to handle large changes in the free surface while maintaining mesh quality. The only anomaly is in the second frame ($t^* = 0.5$), where there is a small kink in the corner resulting from the mass redistribution near the corner. The kink does not affect the results elsewhere in the domain, and other results of the same test case reported in the literature reveal similar anomalies.

To compare the numerical results with the experiments, a plot of front position w^* against time is given in Figure 6.7. It is important to point out that the experimental results have undergone a time shift of $\Delta t^* = 0.3$ in order to compensate for uncertainties in the time origin. This shift is consistent with what is performed (although not acknowledged) in other numerical studies in the literature. With this shift, there is excellent agreement between the experimental and computed results. The plot also shows that the solution is nearly mesh-independent.



(a) Problem definition (b) Initial mesh



(c) Final mesh, $\frac{(\omega l)^2}{6gh_0} = \frac{1}{6}$ (d) Final mesh, $\frac{(\omega l)^2}{6gh_0} = \frac{1}{2}$

Figure 6.5: Rotating slice test case. The analytical free surface positions are indicated with a dashed line.

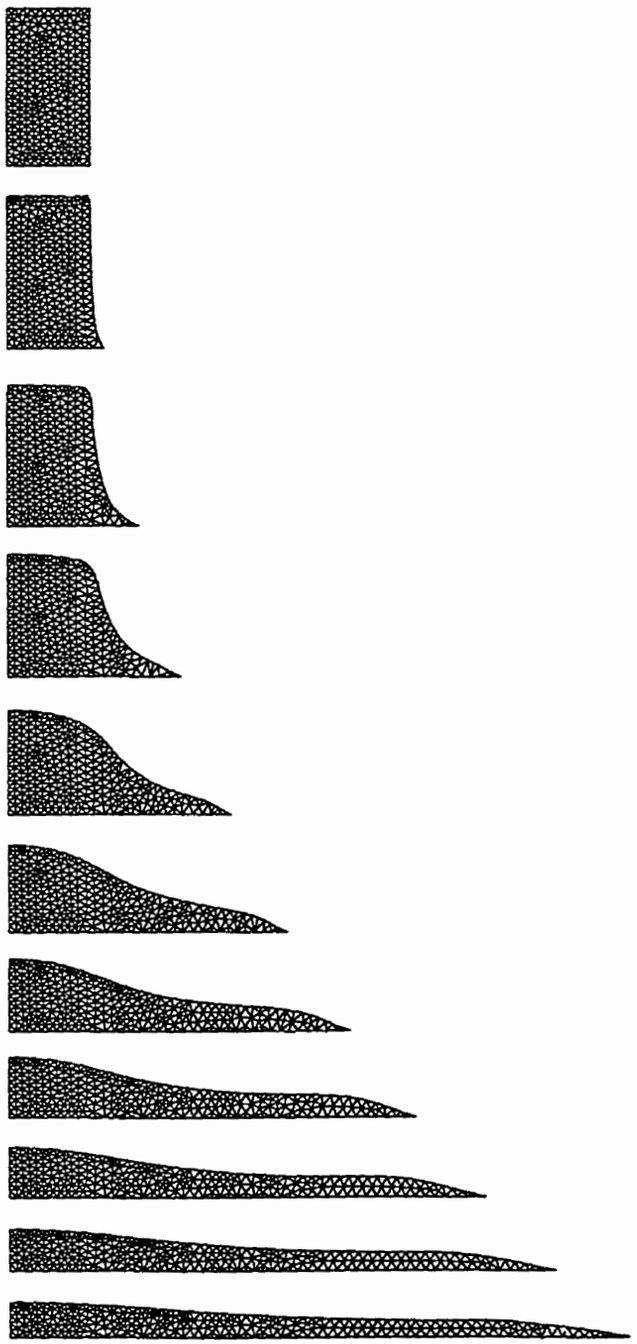


Figure 6.6: Mesh development for breaking dam test case on time planes $t^* = 0, .5, 1, 1.5, \ldots, 5.0$.

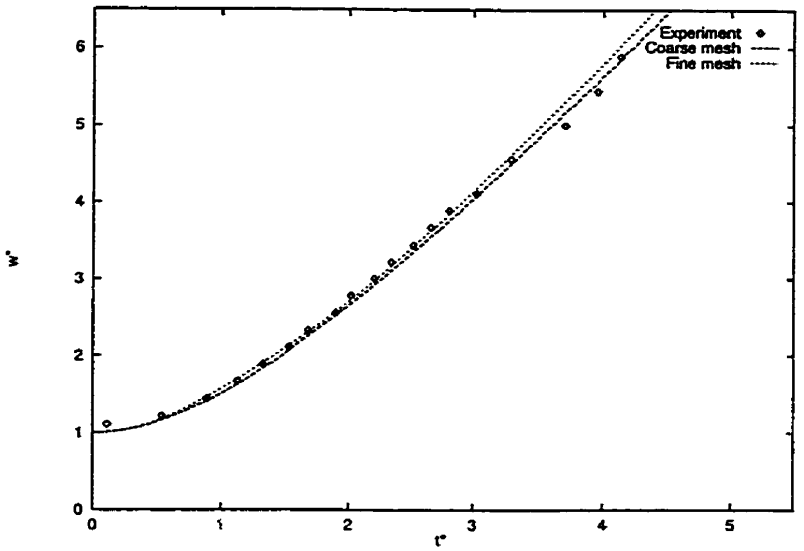


Figure 6.7: Comparison between calculated and experimental results for the breaking dam test case. The plot shows the evolution of the dimensionless front position w^* with dimensionless time t^*.

