CHAPTER 3.

CHAPTER 3.

The SWIFT Code

SWIFT - **S**tratified **WI**nd **F**low over **T**opography - is a finite-volume, incompressible flow solver developed by this author at the University of Surrey from an original two-dimensional laminar code. The acronymic title is somewhat restrictive. During the course of the research described here the original code has been entirely rewritten. Major additions include the capacity to calculate three-dimensional flows, the insertion of various turbulence models and, most significantly, the incorporation of a non-orthogonal curvilinear coordinate geometry, allowing the computation of flow over arbitrary (smooth) topography as well as rectangular blockages. Some of the numerical procedures have also been enhanced, most notably through the inclusion of techniques for accelerating the solution of the pressure equation. To compute dispersion of contaminants a scalar equation has been added, with the option of imbedding a region of analytically-specified concentrations to minimise numerical diffusion from sources considerably smaller than the mesh size. Finally, gridding routines with the capability of generating a smooth surface from arbitrary terrain data and a graphical postprocessor have been written to complete a suite of programs for handling realistic topography.

A summary of the code is given in Table 3.1. The two principle aspects of the fluid dynamical problem are the specification of the model equations and the numerical procedures for solving them. These will be dealt with separately in the following Sections.

3.1 Governing Equations

3.1.1 Mean Flow

The mean flow equations to be solved are those for mass conservation (continuity), momentum (Navier-Stokes) and the transport of passive/non-passive scalars. In incompressible flow, with the Boussinesq approximation and subject to buoyancy and Coriolis forces, these take the form

Governing equations:	Mass, momentum, passive and non-passive scalars. Turbulent kinetic energy and dissipation rate.
Approximations:	Incompressible, Boussinesq.
Main options:	1/2/3-dimensional. Cartesian/curvilinear geometry. Steady/time-dependent. Neutral/stably stratified. Laminar/turbulent (*k* and *k*-ε models).
General form of discretisation:	Finite volume.
Control volume geometry:	Staggered grid. Cell-centred control volumes.
Advection schemes:	Upwind, Van Leer, QUICK.
Time-dependence scheme:	Backward Euler.
Equation coupling algorithms:	SIMPLE, SIMPLEC, SIMPLER, SIMPLEX.
Matrix solution method:	Line-iteration procedures; optional block-correction and anticipated-correction acceleration techniques

Table 3.1: Summary of the SWIFT code.

$$\frac{\partial U_j}{\partial x_j} = 0$$

$$\frac{DU_i}{Dt} = -\frac{1}{\rho_0}\frac{\partial}{\partial x_i}(P-P_a) + \frac{(\rho-\rho_a)}{\rho_0}g_i - 2\in_{ijk}\Omega_j U_k + \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial U_i}{\partial x_j}+\frac{\partial U_j}{\partial x_i}\right) - \overline{u_i' u_j'}\right] \qquad (3.1)$$

$$\frac{D\Theta}{Dt} = \frac{\partial}{\partial x_j}\left(\kappa_\theta\frac{\partial\Theta}{\partial x_j} - \overline{\theta' u_j'}\right)$$

Here, as throughout this Thesis, upper case will be used for mean flow variables and primed lower case for turbulent fluctuations. $\vec{u}=(u,v,w)$, p, ρ and θ are the velocity, pressure, density and (passive or non-passive) scalar respectively. Mean pressure and density are expressed as differences from a reference state (subscript a) in hydrostatic equilibrium. ν is the kinematic viscosity and κ_θ the molecular diffusivity of scalar θ. Buoyancy and Coriolis forces are included in the momentum equation, with gravitational acceleration $\vec{g}=(0,0,-g)$ and angular velocity of the rotating frame $\vec{\Omega}=(0,0,\Omega)$ aligned with the vertical (z) coordinate. $\frac{D}{Dt}=\frac{\partial}{\partial t}+\vec{U}\bullet\nabla$ is the derivative following the mean flow.

The incompressibility approximation requires that

$$\left|\frac{1}{\rho}\frac{D\rho}{Dt}\right| \ll \frac{U_0}{L} \tag{3.2}$$

where U_0 and L are typical velocity and length scales. This condition is generally satisfied except in high-speed gas flows (large pressure variations) or highly convecting flows (large temperature variations). Whilst incompressibility does not imply uniform density, it is common in environmental flow modelling to adopt the *Boussinesq approximation* (Turner, 1979) that density variations may be neglected in the advection (and molecular transport) terms, but retained in the buoyancy term, where it is the deviation from the hydrostatic balance which drives the flow; ie,

$$\rho\frac{D}{Dt} = \rho_0(1+\frac{\Delta\rho}{\rho_0})\frac{D}{Dt} \approx \rho_0\frac{D}{Dt} \tag{3.3}$$

$$-\frac{\partial p}{\partial x_i}+\rho g_i = -\frac{\partial}{\partial x_i}(p-p_a)+(\rho-\rho_a)g_i \tag{3.4}$$

For atmospheric flows the Boussinesq approximation is justified when vertical displacements are much smaller than the scale height of the atmosphere, RT/g (≈8km).

In the lower levels of the atmosphere density variations are largely determined by variations in potential temperature, $\theta=T(P/P_0)^{-R/c_p}$, the temperature which would be obtained if an air parcel was brought adiabatically to some reference pressure P_0. For flows in hydrostatic equilibrium,

$$\frac{T}{\theta}\frac{\partial\theta}{\partial z} = \frac{\partial T}{\partial z} + \frac{g}{c_p} \tag{3.5}$$

so that, up to the Boussinesq approximation, upstream values of θ may be determined by

$$\theta = T+\frac{g}{c_p}z \tag{3.6}$$

Note that the potential temperature equation is essentially the entropy equation in disguise

($s=c_p\ln(\theta)$) with sources of heat (viscous dissipation of kinetic energy, latent heat of phase changes, radiation divergence, etc) neglected in comparison with the transport terms. Under the assumptions implicit in the incompressible and Boussinesq approximations, density variations are primarily determined by temperature differences:

$$\frac{\rho-\rho_a}{\rho_0} = -\alpha(\Theta-\Theta_0) \tag{3.7}$$

where $\alpha \equiv -\frac{1}{\rho_0}\left(\frac{\partial\rho}{\partial\theta}\right)_p$ is the coefficient of expansion; (α=1/T for an ideal gas).

Formally, a similar variation of density with a non-passive scalar (in this case, salinity) can be observed in the ocean or in a laboratory tank. However, in the salinity-stratified case the molecular Prandtl number $Pr=\nu/\kappa_\theta$ is about 1000 times larger than in air and, unlike heat, salt does not diffuse through solid boundaries.

3.1.2 Turbulence Models

Turbulence modelling merits its own Chapter in this Thesis and we shall have more to say about various schemes in Chapter 4, before concentrating on atmospheric boundary-layer applications in Chapter 5. In this Section we shall summarise very briefly the turbulence models encoded in SWIFT.