6.3.4 Overturning Wave

A final test case illustrates the capability of this free surface algorithm on a challenging flow — an overturning wave. The wave is generated in a water channel by a piston wavemaker, as described by Dommermuth *et al.* [18]. The piston has a time-varying frequency, amplitude, and phase carefully chosen to generate a plunging breaker some distance downstream. These authors performed both an experimental study and numerical calculations using a nonlinear panel method.

The wave channel geometry is illustrated in Figure 6.8. The dimensions are normalized by the undisturbed water height, so that the height is 1 m and the length 20 m. Inviscid flow is assumed, and the density and gravitational constant are set to unity. These conditions are consistent with those in the numerical calculations of Dommermuth *et al.* [18]. The piston velocity is expressed in terms of its amplitude, frequency, and phase, which are in turn defined by Fourier series. The dominant frequencies are in the range of 1-2 rad/s.

The initial spatial mesh used has 23,170 triangles, whose size vary with position. For $x < 10$, where the primary process is wave propagation, the spacing along the top decreases from 0.04 at $x = 0$ to 0.03 at $x = 10$; along the bottom the spacing is 0.05. For $11 < x < 12.2$, where the wave crests and overturns, a more dense spacing of 0.01 is used near the surface. For $x > 12.2$, which is downstream of the region of interest, the spacing increases smoothly to 0.2 at $x = 20$.

For the overturning portion of the wave, even the spacing of 0.01 is not adequate, for the thickness of breaking wave may be as small as 0.03. Good resolution is particularly important near regions of high curvature at the plunger nose. If the

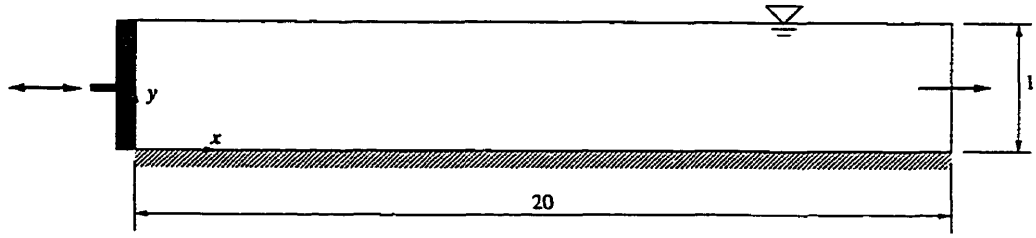


Figure 6.8: Wave channel geometry used in overturning wave test case.

resolution is inadequate, small kinks may appear in the free surface, particularly when there are irregular space-time cells arising from the addition of a vertex. An example of this behaviour is shown in Figure 6.9. Although the kinks vanish in the next time slab, when the space-time cells are of better quality, they clearly indicate an inadequate spatial resolution of the nose. A reasonable resolution for the overturning phase is obtained by reducing the spacing in the range $11.5 < x < 12.2$ for $t > 50.5$ according to $0.01/f$, where

$$f = 1 + 4\frac{t - 50.5}{51.8 - 50.5}. \tag{6.31}$$

This modification forces the spacing to decrease by a factor of five at $t = 51.8$, when collision occurs. These spacing functions lead to a mesh which is considerably finer at the end of the computations than at the beginning: the final mesh, after 3200 time slabs, has 62,586 triangles.

An adaptive time step is used, such that no vertex may move more than a specified fraction of its local spacing. The fraction is 15% if the vertex is moving into the domain and 18% if the vertex is moving outward. The difference occurs because the space-time meshing algorithm is capable of handling larger time steps if the domain expands than if it contracts. With this adaptive procedure, the time step is about 0.00035 at the end of the computations. The initial time step is set to 0.1, which is approximately one-fortieth of the piston period. Typically 3–5 iterations were required each time step to reduce all residuals below 10^{-3}.

Several enhancements to the space-time meshing algorithm described in Chapter 4 were required for this test case. First, the algorithm has been extended to nonuniform mesh spacings by basing the decision to modify the topology on local, rather than global, length scales. A second change is the smoothing of two layers of interior vertices rather than only one. Third, the topological modifications have been improved to ensure good quality mesh near regions where concave or convex surfaces (during wave cresting and overturning) undergo perpendicular motion. This step requires insertion and removal of interior vertices based on tangential criteria in addition to the perpendicular criteria outlined in Chapter 4. Finally, to avoid complexities near the corner of the piston and the free surface, vertices are not added or removed due to the piston motion. Instead, the $x-$positions of the vertices between the piston and $x = 1$ are calculated by smoothing.

Two solver difficulties have also been encountered. The first is related to the

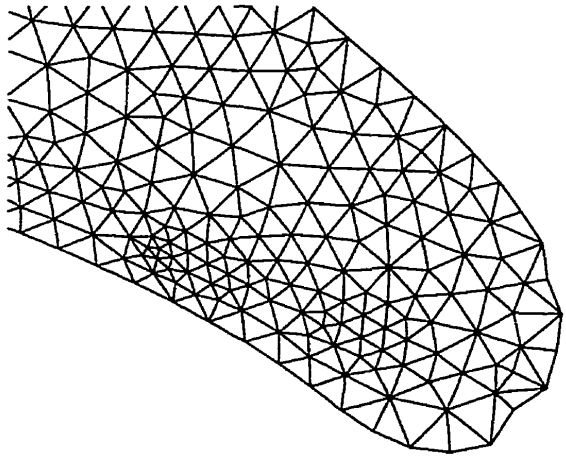


Figure 6.9: A kink near the nose of the overturning wave arising from inadequate resolution and irregular space-time cells.

weighting factors in the least-squares gradient computation. Originally the weights were the square of the inverse distance between the points used in the gradient stencil; however, this proved to be unstable near $x = 12$ where the mesh has a highly nonuniform spacing. By changing the weights to inverse distances, the instabilities vanished. The final run used a mesh with a smaller nonuniformity and was stable with the original weights. Nevertheless, the modified weights were used to generate the results reported here. All previous test cases were also re-executed with the modification to ensure that the change does not have adverse consequences.

The second solver difficulty is related to the behaviour of the pressure calculation for the very small time steps experienced toward the end of the simulation. According to Eq. (5.39), the pressure dissipation coefficient is

$$f_t = \frac{d_t}{1 - \frac{\rho d_t}{\Delta t}}. \tag{6.32}$$

For small Δt, f_t tends toward $\Delta t/\rho$, in which case it may become very small. As a result, small changes in normal velocities (generated by vertex smoothing) induce large pressure spikes, which in turn contaminate the solution. As a work-around, the coefficient was changed to

$$f_t = \frac{d_t}{1 - \frac{\rho d_t}{r\Delta t}}, \tag{6.33}$$

where $r = 1$ for $t < 50.5$, 10 for $50.5 < t < 51$, and 100 for $t > 51$. This modification recognizes that the expression for f_t is somewhat arbitrary.

The experimental and numerical results provided by Dommermuth *et al.* [18] include surface elevations at various locations along the wave channel. The elevations at the same locations using the current results are plotted in Figure 6.10.

The agreement with the experiments and computations of Dommermuth *et al.* is excellent, the biggest difference that the second-last wave passing through $x = 9.17$ has a lower amplitude in the current results. It is not clear which amplitude is more consistent with the experiments.

Outlines of the predicted surface profile during the overturning stage are plotted for various times in Figure 6.11. The final frame ($t = 51.81$) illustrates that the method does not handle the collision phase: a limitation in implementation but not in concept.

Figure 6.12 shows a close-up of the mesh at $t = 51.80$, just before collision. The shape differs somewhat from the more vertical plunge predicted by Dommermuth *et al.*. Consequently, they have a shorter time (by about 0.2 seconds) to collision. From a qualitative perspective, the current results appear more consistent with real waves [52].