By analogy with the molecular diffusion process, turbulent fluxes are prescribed by an eddy-viscosity/gradient-transport model:

$$\begin{aligned} -(\overline{u_i'u_j'}-\frac{2}{3}k\delta_{ij}) &= \nu_t\left[\frac{\partial U_i}{\partial x_j}+\frac{\partial U_j}{\partial x_i}-\frac{2}{3}(\nabla.\vec{U})\delta_{ij}\right] \\ -\overline{\theta'u_j'} &= \frac{\nu_t}{\sigma_\theta}\frac{\partial\Theta}{\partial x_j} \end{aligned} \tag{3.8}$$

where ν_t is the product of a turbulent velocity scale u_0 and mixing length l_m. (The divergence $\nabla.\vec{U}$ vanishes in incompressible flow and this will be assumed hereafter). Having assumed an isotropic eddy-diffusivity coefficient the most appropriate turbulent velocity scale is

proportional to the square root of the turbulent kinetic energy; thus

$$u_0 = C_\mu^{1/4} k^{1/2} \tag{3.9}$$

In comparatively simple geometries the mixing length, l_m, may be specified directly. In the one-equation (k) model we assume

$$l_m = \frac{\kappa z}{1 + \kappa z / l_{max}} \tag{3.10}$$

where z is the distance to the nearest surface and l_{max} is some maximum mixing length. For more complex flows, however, l_m is better related to the scales of the turbulent motion through the dissipation rate ε, via

$$l_m = l_\epsilon \equiv \frac{u_0^3}{\epsilon} \tag{3.11}$$

Combining (3.9) and (3.11) we obtain the standard form of the k-ε model:

$$\nu_t = C_\mu \frac{k^2}{\epsilon} \tag{3.12}$$

k and ε are determined from modelled transport equations. In the standard model these are:

$$\begin{aligned} \frac{Dk}{Dt} &= \frac{\partial}{\partial x_j}\left[(\nu + \frac{\nu_t}{\sigma_k})\frac{\partial k}{\partial x_j}\right] + \Pi - \epsilon \\ \frac{D\epsilon}{Dt} &= \frac{\partial}{\partial x_j}\left[(\nu + \frac{\nu_t}{\sigma_\epsilon})\frac{\partial \epsilon}{\partial x_j}\right] + (C_{\epsilon 1}\Pi - C_{\epsilon 2}\epsilon)\frac{1}{\tau_\epsilon}, \qquad \tau_\epsilon = \frac{k}{\epsilon} \end{aligned} \tag{3.13}$$

where Π is the overall production rate of turbulent kinetic energy, the sum of terms P and G due to shear and buoyancy forces respectively:

$$\begin{aligned} P &\equiv -\overline{u_i' u_j'}\frac{\partial U_i}{\partial x_j} = \nu_t\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right)\frac{\partial U_i}{\partial x_j} \\ G &\equiv \alpha g \overline{\theta' w'} = -\alpha g \frac{\nu_t}{\sigma_\theta}\frac{\partial \Theta}{\partial z} \end{aligned} \tag{3.14}$$

(It is unfortunate that P is used widely for both mean pressure and production of turbulent

kinetic energy. In practice, the distinction is usually obvious from the context. F is used elsewhere for the production of turbulent kinetic energy by body forces, but we shall retain the more familiar G when this is due to buoyancy alone. As will be shown in Chapter 4 the fluctuating Coriolis force makes no *net* contribution to the turbulent kinetic energy budget.)

Model constants (C_μ,$C_{\varepsilon 1}$,$C_{\varepsilon 2}$,σ_θ,σ_k,σ_ε) may be set individually. The values used are discussed in Chapter 4.

In the one-equation model, with l_m specified directly, there is only a single transport equation for k: the dissipation rate ε is determined algebraically by inverting (3.11).

Various modifications to the "standard" model have been tested - see Chapters 4 and 5. For completeness we list the variations without explanation here.

- *Streamline curvature modification:*

$$C_\mu \rightarrow \frac{C_{\mu 0}}{1+4\phi^2 \frac{k^2}{\varepsilon^2}\frac{\partial U_s}{\partial n}\frac{U_s}{R_c}} \tag{3.15}$$

- *Preferential response of dissipation to normal strains:*

$$P_\varepsilon = C_{\varepsilon 1}\frac{\Pi}{\tau_\varepsilon} \rightarrow [C'_{\varepsilon 1}\Pi - C''_{\varepsilon 1}\nu_t(2S_{ns})^2]\frac{1}{\tau_\varepsilon} \tag{3.16}$$

- *Limited-length-scale modification:*

$$P_\varepsilon = C_{\varepsilon 1}\frac{\Pi}{\tau_\varepsilon} \rightarrow \left[C_{\varepsilon 1} + (C_{\varepsilon 2}-C_{\varepsilon 1})\frac{l_m}{l_{max}}\right]\frac{\Pi}{\tau_\varepsilon} \tag{3.17}$$

3.1.3 Non-Dimensionalisation

It is advantageous to non-dimensionalise all variables within SWIFT. The incompressible flow equations become:

$$
\begin{aligned}
\frac{\partial \hat{u}_j}{\partial \hat{x}_j} &= 0 \\
\frac{D\hat{u}_i}{D\hat{t}} &= -\frac{\partial \hat{p}}{\partial \hat{x}_i} + \frac{1}{Fr^2}\hat{\theta}\delta_{i3} + \frac{1}{Ro}\in_{ij3}\hat{u}_j + \frac{\partial}{\partial \hat{x}_j}\left[\frac{1}{Re}\left(\frac{\partial \hat{u}_i}{\partial \hat{x}_j}+\frac{\partial \hat{u}_j}{\partial \hat{x}_i}\right)\right] \\
\frac{D\hat{\theta}}{D\hat{t}} &= \frac{\partial}{\partial \hat{x}_j}\left(\frac{1}{Re.Pr}\frac{\partial \hat{\theta}}{\partial \hat{x}_j}\right)
\end{aligned}
\tag{3.18}
$$

Dimensional variables (mean+turbulent) can be recovered from their dimensionless counterparts (denoted by a hat) via

$$
\begin{aligned}
x_i &= L\hat{x}_i \\
t &= (L/U_0)\hat{t} \\
u_i &= U_0\hat{u}_i \\
p &= P_0 - \rho_0 g z + \rho_0 U_0^2 \hat{p} \\
\rho &= \rho_0 - \Delta\rho\,\hat{\theta} \\
\theta &= \theta_0 + \Delta\theta\,\hat{\theta}
\end{aligned}
\tag{3.19}
$$

L, U_0 and ρ_0 are length, velocity and density scales, with $\Delta\rho=\alpha\rho_0\Delta\theta$ a typical density variation. The flow dynamics depend on the boundary conditions and the non-dimensional combinations

$$
\begin{aligned}
Re &= \frac{U_0 L}{\nu} && \text{(Reynolds number)} \\
Pr &= \frac{\nu}{\kappa_\theta} && \text{(Prandtl number)} \\
Fr &= \frac{U_0}{\sqrt{\frac{\Delta\rho}{\rho_0} g L}} && \text{(Froude number)} \\
Ro &= \frac{U_0}{fL} && \text{(Rossby number)}
\end{aligned}
\tag{3.20}
$$

3.2 Numerical Methods

3.2.1 Working Variables and Form of Equations

SWIFT is an incompressible flow solver employing *primitive* working variables (velocity and pressure), rather than *derived* quantities (such as stream-function or vorticity). The obvious advantages are of generality and physical interpretation, but there are additional benefits associated with, for example, the ease of imposing non-slip boundary conditions. The general form of discretisation is the *finite-volume* method: conservation equations are expressed in integral form with fluxes across cell faces approximated directly. Again, this has the benefit of immediate physical interpretation, a feature which is often obscured in *finite-difference* methods (differential flow equations discretised directly) or *finite-element* methods (governing equations rephrased as variational principles).

3.2.2 Control-Volume Geometry

SWIFT employs a *staggered-grid* arrangement (Harlow and Welch, 1965) with *cartesian* velocity components $\vec{U}=(U,V,W)$ stored half-way between the pressure nodes which drive them. Scalar transport variables are stored at the pressure nodes, whilst derived variables, such as shear stress or stream-function (both related to velocity gradients), are defined at their "natural" storage locations (Figure 3.1). By convention, the velocity components are indexed by the scalar node to which they point.

The computational domain is subdivided into cells or *control volumes* over which the flow equations are formulated as integral conservation laws. Control-volume faces coincide with coordinate planes half-way between adjacent nodes (Figure 3.2). With the staggered-grid arrangement separate control volumes must be constructed for transported scalars and for each velocity component.