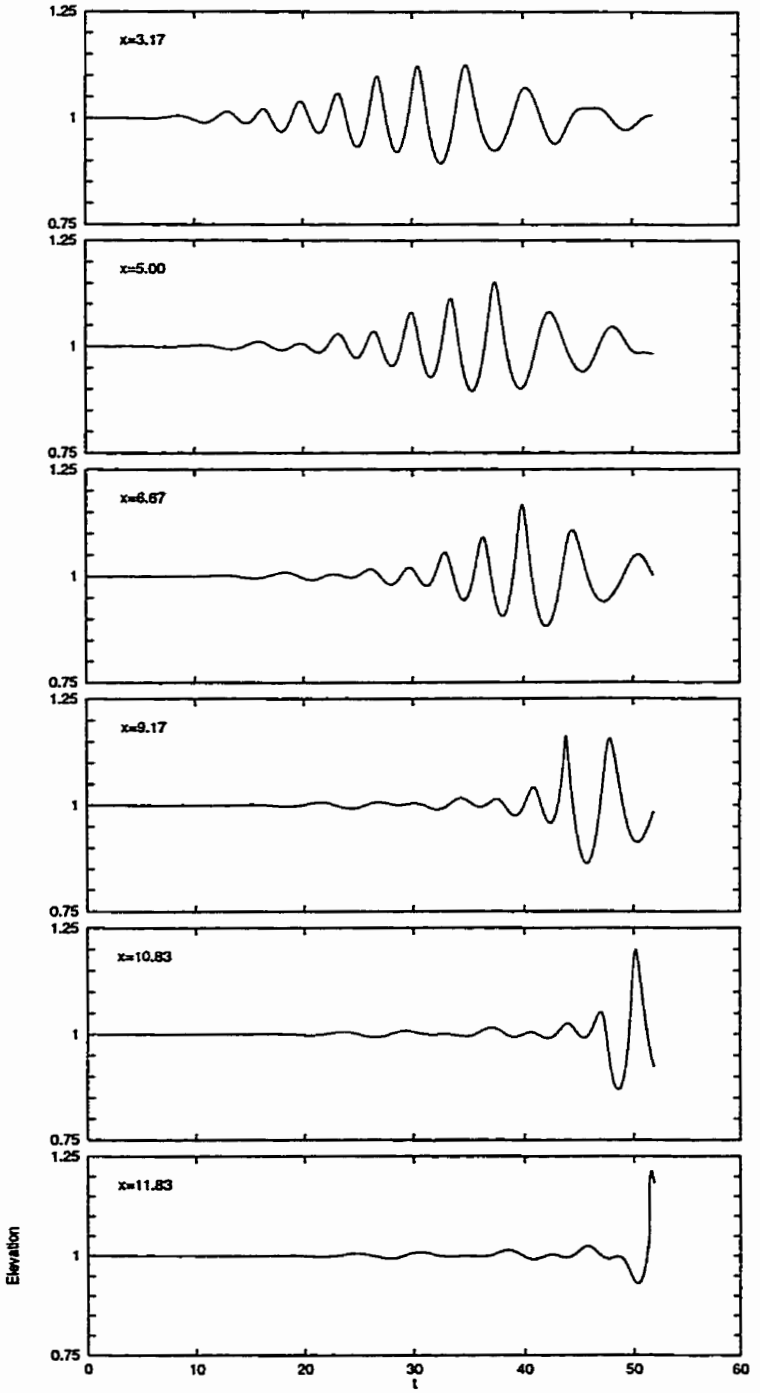


Figure 6.10: Free surface elevations against time at various locations in the wave channel for the overturning wave test case.