For a cartesian grid, the advantages of the staggered over the *co-located* arrangement are that the former avoids odd-even pressure-node decoupling (pressure gradient $\partial P/\partial x^{\alpha}$ acts as a

source for the momentum component U^{α}), velocity components are stored directly on scalar control volume faces where the advective flux is required (for both continuity and scalar transport equations) and there is no need for pressure boundary conditions. A less immediately apparent, yet very significant, advantage is the applicability of enhanced pressure-correction algorithms such as SIMPLER and SIMPLEX, which accelerate convergence rates in flows with strong pressure-velocity coupling. The disadvantages of staggered meshes arise primarily from the geometrical complexity of specifying different control volumes for each type of storage location.

The advantages of the staggered arrangement with cartesian velocity decomposition carry over from rectangular to curvilinear meshes ... provided the distortion is not too great. If, however, the coordinate lines turn through 90° then the pressure gradient along a coordinate line and the intermediate velocity component become totally misaligned (Figure 3.3). For this reason, codes which purport to compute flow on arbitrary (structured) meshes generally use a co-located arrangement and/or a *contravariant* velocity decomposition (ie, along coordinate lines). The odd-even pressure decoupling associated with co-located pressure and velocity can be overcome by the Rhie-Chow momentum interpolation algorithm (Rhie and Chow, 1983; Majumdar et al., 1992). As we shall see, a contravariant velocity decomposition introduces additional, generally non-conservative, geometric terms. For the application to flow over smooth topography, however, the coordinate mesh is only weakly distorted and the advantages of a cartesian velocity decomposition using the staggered-storage arrangement make this the preferred option.

3.2.3 Transport Equations in Curvilinear Coordinate Systems

Extension of the finite-volume technique to a general non-orthogonal curvilinear coordinate mesh makes it necessary to choose the form of velocity decomposition. The most common types are:

- *cartesian*: components with respect to a fixed frame, treated as individual scalars;
- *contravariant*: components with respect to basis vectors along the coordinate lines;
- *covariant*: components with respect to basis vectors normal to the coordinate surfaces;

but a hybrid decomposition is possible; for example, the centre-line-oriented local-cartesian-basis system of Lien (1992) for computing flow in curved ducts.

A coordinate mapping $\{\xi^i\}$ of N-dimensional space defines two particular sets of basis vectors at any point in space (Figure 3.4):

- the *natural basis* along the coordinate lines:

$$\vec{g}_i = \frac{\partial \vec{r}}{\partial \xi^i} \tag{3.21}$$

- the *dual basis* normal to the coordinate surfaces:

$$\vec{g}^i = \nabla \xi^i \tag{3.22}$$

In general, neither set are unit vectors, but they have the orthonormal property that

$$\vec{g}_i \bullet \vec{g}^j = \delta_i^j \tag{3.23}$$

From the definition we have immediately that

$$\frac{\partial \vec{g}_i}{\partial \xi^j} = \frac{\partial \vec{g}_j}{\partial \xi^i} \tag{3.24}$$

In three dimensions the dual basis can be constructed from the natural basis via

$$\vec{g}^1 = \frac{\vec{g}_2 \wedge \vec{g}_3}{\vec{g}_1 \bullet \vec{g}_2 \wedge \vec{g}_3} \tag{3.25}$$

(with the remaining two vectors by cyclic permutation). The scalar triple products are given by

$$\begin{aligned} \vec{g}_1 \bullet \vec{g}_2 \wedge \vec{g}_3 &= \det\left(\frac{\partial x^i}{\partial \xi^j}\right) = J \\ \vec{g}^1 \bullet \vec{g}^2 \wedge \vec{g}^3 &= \det\left(\frac{\partial \xi^i}{\partial x^j}\right) = J^{-1} \end{aligned} \tag{3.26}$$

where J, the Jacobian of the transformation $\{\xi^i\}\rightarrow\{x^i\}$, constitutes a volume dilatation factor.

The *metric* tensor g_{ij} is the symmetric tensor defined by the distance function ds^2 (an invariant) such that

$$ds^2 = g_{ij}d\xi^i d\xi^j \tag{3.27}$$

The transformation is said to be *orthogonal* if g_{ij} has no non-zero off-diagonal components, but such transformations are an over-restrictive subclass of the available systems. In terms of an underlying cartesian system $\{x^i\}$ we have

$$ds^2 = dx^k dx^k = \frac{\partial x^k}{\partial \xi^i}\frac{\partial x^k}{\partial \xi^j}d\xi^i d\xi^j \tag{3.28}$$

and, since $d\xi^i d\xi^j$ is an arbitrary symmetric tensor, it follows that g_{ij} is given in terms of the underlying cartesian basis by

$$g_{ij} = \frac{\partial x^k}{\partial \xi^i}\frac{\partial x^k}{\partial \xi^j} \tag{3.29}$$

Using the laws of linear algebra ("determinant of a product is a product of the determinants") we have that

$$g \equiv \det(g_{ij}) = J^2 \tag{3.30}$$

g_{ij} are the components of a tensor with respect to the basis $\vec{g}^i\otimes\vec{g}^j$. ($\otimes$ denotes a tensor product.) The components with respect to its dual basis $\vec{g}_i\otimes\vec{g}_j$ are

$$g^{ij} = \frac{\partial \xi^i}{\partial x^k}\frac{\partial \xi^j}{\partial x^k} \tag{3.31}$$

The sets of components have the inverse property that

$$g_{ik}g^{kj} = \delta_i^j \tag{3.32}$$

The metric components may also be deduced from the basis vectors, via

$$g_{ij} = \vec{g}_i \bullet \vec{g}_j$$
$$g^{ij} = \vec{g}^i \bullet \vec{g}^j \qquad (3.33)$$

When the finite-volume mesh is given directly in terms of the coordinates of nodes or cell vertices, rather than by an analytical transformation formula, this is the appropriate way of evaluating the metric tensor (since the natural basis vectors connect adjacent nodes/vertices). The volume dilatation factor $J=\sqrt{g}$ can be determined from the physical cell volume.

With a cartesian system (basis vectors $\vec{e}_i$) contravariant and covariant tensor components are equal (since the basis vectors are the same in each case). Denoting components in the generalised system by a tilde we have, in terms of each basis,

$$\vec{U} = U^i \vec{e}_i = \tilde{U}^i \vec{g}_i = \tilde{U}_i \vec{g}^i \qquad (3.34)$$

from which we may deduce the *contravariant* and *covariant* components $\tilde{U}^i$ and $\tilde{U}_i$ respectively:

$$\tilde{U}^i = \frac{\partial \xi^i}{\partial x^j} U^j \qquad \tilde{U}_i = \frac{\partial x^j}{\partial \xi^i} U_j \qquad (3.35)$$

whilst the two sets of components can be related by raising and lowering indices using the metric tensor:

$$\tilde{U}^i = g^{ij} \tilde{U}_j \qquad \tilde{U}_i = g_{ij} \tilde{U}^j \qquad (3.36)$$

In a similar manner, we have the transformation laws for second rank tensors; eg, for contravariant components $\tilde{T}^{ij}$ (with respect to basis $\vec{g}_i \otimes \vec{g}_j$):

$$\tilde{T}^{ij} = \frac{\partial \xi^i}{\partial x^k} \frac{\partial \xi^j}{\partial x^l} T^{kl} \qquad (3.37)$$

and, again, individual indices may be raised or lowered by contraction with the metric tensor.

The difficulty with a contravariant (or, less commonly, covariant) decomposition is that the basis vectors are not fixed, but vary with position in space - the curvature property.

Christoffel symbols (of the *second kind*) Γ_{ij}^{k} are defined by

$$\frac{\partial \vec{g}_i}{\partial \xi^j} = \Gamma_{ij}^{k} \vec{g}_k \tag{3.38}$$

By differentiating the determinant with respect to each of its components, it can readily be shown (Spiegel, 1974) that

$$\Gamma_{ij}^{k} = \Gamma_{ji}^{k} = \frac{1}{2} g^{kl} \left(\frac{\partial g_{li}}{\partial \xi^j} + \frac{\partial g_{lj}}{\partial \xi^i} - \frac{\partial g_{ij}}{\partial \xi^k} \right) \tag{3.39}$$

and that the contracted form

$$\Gamma_{ij}^{j} = \frac{1}{\sqrt{g}} \frac{\partial}{\partial \xi^i} (\sqrt{g}) \tag{3.40}$$

Having established how the basis vectors vary in space we may now differentiate vectors and second-rank tensors according to the product rule:

$$\begin{aligned}
\nabla_k \vec{V} \equiv \frac{\partial}{\partial \xi^k} (\tilde{V}^i \vec{g}_i) &= \frac{\partial \tilde{V}^i}{\partial \xi^k} \vec{g}_i + \tilde{V}^i \Gamma_{ik}^{l} \vec{g}_l \\
&= \left(\frac{\partial \tilde{V}^i}{\partial \xi^k} + \Gamma_{lk}^{i} \tilde{V}^l \right) \vec{g}_i \\
\nabla_k \vec{\tilde{T}} \equiv \frac{\partial}{\partial \xi^k} (\tilde{T}^{ij} \vec{g}_i \otimes \vec{g}_j) &= \frac{\partial \tilde{T}^{ij}}{\partial \xi^k} \vec{g}_i \otimes \vec{g}_j + \tilde{T}^{ij} \Gamma_{ik}^{l} \vec{g}_l \otimes \vec{g}_j + \tilde{T}^{ij} \Gamma_{kj}^{l} \vec{g}_i \otimes \vec{g}_l \\
&= \left(\frac{\partial \tilde{T}^{ij}}{\partial \xi^k} + \Gamma_{kl}^{i} \tilde{T}^{lj} + \Gamma_{kl}^{j} \tilde{T}^{il} \right) \vec{g}_l \otimes \vec{g}_j
\end{aligned} \tag{3.41}$$

from which we may define *covariant derivatives* whose components are

$$\begin{aligned}
\tilde{V}_{;k}^{i} \equiv \nabla_k \tilde{V}^i &= \frac{\partial \tilde{V}^i}{\partial \xi^k} + \Gamma_{lk}^{i} \tilde{V}^l \\
\tilde{T}_{;k}^{ij} \equiv \nabla_k \tilde{T}^{ij} &= \frac{\partial \tilde{T}^{ij}}{\partial \xi^k} + \Gamma_{kl}^{i} \tilde{T}^{lj} + \Gamma_{kl}^{j} \tilde{T}^{il}
\end{aligned} \tag{3.42}$$

The *divergence* of a vector or higher-rank tensor may be evaluated by contracting on one

index; thus,

$$\begin{aligned} div\vec{V} \equiv \nabla_i \tilde{V}^i &= \frac{\partial \tilde{V}^i}{\partial \xi^i} + \Gamma^i_{li}\tilde{V}^l \\ &= \frac{1}{\sqrt{g}}\frac{\partial}{\partial \xi^i}(\sqrt{g}\tilde{V}^i) \\ (div\vec{\vec{T}})^j \equiv \nabla_i \tilde{T}^{ij} &= \frac{\partial \tilde{T}^{ij}}{\partial \xi^i} + \Gamma^i_{il}\tilde{T}^{lj} + \Gamma^j_{il}\tilde{T}^{il} \\ &= \frac{1}{\sqrt{g}}\frac{\partial}{\partial \xi^i}(\sqrt{g}\tilde{T}^{ij}) + \Gamma^j_{il}\tilde{T}^{il} \end{aligned} \tag{3.43}$$

where we have used the result (3.40).

The conservation equation for a *scalar* may be written in coordinate-independent form

$$\frac{\partial}{\partial t}(\rho\phi) + \nabla\bullet(\rho\vec{U}\phi + \vec{F}^{(\phi)}) = S^{(\phi)} \tag{3.44}$$

where $\vec{F}^{(\phi)}$ is the flux vector and $S^{(\phi)}$ the source density. In terms of contravariant flux components this may be written

$$\frac{\partial}{\partial t}(\rho\phi) + \frac{1}{\sqrt{g}}\frac{\partial}{\partial \xi^j}[\sqrt{g}(\rho\tilde{U}^j\phi + \tilde{F}^j)] = S^{(\phi)} \tag{3.45}$$

In the absence of sources (including temporal changes) this takes the *strong conservative* form

$$\frac{\partial}{\partial \xi^j}[\sqrt{g}(\rho\tilde{U}^j\phi + \tilde{F}^j)] = 0 \tag{3.46}$$

The *vector* momentum equation may be written as a tensor equation:

$$\frac{\partial}{\partial t}(\rho\vec{U}) + \nabla\bullet(\rho\vec{U}\otimes\vec{U} + \vec{\vec{T}}) = \vec{S}^{(\vec{U})} \tag{3.47}$$

Componentwise, this becomes (in the absence of sources)

$$\frac{\partial}{\partial \xi^j}[\sqrt{g}(\rho \tilde{U}^i \tilde{U}^j + \tilde{T}^{ij})] + \sqrt{g}\Gamma^i_{kj}(\rho \tilde{U}^k \tilde{U}^j + \tilde{T}^{kj}) = 0 \tag{3.48}$$

It is the presence of the non-conservative terms arising from the rotation of the basis vectors which creates difficulties with contravariant velocity decomposition, since their discretised form will not, in general, cancel on integration over the whole domain. The importance of retaining the governing equations in a strong conservative form generally outweighs any benefits arising from the use of coordinate-wise (contravariant) velocity components (for which the rationale of staggering would be more appropriate). For this reason SWIFT, in common with the vast majority of other primitive-variable solvers, treats each *cartesian* velocity component as an individual scalar variable.

3.2.4 Discretisation of the Scalar Transport Equation

To recap: the prototype scalar transport equation for a flow variable ϕ can be written in conservation form as

$$\frac{\partial}{\partial t}(\rho\phi) + \frac{\partial}{\partial x^i}(\rho U^i \phi + F^i) = \rho s \tag{3.49}$$

where U^i are the cartesian velocity components, F^i is the (diffusive) flux vector and s is the source (per unit mass). In many cases diffusion can be described by a gradient transport hypothesis:

$$F^i = -\Gamma^{ij}\frac{\partial \phi}{\partial x^j} \tag{3.50}$$

where Γ^{ij} is the diffusivity tensor. In complex geometries it is often desirable to work in a curvilinear coordinate system $\{\xi^i\}$. A general result from tensor calculus is

$$\nabla \bullet \vec{V} \equiv \frac{1}{\sqrt{g}}\frac{\partial}{\partial \xi^i}(\sqrt{g}\tilde{V}^i) \tag{3.51}$$

where the metric tensor g_{ij} is given in terms of a base cartesian system by

$$g_{ij} = \frac{\partial x^k}{\partial \xi^i}\frac{\partial x^k}{\partial \xi^j} , \qquad g^{ij} = \frac{\partial \xi^i}{\partial x^k}\frac{\partial \xi^j}{\partial x^k} \tag{3.52}$$

and $g=\det(g_{ij})$. Contravariant components of vectors and second rank tensors (typified by velocity and diffusivity respectively) transform according to

$$\begin{aligned} \tilde{U}^i &= \frac{\partial \xi^i}{\partial x^j} U^j \\ \tilde{\Gamma}^{ij} &= \frac{\partial \xi^i}{\partial x^k}\frac{\partial \xi^j}{\partial x^l}\Gamma^{kl} \end{aligned} \tag{3.53}$$

where a tilde denotes a tensorial component in the curvilinear system.