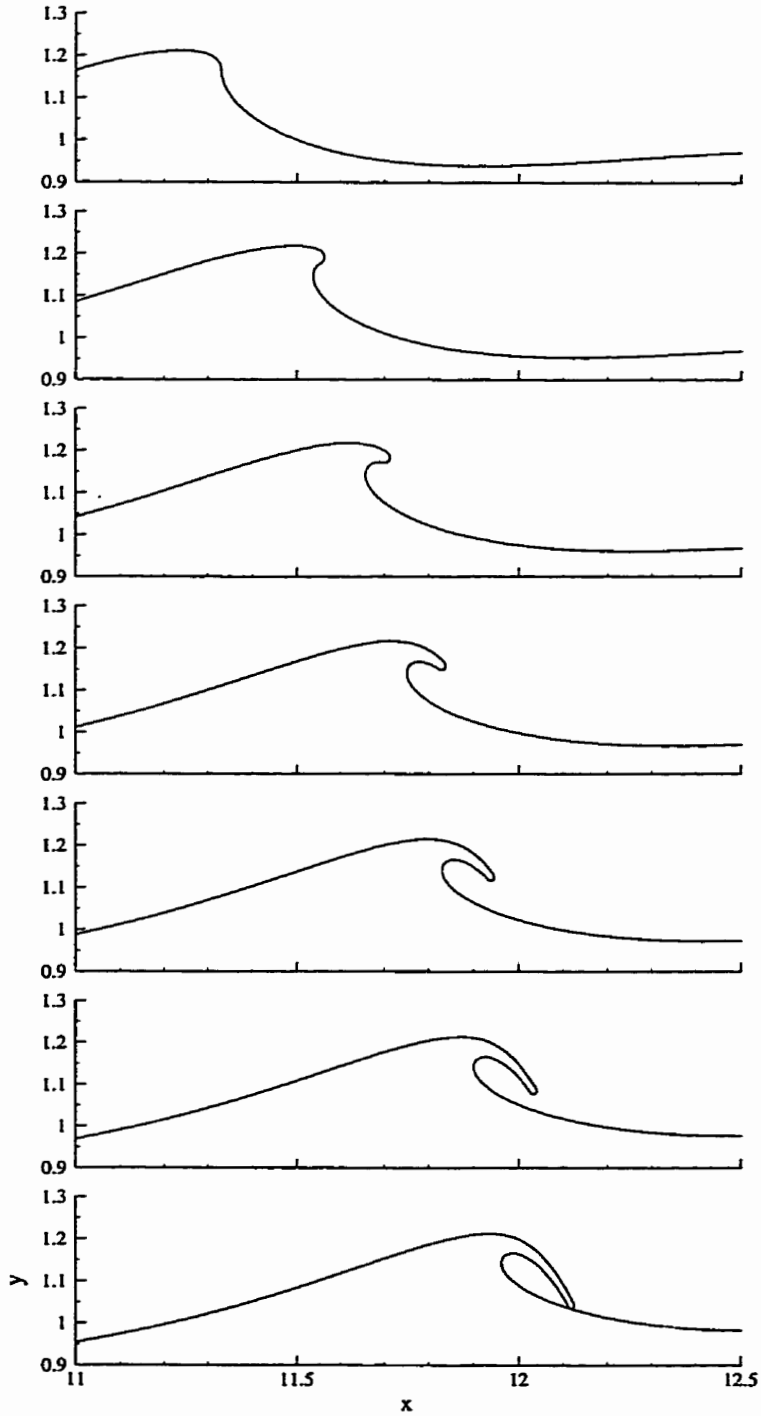


Figure 6.11: Outlines of the overturning wave at times $t = 50.70$, 51.05, 51.24, 51.40, 51.54, 51.65, 51.76.

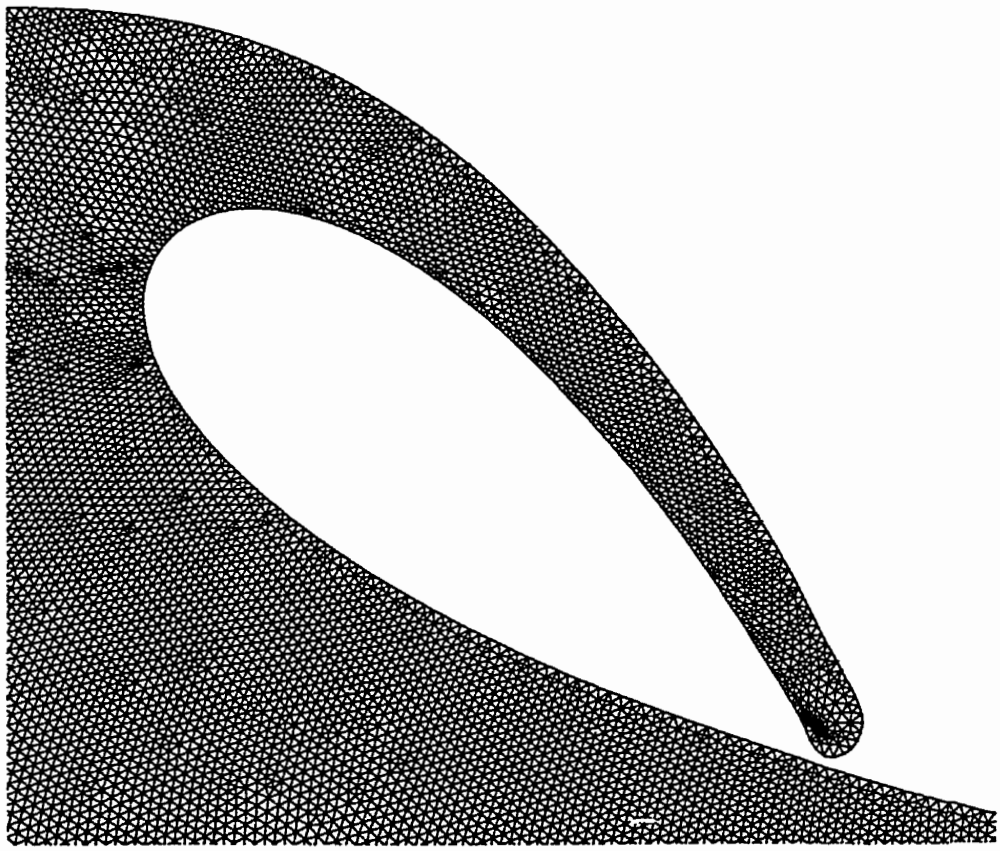


Figure 6.12: Close-up of the overturning wave at $t = 51.75$.

Chapter 7

Conclusions and Recommendations

7.1 Conclusions

This research has developed a finite volume methodology for steady flow, unsteady flow, and free surface flow. The new attributes for steady flow are summarized below.

- The diffusion discretization is second-order and linearly-exact. This is achieved by decomposing the diffusive flux into orthogonal and nonorthogonal components. Linearity-preserving gradients are used for the nonorthgonal component. By making the decomposition optimal, the discretization extends unambiguously to anisotropic media.
- A second-order correction to the approximation for face pressures and advecting velocities ensures that these terms are linearly exact.
- Extrapolating pressure to wall boundaries based on consistency with the discretization of advecting velocity is useful for triangular meshes where the least-squares matrix for the cell pressure gradient would otherwise be singular.
- Good convergence behaviour is achieved by targetting cell gradients and those regions where the mesh is nonorthogonal for underrelaxation.

The attributes for unsteady flow are given below.