The scalar transport equation (with gradient diffusion) can then be written:

$$\frac{\partial}{\partial t}(\rho\phi) + \frac{1}{\sqrt{g}}\frac{\partial}{\partial \xi^i}(\sqrt{g}\rho\tilde{U}^i\phi - \sqrt{g}\tilde{\Gamma}^{ij}\frac{\partial \phi}{\partial \xi^j}) = \rho s \tag{3.54}$$

If diffusion is isotropic then $\Gamma^{ij}=\Gamma\delta^{ij}$ in the cartesian system, and, more generally,

$$\tilde{\Gamma}^{ij} = \Gamma g^{ij} \tag{3.55}$$

An equivalent (and, actually, more fundamental) integral equation can be derived by integrating (3.49) over an arbitrary fixed volume V:

$$\frac{d}{dt}\int_V \rho\phi \, dV + \int_{\partial V}(\rho U^i\phi - \Gamma^{ij}\frac{\partial \phi}{\partial x^j}) \, dA_i = \int_V \rho s \, dV \tag{3.56}$$

Here, we integrate (3.54) over a single curvilinear control volume with *physical* volume $\Delta V=\sqrt{g}\,\Delta V_\xi \equiv \sqrt{g}\,\Delta\xi\Delta\eta\Delta\zeta$ and obtain the discrete form

$$\begin{aligned} \frac{d}{dt}(\rho_p\phi_p)\sqrt{g}\Delta V_\xi + \sum_\alpha \left[\sqrt{g}\rho\tilde{U}^\alpha\phi - \sqrt{g}\tilde{\Gamma}^{\alpha\alpha}\frac{\partial \phi}{\partial \xi^\alpha}\right]_{\alpha-}^{\alpha+} A_\alpha &= \rho_p s_p \sqrt{g}\Delta V_\xi \\ &+ \sum_\alpha \sum_{\beta\neq\alpha} \left[\sqrt{g}\tilde{\Gamma}^{\alpha\beta}\frac{\partial \phi}{\partial \xi^\beta}\right]_{\alpha-}^{\alpha+} A_\alpha \end{aligned} \tag{3.57}$$

where subscript p denotes a nodal value at the control volume centre, $A_\xi=\Delta\eta\Delta\zeta$ for example,

and the square brackets indicate the *difference* in forward fluxes through cell faces in the $\alpha=\xi,\eta,\zeta$ directions. (Here, as in the remainder of this Thesis, there is no implied summation over repeated greek indices).

ϕ-derivatives are obtained by central differencing. Thus, on the positive ξ-face, for example, the derivative along the coordinate line is given by

$$\left(\frac{\partial \phi}{\partial \xi}\right)_{p+\frac{1}{2}\xi} = \frac{\phi_{p+\xi}-\phi_p}{\Delta \xi} \tag{3.58}$$

whilst, in the tangential direction, average values are first calculated at the cell corners; for example,

$$\left(\frac{\partial \phi}{\partial \eta}\right)_{p+½\xi} = \frac{\phi_{p+½\xi+½\eta}-\phi_{p+½\xi-½\eta}}{\Delta \eta} \tag{3.59}$$

Note that diffusion terms involving tangential derivatives have been transferred to the source side of equation (3.57).

The major advection discretisation problem amounts to specifying the value ϕ on the control volume face, half-way between adjacent nodes. We shall return to the various advection schemes in a later Section. For the moment we note that most schemes tacitly subtract from (3.44) or (3.57) (ϕ_p times) the mass conservation equation

$$\frac{\partial \rho}{\partial t} + \nabla\bullet(\rho \vec{U}) = 0 \tag{3.60}$$

or, in its discrete integral form,

$$\frac{d\rho_p}{dt}\sqrt{g}\Delta V_\xi + \sum_\alpha \left[\sqrt{g}\rho \tilde{U}^\alpha\right]_{\alpha-}^{\alpha+} A_\alpha = 0 \tag{3.61}$$

In the incompressible case this amounts to a velocity field with zero net divergence. In solving a complete system of flow equations, mass conservation is ensured in the final, converged, solution. If, however, the flow field is subsequently interpolated onto a new grid - for example to calculate the dispersion of a passive contaminant from a point source - the

interpolated flow field *may not* satisfy mass conservation exactly. (In two dimensions continuity may be ensured indirectly by interpolating for the stream-function and then deriving the velocity components by differentiation. No such process is possible in three dimensions.) The discretisation of the scalar equation must be performed carefully to ensure overall conservation of material.

The final form of the scalar transport equation is

$$\rho_p \frac{d\phi_p}{dt}\sqrt{g}\Delta V_\xi + \sum_\alpha \left[G_\alpha(\phi - \phi_p) + T_\alpha(\phi_p - \phi_e)\right]_{\alpha-}^{\alpha+} = \rho_p S_p \sqrt{g}\Delta V_\xi + \sum_\alpha \sum_{\beta\neq\alpha} \left[\sqrt{g}\tilde{\Gamma}^{\alpha\beta}\frac{\partial\phi}{\partial\xi^\beta}\right]_{\alpha-}^{\alpha+} A_\alpha \tag{3.62}$$

where subscript *e* denotes a value at the "external" node; ie, along the outward normal. The mass flux and diffusion transport coefficients are

$$\begin{aligned} G_\alpha &= \sqrt{g}\rho\tilde{U}^\alpha A_\alpha \\ T_\alpha &= \frac{\sqrt{g}\tilde{\Gamma}^{\alpha\alpha}A_\alpha}{\Delta\xi_\alpha} \end{aligned} \tag{3.63}$$

Since $\sqrt{g}$ is the volume dilatation factor, $\sqrt{g}V_\xi$ is the *physical* cell volume. Similarly, G_α is the actual mass flux across a cell face - a property which allows a direct check of coding.

Normalised Matrix Equations

When ϕ_f has been specified in terms of nodal values, (3.62) can be put in the form

$$\sum_d A_{p+d}(\phi_p - \phi_{p+d}) = B_p + S_p\phi_p \tag{3.64}$$

where the $\{A_{p+d}\}$ are a set of connecting coefficients and the RHS is a linearisation of the source term with $S_p \leq 0$. (The non-standard, but eminently workable, notation uses subscript $p+d$ to describe the node in direction d from node p, or the matrix coefficient connecting p

with that node. *d* may be one of $\{\pm\xi,\pm\eta,\pm\zeta\}$. The notation is particularly beneficial in more than two dimensions when the conventional N,E,S,W compass-point notation becomes inadequate.) Transferring the implicit source term to the LHS and dividing by $\sum_d A_{p+d} - S_p$, one obtains *normalised* equations

$$\phi_p - \sum_d a_{p+d}\phi_{p+d} = b_p \tag{3.65}$$

where normalised coefficients are written in lower case:

$$a_{p+d} = \frac{A_{p+d}}{\sum A_{p+d} - S_p}, \qquad b_{p+d} = \frac{B_{p+d}}{\sum A_{p+d} - S_p} \tag{3.66}$$

The non-linearity and coupling which exists generally necessitates the *relaxation* of equations so that the value of a variable does not change too much on each iteration. Without such relaxation the increment in ϕ_p would be

$$\Delta\phi_p = \sum_d a_{p+d}\phi_p + b_p - \phi_p^* \tag{3.67}$$

where superscript * denotes a value from a previous iteration. If we permit only a fraction *r* of this increment then

$$\begin{aligned} \phi_p &= \phi_p^* + r\Delta\phi_p \\ &= r\left(\sum_d a_{p+d}\phi_{p+d} + b_p\right) + (1-r)\phi_p^* \end{aligned} \tag{3.68}$$

which can be effected by a simple change in normalised coefficients:

$$\begin{aligned} a_{p+d} &\rightarrow r\,a_{p+d} \\ b_p &\rightarrow r\,b_p + (1-r)\phi_p^* \end{aligned} \tag{3.69}$$

With the values at all grid nodes stored in a single column vector the system of equations (3.65) can be written in matrix form

$$A\vec{\Phi} = \vec{b} \tag{3.70}$$

where A is a banded matrix; (5 non-zero bands in two dimensions, 7 in three dimensions).