- The IST framework enforces discrete conservation in both space and time, even when vertices are added and removed. There is no need to consider the geometrical conservation law or the Leibnitz Rule.
- The discretizations of space and time are unified. Second-order accuracy in time is thereby reached in the same manner as in space. If limiters are used to enforce boundedness with time slabs, however, accuracy may be reduced to first-order.

- The space-time meshing strategy for one- and two-dimensional moving boundary problems is based on making local mesh modifications near the boundary.

The attributes for free surface flow are described below.

- The kinematic condition is applied to vertices rather than faces. As a result, it is enforced not for each face independently, but rather for a subset of faces in the neighbourhood of each vertex. The mass redistribution mechanism for damping wiggles is an essential part of the method.
- The method applies to flows experiencing severe boundary motion, such as overturning waves.

7.2 Recommendations

Although this work represents a significant advance in the modelling of moving boundary problems, more work remains to be done to extend it to new types of problems. Some possibilities are given below.

- The space-time meshing algorithm has been adequate for the flows considered in this thesis. However, it does have some disadvantages: it is relatively cumbersome to code, it is not clear how to extend it to three-dimensional problems, and it is not clear how to extend it to time-accurate adaptive meshing. Further work must be done in generalizing the algorithm or in developing another mesh generation framework.
- The basic concepts of the free surface flow algorithm appear to hold for three-dimensional flows, with or without IST. Demonstrating that this is the case would be a valuable accomplishment. Incorporating surface tension in the algorithm would also be useful.
- The use of IST to achieve a conservative time-accurate solution-adaptive meshing algorithm would be useful.

Bibliography

[1] Advanced Scientific Computing Ltd. *TASCflow Theory Documentation*, 1994.

[2] M. Aftosmis, D. Gaitonde, and T. S. Tavares. Behavior of linear reconstruction techniques on unstructured meshes. *AIAA Journal*, 33:2038–2049, 1995.

[3] M. J. Aftosmis. Upwind method for simulation of viscous flow on adaptively refined meshes. *AIAA Journal*, 32:268–277, 1994.

[4] B. Alessandrini and G. Delhommeau. Simulation of three-dimensional unsteady viscous free surface flow around a ship model. *International Journal for Numerical Methods in Fluids*, 19:321–342, 1994.

[5] W. K. Anderson, J. L. Thomas, and B. van Leer. Comparison of finite volume flux vector splittings for the Euler equations. *AIAA Journal*, 24:1453–1460, 1986.

[6] B. F. Armaly, F. Durst, J. C. F. Pereira, and B. Schönung. Experimental and theoretical investigation of backward-facing step flow. *Journal of Fluid Mechanics*, 127:473–496, 1983.

[7] B. R. Baliga and S. V. Patankar. A control volume finite-element method for two-dimensional fluid flow and heat transfer. *Numerical Heat Transfer*, 6:245–261, 1983.

[8] T. J. Barth. A 3-D upwind Euler solver for unstructured meshes. *AIAA Paper 91-1548*, 1991.

[9] T. J. Barth. Aspects of unstructured grids and finite volume solvers for the Euler equations. *Agard Report 787: Special Course on Unstructured Grid Methods for Advection Dominated Flows*, 1992.

[10] T. J. Barth and D. C. Jesperson. The design and application of upwind schemes on unstructured meshes. *AIAA Paper 89-0366*, 1989.

[11] A. N. Brooks and T. J. R. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 32:199–259, 1982.

[12] S. C. Chang. The method of space-time conservation element and solution element — a new approach for solving the Navier-Stokes and Euler equations. *Journal of Computational Physics*, 119:295–324, 1995.

[13] A. J. Chorin. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics*, 2:12–26, 1967.

[14] H. Deconinck, H. Paillère, R. Struijs, and P. L. Roe. Multidimensional upwind schemes based on fluctuation-splitting for systems of conservation laws. *Computational Mechanics*, 11:323–340, 1993.

[15] I. Demirdžić and M. Perić. Space conservation law in finite volume calculations of fluid flow. *International Journal for Numerical Methods in Fluids*, 8:1037–1050, 1988.

[16] I. Demirdžić and M. Perić. Finite volume method for prediction of fluid flow in arbitrarily shaped domains with moving boundaries. *International Journal for Numerical Methods in Fluids*, 10:771–790, 1990.

[17] I. Demirdžić, Ž. Lilek, and M. Perić. Fluid flow and heat transfer test problems for non-orthogonal grids: bench-mark solutions. *International Journal for Numerical Methods in Fluids*, 15:329–354, 1992.

[18] D. G. Dommermuth, D. K. P. Yue, W. M. Lin, R. J. Rapp, E. S. Chan, and W. K. Melville. Deep-water plunging breakers: a commparison between potential theory and experiments. *Journal of Fluid Mechanics*, 189:423–442, 1999.

[19] J. Farmer, L. Martinelli, and A. Jameson. Fast multigrid method for solving incompressible hydrodynamic problems with free surfaces. *AIAA Journal*, 32:1175–1182, 1994.

[20] J. H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. Springer-Verlag, 1996.