General Rules for Discretisation

Patankar (1980) lists four rules to be observed in a finite-volume discretisation.

(i) Consistency of fluxes for control volume faces.

(ii) Non-negative flux coefficients: $A_{p+d} \geq 0$.

(iii) $\sum_d a_{p+d} = 1$ when source term is zero. (This is automatically satisfied by (3.65)).

(iv) Negative feedback in the source term: $S_p \leq 0$.

(i) is a conservation requirement and can be utilised in efficient coding, since it should not be necessary to evaluate fluxes twice for cells sharing a common face. (ii) ensures that in a purely advection-diffusion process the increase (say) in ϕ at one node should not lead to a decrease at an adjacent node. (iii) ensures that ϕ=*constant* is a solution for the advection-diffusion problem in the absence of sources or time-dependence. (iv) promotes stability and generally enhances convergence since it makes the solution matrix more diagonally dominant when the term $S_p\phi_p$ is switched to the LHS.

It is now accepted that condition (ii) is a sufficient but not necessary condition for ϕ_p to be bounded by the values at adjacent nodes. For example, the popular QUICK scheme of Leonard (1979) introduces negative coefficients, but remains oscillation-free and bounded in smooth regions of the flow. Necessary and sufficient conditions for oscillation-free discretisation in three-point, upwind-biased schemes have been formalised in the "convection-boundedness criterion" of Gaskell and Lau (1988) to which we shall return in Section 3.2.6.

Positive coefficients remain, however, a requirement of some matrix solution algorithms, including the tri-diagonal solver used in SWIFT. For this reason it is necessary to transfer part of the advection term to the source side of the equation as an explicit or *deferred* correction.

Time Dependence

In steady-state conditions the discretised flow equations take the form

$$F(\phi_p) = 0 \tag{3.71}$$

where $F(\phi_p)=\sum_d A_{p+d}(\phi_p-\phi_{p+d})-(B_p+S_p\phi_p)$ denotes the difference between integral fluxes and sources. In the time-dependent case we have the semi-discrete equation

$$\frac{d\phi_p}{dt}\Delta M + F(\phi_p) = 0 \tag{3.72}$$

where mass element $\Delta M=\rho_p\sqrt{g}\Delta V_\xi$. Denoting successive time levels by superscripts n and n+1, and using a weighted average of flux terms at the two time levels, we have

$$\frac{\phi_p^{n+1}-\phi_p^n}{\Delta t}\Delta M + rF(\phi_p^{n+1}) + (1-r)F(\phi_p^n) = 0 \tag{3.73}$$

If r=0 we have the explicit update

$$\phi_p^{n+1} = \phi_p^n - \frac{\Delta t}{\Delta M}F(\phi_p^n) \tag{3.74}$$

for which there is a Courant-type restriction on Δt to ensure that the coefficient of ϕ_p on the RHS is positive.

If $r\neq 0$ then

$$F(\phi_p^{n+1}) = \frac{1}{r}\left[\frac{\Delta M}{\Delta t}\phi_p^n - (1-r)F(\phi_p^n)\right] - \frac{\Delta M}{r\Delta t}\phi_p^{n+1} \tag{3.75}$$

so that time-dependence can formally be accommodated in a steady-state solver by a simple modification of the source terms. In particular, for the fully-implicit, unconditionally stable backward Euler scheme (r=1) we have:

$$B_p \rightarrow B_p + \frac{\Delta M}{\Delta t}\phi_p^n , \qquad S_p \rightarrow S_p - \frac{\Delta M}{\Delta t} \tag{3.76}$$

Special Treatment of the Momentum Equation

SWIFT (in common with many other codes) treats the *cartesian* velocity components with respect to some fixed frame as individual scalars, thus enabling essentially the same discretisation scheme to be used for each component as for any scalar entity. Alternative methods are to solve for either contravariant or covariant components. In these cases the divergence of a second-rank tensor rather than a vector is required and the analogue of (3.54) requires additional transformation components (Christoffel symbols) to be included in the source term.

In many flows the most important coupling between flow variables is that between the momentum components and pressure. A pressure derivative appears in the source term; eg, for x^α-momentum:

$$\begin{aligned} -\frac{\partial P}{\partial x^\alpha}(\sqrt{g}\Delta V_\xi) &= -\sqrt{g}\frac{\partial \xi^j}{\partial x^\alpha}\frac{\partial P}{\partial \xi^j}\Delta V_\xi \\ &= -D_p^{(\alpha)}(P_p - P_{p-\alpha}) - \sum_{\beta\neq\alpha}\sqrt{g}\frac{\partial \xi^\beta}{\partial x^\alpha}\frac{\partial P}{\partial \xi^\beta}\Delta V_\xi \end{aligned} \tag{3.77}$$

(assuming the staggered-grid arrangement and indexing convention in Figure 3.1) where

$$D_p^{(\alpha)} = (\sqrt{g}\frac{\partial \xi^\alpha}{\partial x^\alpha})\frac{\Delta V_\xi}{\Delta \xi^\alpha} \tag{3.78}$$

and, for curvilinear systems, we have separated the *cartesian* pressure gradient into parts involving the gradient along the corresponding curvilinear coordinate line and parallel to the corresponding cell face. The first term is retained apart from the rest of the source. Hence, after normalising, (3.65) becomes:

$$U_p^{\alpha} - \sum_d a_{p+d}^{(\alpha)} U_{p+d}^{\alpha} = b_p^{(\alpha)} + d_p^{(\alpha)}(P_{p-\alpha} - P_p) \tag{3.79}$$

where we have made explicit the coefficients for a particular momentum component by a superscript (α).

3.2.5 The Pressure Equation

By taking the divergence of the Navier-Stokes equation a Poisson equation is obtained for pressure. A finite-volume analogue is derived by considering the integrated mass divergence from a scalar control volume.

For an incompressible fluid, $\nabla.\vec{U}=0$, or

$$\frac{\partial}{\partial \xi^i}(\sqrt{g}\tilde{U}^i) = 0 \tag{3.80}$$

Integrating over a control volume $\Delta\xi\Delta\eta\Delta\zeta$ in ξ-space we obtain the volume divergence

$$D[\vec{U}] \equiv \sum_{\alpha} \left[\sqrt{g}\tilde{U}^{\alpha}\right]_{\alpha-}^{\alpha+} A_{\alpha} = 0 \tag{3.81}$$

Rewriting (3.81) in terms of the cartesian velocity components, for which we have expressions such as (3.79), it is convenient to decompose the divergence operator into "normal" and "tangential" parts:

$$\begin{aligned} D[\vec{U}] &\equiv D_n[\vec{U}] + D_t[\vec{U}] \\ &= \sum_{\alpha}\left[\sqrt{g}\frac{\partial \xi^{\alpha}}{\partial x^{\alpha}}U^{\alpha}\right]_{\alpha-}^{\alpha+} A_{\alpha} + \sum_{\alpha}\left[\sum_{\beta\neq\alpha}\sqrt{g}\frac{\partial \xi^{\alpha}}{\partial x^{\beta}}U^{\beta}\right]_{\alpha-}^{\alpha+} A_{\alpha} \end{aligned} \tag{3.82}$$

In cartesian coordinates $D_t \equiv 0$ and $D[\vec{U}]=D_n[\vec{U}]$ is simply

$$D[\vec{U}] \equiv (U_{p+x}-U_p)A_x + (V_{p+y}-V_p)A_y + (W_{p+z}-W_p)A_z \tag{3.83}$$

(3.79) can be written as

$$U_p^{\alpha} = \hat{U}_p^{\alpha} + d_p^{(\alpha)}(P_{p-\alpha} - P_p) \tag{3.84}$$

where the *pseudovelocity,* $\hat{U}_p^{\alpha}$, is given by

$$\hat{U}_p^{\alpha} = \sum_d a_{p+d}^{(\alpha)} U_{p+d}^{\alpha} + b_p^{(\alpha)} \tag{3.85}$$

In suggestive notation,

$$\vec{U} = \hat{U} + dP \tag{3.86}$$

For an incompressible fluid we require that $D[\vec{U}]=0$ and hence

$$D_n[dP] = -D[\hat{U}] - D_t[dP] \tag{3.87}$$

Expansion of the divergence operator on the LHS yields a matrix equation for pressure:

$$\sum_d a_{p+d}^{(P)}(P_p - P_{p+d}) = -D[\hat{U}] - D_t[dP] \tag{3.88}$$

where

$$a_{p+\alpha}^{(P)} = (\sqrt{g}\frac{\partial \xi^{\alpha}}{\partial x^{\alpha}}) d_{p+\alpha}^{(\alpha)} A_{\alpha} , \qquad a_{p-\alpha}^{(P)} = (\sqrt{g}\frac{\partial \xi^{\alpha}}{\partial x^{\alpha}}) d_p^{(\alpha)} A_{\alpha} \tag{3.89}$$

with transformation components evaluated at the cell faces. Note that in curvilinear coordinates the equations become implicit, since a pressure term D_t[dP] must be transferred to the RHS.

Equation (3.88) is an exact equation for pressure once the velocity field is known. In practice, we require the simultaneous solution of coupled velocity and pressure equations and the solution of a *pressure/velocity-correction* equation is used to correct both velocity and pressure fields iteratively; (see Section 3.2.8 below).

The numerical solution of the pressure equation may, on occasion, be hampered by loss of

precision, particularly using FORTRAN single precision variables and in density-stratified flows where a large proportion of the pressure variation is hydrostatic. For this reason it is convenient: (a) to absorb the existing pressure field into the source coefficient b_p and treat (3.88) as an equation for the pressure correction; (since P' vanishes in the converged solution we can neglect $D_t[dP']$); (b) to subtract the hydrostatic pressure gradient (evaluated from the inflow Θ profile) from the momentum source term, so solving, in effect, for the non-hydrostatic pressure.

3.2.6 Advection Schemes

The integral or finite-volume formulation of the flow equations requires that the advective flux $\int \rho\phi\vec{U}.d\vec{A}$ of a general scalar ϕ be specified for each control volume face. If the mass flux across the cell face is known then the advection discretisation problem amounts to specifying the face value ϕ_f at a position intermediate between adjacent nodes.

The *conservation* property demands that ϕ_f be a property of the cell face, not the control volume on either side. The *transportive* property - that in the absence of sources or diffusion ϕ is constant along a streamline: $\vec{U}.\nabla\phi=0$ - is addressed by building upwind bias into the specification of ϕ_f. To this end it is convenient to define **U**pwind, **C**entral and **D**ownwind nodes depending on the direction of the mass flux through a cell face as in Figure 3.5, with corresponding nodal values ϕ_U, ϕ_C and ϕ_D respectively.

A number of well-known advection schemes fit into this three-point, upwind-biased framework; for example:

upwind differencing (UD): $\phi_f=\phi_C$

linear upwind differencing (LUD): $\phi_f=\phi_C+\frac{1}{2}(\phi_C-\phi_U)$

quadratic upwind differencing (QUICK): $\phi_f=\phi_C+\frac{1}{2}(\phi_D-\phi_C)-\frac{1}{8}(\phi_D-2\phi_C+\phi_U)$

These are, respectively, first-, second- and third-order accurate in space (on a uniform grid).

These schemes are defined coordinate-wise and depend only upon the sign of the mass flux, not the actual direction of the velocity vector. Skew upwind-differencing schemes (eg,

Raithby, 1976; Lillington, 1981) extrapolate back along the velocity vector to find an upwind value of ϕ, thus eliminating some of the numerical truncation errors that arise when the velocity is misaligned with the grid. However, such schemes are difficult to formulate on a curvilinear mesh and have not been pursued here.

The counter-requirements of numerical stability/boundedness and order/accuracy can be largely resolved by *flux-limiting*, or permitting just sufficient numerical diffusion to forestall unphysical oscillations whilst retaining high-order accuracy wherever possible. Such a limiting procedure for upwind schemes has been described by Leonard and Mokhtari (1990) in terms of normalised variables

$$\tilde{\phi} = \frac{\phi - \phi_U}{\phi_D - \phi_U} \tag{3.90}$$

Thus $\tilde{\phi}_U=0$, whilst $\tilde{\phi}_D=1$, and the behaviour is *monotonic* if $0 \le \tilde{\phi}_C \le 1$. In the normalised variable diagram $(\tilde{\phi}_f, \tilde{\phi}_C)$ (see Figure 3.6) the conditions for non-oscillatory results (Gaskell and Lau, 1988) are: that the scheme be continuous with $\partial\tilde{\phi}_f/\partial\tilde{\phi}_C > 0$; it must pass through the points (0,0) and (1,1); and, in the monotonic region, must satisfy $\tilde{\phi}_C \le \tilde{\phi}_f \le 1$ (delimited by the hatching in Figure 3.6. A steep but finite positive slope is needed as $\tilde{\phi}_C \to 0+$ to avoid non-uniqueness (see Gaskell and Lau, 1988). A scheme will be second-order accurate if it passes through the point (1/2,3/4) and third-order accurate if it does so with slope 3/4. The additional "total-variation-diminishing" constraint (Sweby, 1984) is $\tilde{\phi}_f \le 2\tilde{\phi}_C$ in the monotonic region, but this is over-restrictive when it comes to preventing numerical oscillations.

The standard upwind schemes may be rewritten in normalised variable form:

first-order upwind differencing: $\tilde{\phi}_f = \tilde{\phi}_C$

linear upwind differencing: $\tilde{\phi}_f = \frac{3}{2}\tilde{\phi}_C$

quadratic upwind differencing: $\tilde{\phi}_f = \frac{3}{8} + \frac{3}{4}\tilde{\phi}_C$

Apart from first-order differencing a limiter would be required to constrain the normalised face value within the monotonic region of Figure 3.6.

We have found the harmonic scheme of Van Leer (Van Leer, 1974; Leonard and Mokhtari, 1990) to be particularly suited to separated-flow problems. In normalised variables the

incompressible version can be written:

$$\tilde{\phi}_f = \begin{cases} \tilde{\phi}_C(2-\tilde{\phi}_C) & 0 \le \phi_C \le 1 \\ \tilde{\phi}_C & \text{otherwise} \end{cases} \tag{3.91}$$

This is illustrated in Figure 3.6 together with the third-order UMIST scheme (Lien and Leschziner, 1993), based on a limited form of QUICK.

In unnormalised variables Van Leer's scheme is

$$\phi_f = \begin{cases} \phi_C + \dfrac{(\phi_D-\phi_C)(\phi_C-\phi_U)}{\phi_D-\phi_U} & \text{if } (\phi_D-\phi_C)(\phi_C-\phi_U)>0 \\ \phi_C & \text{otherwise} \end{cases} \tag{3.92}$$

The scheme is second-order accurate where monotonic and reduces to first-order upwinding otherwise.

In practice, the flux-limited schemes may be incorporated into TEACH-like codes by moving the departure from upwind differencing into the source term. Write them as

$$\phi_f = \phi_C + \theta_f(\phi_D-\phi_C) \tag{3.93}$$

where θ_f is a *downwind weighting factor*. The contribution to the advective flux (equation (3.62)) is

$$G_f(\phi_f-\phi_p) = G_f(\phi_C-\phi_p) + G_f\theta_f(\phi_D-\phi_C) \tag{3.94}$$

where G_f is the outward mass flux through a cell face. The first term gives the necessary non-negative contribution $\max(-G_f,0)$ to the matrix coefficient A_{p+f}; the second part is absorbed in the source term.