[21] J. M. Floryan and J. Rasmussen. Numerical methods for viscous flows with moving boundaries. *Applied Mechanics Reviews*, 42:323–341, 1989.

[22] P. A. Forsyth and P. H. Sammon. Quadratic convergence for cell-centered grids. *Applied Numerical Mathematics*, 4:377–394, 1988.

[23] N. T. Frink. Recent progress toward a three-dimensional unstructured Navier-Stokes flow solver. *AIAA Paper 94-0061*, 1994.

[24] A. M. Froncioni, A. Garon, and R. Camarero. *h*-adaptive strategies for space-time finite elements. In *Proceedings of the 3rd Annual Conference of the CFD Society of Canada*, pages 47–54, 1995.

[25] D. K. Gartling. A test problem for outflow boundary conditions — flow over a backward-facing step. *International Journal for Numerical Methods in Fluids*, 11:953–967, 1990.

[26] U. Ghia, K. N. Ghia, and C. T. Shin. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 48:387–411, 1982.

[27] S. K. Godunov. A finite difference method for the numerical computation of discontinuous solutions of the equations of fluid dynamics. *Mathematischeskii Sbornik*, 47:357–393, 1959.

[28] P. M. Gresho, D. K. Gartling, K. A. Cliff, T. J. Garret, A. Spence, K. H. Winters, J. W. Goodrich, and J. R. Torczynski. Is the steady viscous incompressible 2d flow over a backward-facing step at Re=800 stable?". *International Journal for Numerical Methods in Fluids*, 17:501–541, 1993.

[29] P. Hansbo. The characteristic streamline diffusion method for convection-diffusion problems. *Computer Methods in Applied Mechanics and Engineering*, 96:239–253, 1992.

[30] P. Hansbo. The characteristic streamline diffusion method for the time-dependent Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 99:171–186, 1992.

[31] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids*, 8:2182–2189, 1965.

[32] A. Harten. On a class of high resolution total-variation-stable finite-difference schemes. *SIAM Journal of Numerical Analysis*, 21:1–23, 1984.

[33] C. W. Hirt, A. A. Amsden, and J. L. Cook. An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *Journal of Computational Physics*, 14:227–253, 1974.

[34] C. W. Hirt and B. D. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39:201–225, 1981.

[35] T. J. R. Hughes and G. M. Hulbert. Space-time finite element methods for elastodynamics: formulations and error estimates. *Computer Methods in Applied Mechanics and Engineering*, 66:339–363, 1988.

[36] S. F. Kistler and L. E. Scriven. Coating flow theory by finite element and asymptotic analysis of the Navier-Stokes system. *International Journal for Numerical Methods in Fluids*, 4:207–229, 1984.

[37] R. J. LeVeque. *Numerical Methods for Conservation Laws*. Birkhäuser Verlag, 1990.

[38] V. G. Levich and V. S. Krylov. Surface-tension-driven phenomena. *Anuual Review of Fluid Mechanics*, 1:293–314, 1969.

[39] S. Majumdar. Role of underrelaxation in momentum interpolation for calculation of flow with nonstaggered grids. *Numerical Heat Transfer*, 13:125–132, 1988.

[40] J. C. Martin and W. J. Moyce. An experimental study of the collapse of liquid columns on a rigid horizontal plane. *Philosophical Transactions of the Royal Society of London, Series A*, 244:312–324, 1952.

[41] S. R. Mathur and J. Y. Murthy. A pressure-based method for unstructured meshes. *Numerical Heat Transfer, Part B*, 31:195–215, 1997.

[42] D. J. Mavriplis. Three-dimensional multigrid Reynolds-averaged Navier-Stokes solver for unstructured meshes. *AIAA Journal*, 33:445–453, 1995.

[43] H. Miyata. Finite-difference simulation of breaking waves. *Journal of Computational Physics*, 65:179–214, 1986.

[44] H. Miyata, M. Zhu, and O. Watanabe. Numerical study on a viscous flow with free-surface waves about a ship in steady straight course by a finite-volume method. *Journal of Ship Research*, 36:332–345, 1992.

[45] S. Muzaferija and M. Perić. Computation of free-surface flows using the finite-volume method and moving grids. *Numerical Heat Transfer, Part B*, 32:369–384, 1997.

[46] S. Muzaferija and M. Perić. Computation of free-surface flows using interface-tracking and interface-capturing methods. In O. Mahrenholtz and M Markiewicz, editors, *Nonlinear Water Wave Interaction*. Computational Mechanics Publications, 1998.

[47] B. Niceno. *EasyMesh 1.4*. World Wide Web, http: //www-dinma.univ.trieste.it /~nirftc /research /easymesh.

[48] D. Pan and J.-C. Cheng. Upwind finite-volume Navier-Stokes computations on unstructured triangular meshes. *AIAA Journal*, 31:1618–1625, 1993.

[49] D. Pan, Y.-S. Yang, and C.-H. Chang. Computation of internal flow with free surfaces using artificial compressibility. *Numerical Heat Transfer, Part B*, 33:119–134, 1998.

[50] S. V. Patankar. *Numerical Heat Transfer and Fluid Flow*. Hemisphere Publishing Corporation, 1980.

[51] T. J. Pedley and K. D. Stephanoff. Flow along a channel with a time-dependent indentation in one wall: the generation of vorticity waves. *Journal of Fluid Mechanics*, 160:337–367, 1985.

[52] D. H. Peregrine. Breaking waves on beaches. *Annual Review of Fluid Dynamics*, 15:149–178, 1983.

[53] M. Perić, R. Kessler, and G. Scheuerer. Comparison of finite-volume numerical methods with staggered and colocated grids. *Computers & Fluids*, 16:389–403, 1988.

[54] G. D. Raithby and G. E. Schneider. Elliptic systems: finite-difference method II. In W. J. Mincowycz, E. M. Sparrow, G. E. Schneider, and R. H. Pletcher, editors, *Handbook of Numerical Heat Transfer*. John Wiley & Sons, 1988.

[55] G. D. Raithby, W.-X. Xu, and G. D. Stubley. Prediction of incompressible free surface flows with an element-based finite volume method. *Computational Fluid Dynamics Journal*, 4:353–371, 1995.

[56] M. E. Ralph and T. J. Pedley. Flow in a channel with a moving indentation. *Journal of Fluid Mechanics*, 190:87–112, 1988.

[57] B. Ramaswamy and M. Kawahara. Lagrangian finite element analysis applied to viscous free surface flow. *International Journal for Numerical Methods in Fluids*, 7:953–984, 1987.

[58] M. Raw. Robustness of coupled algebraic multigrid for the Navier-Stokes equations. *AIAA Paper 96-0297*, 1996.

[59] C. M. Rhie and W. L. Chow. Numerical study of the turbulent flow past an airfoid with trailing edge separation. *AIAA Journal*, 21:1525–1532, 1983.

[60] P. L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43:357–372, 1981.

[61] G. E. Schneider and M. J. Raw. Control volume finite-element method for heat transfer and fluid flow using colocated variables — 1. Computational procedure. *Numerical Heat Transfer*, 11:363–390, 1987.

[62] V. Selmin. The node-centred finite volume approach: Bridge between finite differences and finite elements. *Computer Methods in Applied Mechanics and Engineering*, 102:107–138, 1993.

[63] J. L. Steger and R. F. Warming. Flux vector splitting of the inviscid gasdynamic equations with application to finite-difference methods. *Journal of Computational Physics*, 40:263–293, 1981.

[64] P. K. Sweby. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM Journal of Numerical Analysis*, 21:995–1011, 1984.

[65] T. E. Tezduyar, M. Behr, and J. Liou. A new strategy for finite element computations involving moving boundaries and interfaces — The deforming-spatial-domain/space-time procedure: II. computation of free-surface flows, two-liquid flows, and flows with drifting cylinders. *Computer Methods in Applied Mechanics and Engineering*, 94:353–371, 1992.

[66] J. L. Thé, G. D. Raithby, and G. D. Stubley. Surface-adaptive finite volume method for solving free surface flows. *Numerical Heat Transfer, Part B*, 26:367–380, 1994.

[67] P. D. Thomas and C. K. Lombard. Geometric conservation law and its application to flow computations on moving grids. *AIAA Journal*, 17:1030–1037, 1979.

[68] W. Tsai and D. K. P. Yue. Computation of nonlinear free surface flows. *Annual Review of Fluid Dynamics*, 28:249–278, 1996.

[69] B. van Leer. Towards the ultimate conservative difference scheme. V. A second order sequel to Godunov's method. *Journal of Computational Physics*, 32:101–136, 1979.

[70] B. van Leer. Flux-vector splitting for the Euler equations. *Lecture Notes in Physics*, 270:507–512, 1982.

[71] V. Venkatakrishnan. Convergence to steady state solutions of the Euler equations on unstructured grids with limiters. *Journal of Computational Physics*, 118:120–130, 1995.

[72] V. Venkatakrishnan. Perspective on unstructured grid flow solvers. *AIAA Journal*, 34:533–547, 1996.

[73] J. M. Weiss and W. A. Smith. Preconditioning applied to variable and constant density flows. *AIAA Journal*, 33:2050–2057, 1995.

[74] P. T. Williams and A. J. Baker. Incompressible computational fluid dynamics and the continuity constraint method for the three-dimensional Navier-Stokes equations. *Numerical Heat Transfer, Part B*, 29:137–273, 1996.

[75] H. Zhang, M. Reggio, J. Y. Trépanier, and R. Camarero. Discrete form of the GCL for moving meshes and its implementation in CFD schemes. *Computers & Fluids*, 22:9–23, 1993.

[76] X. D. Zhang, J.-Y. Trépanier, and R. Camarero. A space-time reconstruction algorithm for steady and unsteady Euler equations. *AIAA Paper 96-0529*, 1996.