Accuracy of Differencing Schemes

For a one-dimensional function with values specified at discrete, regularly spaced points, define differential and shift operators D and δ respectively by

$$\begin{aligned} D &\equiv \Delta x \frac{\partial}{\partial x} \\ \delta\phi_i &\equiv \phi_{i+1} \end{aligned} \tag{3.95}$$

Since

$$\phi_{i+1} = \phi_i + \Delta x \frac{\partial \phi}{\partial x_i} + \frac{(\Delta x)^2}{2!}\frac{\partial^2 \phi}{\partial x^2} + \ldots \equiv e^D \phi_i \tag{3.96}$$

we have a formal operator equivalence

$$e^D \equiv \delta \tag{3.97}$$

In these terms,

$$\phi_f = e^{D/2}\phi_i \tag{3.98}$$

and the specification of ϕ_f in terms of the neighbouring nodal values amounts to finding an approximation for $e^{D/2}$ in (positive and negative) powers of δ.

Now

$$e^{D/2} = 1 + \tfrac{1}{2}D + \tfrac{1}{8}D^2 + \tfrac{1}{48}D^3 + \ldots \tag{3.99}$$

whilst, by (3.97),

$$\begin{aligned} \delta - 1 &= D + \tfrac{1}{2}D^2 + \tfrac{1}{6}D^3 + \ldots \\ 1 - \delta^{-1} &= D - \tfrac{1}{2}D^2 + \tfrac{1}{6}D^3 + \ldots \\ \tfrac{1}{2}(\delta - 2 + \delta^{-1}) &= \tfrac{1}{2}D^2 + \tfrac{1}{24}D^4 + \ldots \end{aligned} \tag{3.100}$$

Using these we can readily establish the order of (one-dimensional) upwind schemes. These

are given in Table 3.2 below.

Scheme	Operator	Leading order error	Order
UD	1	$-1/2D$	1
LUD	$1+\frac{1}{2}(1-\delta^{-1})$	$-3/8D^2$	2
QUICK	$1+\frac{1}{2}(\delta-1)-\frac{1}{8}(\delta-2+\delta^{-1})$	$1/16D^3$	3
Van Leer	$1+\frac{1}{2}(\delta-1)-\frac{1}{8}\gamma(\delta-2+\delta^{-1})$	$-1/8D^2$	2

Table 3.2: Operator form and leading order error for common differencing schemes ($G_f>0$). In the last, $\gamma=4|r|/(1+|r|)$, $r=(\delta-1)/(1-\delta^{-1})=1+O(D)$.

The formal mathematical "order" of an advection scheme is a useful indicator of numerical accuracy, but it is worth remembering that it is usually (as above) determined from a one-dimensional, uniform-grid analysis. Thompson and Mastin (1985) show that, contrary to popular belief, uniform-grid order may be formally preserved on a non-uniform grid. This is because the discretisation is, mathematically, a mapping between coordinate nodes and the positive integers (the indices of the grid nodes), which are, of course, equally spaced along the real line. The conservative form of the advection term maps according to

$$\frac{\partial}{\partial x^j}(\rho U^j\phi) \rightarrow \frac{1}{\sqrt{g}}\frac{\partial}{\partial \xi^j}\left(\sqrt{g}\frac{\partial \xi^j}{\partial x^k}\rho U^k\phi\right) \tag{3.101}$$

so that, provided the coordinate derivative terms are included (to comparable accuracy) the formal mathematical order can be maintained. In practice, however, the leading order in the truncation error may become very large if the expansion ratio is too great.

Order is, however, strictly limited when using coordinate-wise schemes in multi-dimensional computations, since ϕ varies across a cell face. For example, in two dimensions, considering a cell face normal to the x-grid line,

$$\phi = \phi_f + \left(\frac{\partial\phi}{\partial y}\right)_f (y-y_0) + \frac{1}{2}\left(\frac{\partial^2\phi}{\partial y^2}\right)_f (y-y_0)^2 + \ldots \qquad (3.102)$$

where ϕ and its derivatives are evaluated at the centre of the face. Integrating between $y=y_0\pm\frac{1}{2}\Delta y$, and assuming U constant, we find that

$$\frac{1}{\Delta y}\int_{y_0-\frac{1}{2}\Delta y}^{y_0+\frac{1}{2}\Delta y}\phi\, dy = \phi_f + \frac{1}{24}\left(\frac{\partial^2\phi}{\partial y^2}\right)_f (\Delta y)^2 + \ldots \qquad (3.103)$$

so that there is generally a truncation error which is second order in Δy.

3.2.7 Solution of Matrix Equations

In common with other TEACH-like codes, SWIFT uses line-iteration procedures for the solution of the discretised flow equations. The normalised matrix equations (3.65) are split into components along each grid line. For example, keeping η and ζ constant and solving in the ξ-direction,

$$\phi_p - a_{p-\xi}\phi_{p-\xi} - a_{p+\xi}\phi_{p+\xi} = b_p + a_{p-\eta}\phi^*_{p-\eta} + a_{p+\eta}\phi^*_{p+\eta} + a_{p-\zeta}\phi^*_{p-\zeta} + a_{p+\zeta}\phi^*_{p+\zeta} \qquad (3.104)$$

Contributions from nodes off this grid line are transferred to the source term. A superscript * denotes the last computed value. The system is solved for this particular ξ-line by the tri-diagonal matrix algorithm.

The iterative algorithm sweeps first along all the ξ-lines (say), then all the η-lines and all the ζ-lines. The process is repeated until satisfactory convergence is achieved.

Two non-standard means of accelerating this iterative procedure have been examined:

(i) the block-correction procedure of Patankar (1988);

(ii) the anticipated-correction procedure [my name] of Van Doormaal and Raithby (1984).

Block Correction Technique

Overall convergence may be accelerated by preceding the directional sweeps by block corrections applied across planes of constant coordinate values; for example, planes of constant ξ:

$$\phi_{ijk} \rightarrow \phi_{ijk} + C_i \tag{3.105}$$

(The shorthand subscript p is used interchangeably with the indexing triplet ijk.) The set of corrections $\{C_i\}$ is deduced by summing the matrix equations over j and k values; thus:

$$\begin{aligned}\sum_{j,k}(\phi_p + C_i) \; & - \sum_{j,k} a_{p-\xi}(\phi_{p-\xi} + C_{i-1}) - \sum_{j,k} a_{p+\xi}(\phi_{p+\xi} + C_{i+1}) \\ & - \sum_{j,k} [a_{p-\eta}(\phi_{p-\eta} + C_i) + a_{p+\eta}(\phi_{p+\eta} + C_i) + a_{p-\zeta}(\phi_{p-\zeta} + C_i) + a_{p+\zeta}(\phi_{p+\zeta} + C_i)] \; = \sum_{j,k} b_p\end{aligned} \tag{3.106}$$

This yields a soluble tri-diagonal system for the $\{C_i\}$:

$$\alpha_i C_i + \beta_i C_{i-1} + \gamma_i C_{i+1} = \delta_i \tag{3.107}$$

where

$$\begin{aligned}\alpha_i &= \sum_{j,k}(1 - a_{p-\eta} - a_{p+\eta} - a_{p-\zeta} - a_{p+\zeta}) \\ \beta_i &= \sum_{j,k} a_{p-\xi} \\ \gamma_i &= \sum_{j,k} a_{p+\xi} \\ \delta_i &= \sum_{j,k}(b_p - \phi_p + \sum_d a_{p+d}\phi_{p+d})\end{aligned} \tag{3.108}$$

Similar block corrections are applied for planes of constant η and constant ζ. The scheme amounts to solving *integral* flow equations.

Anticipated Correction Scheme

In solving the linear equations along a ξ-grid line (say), contributions from nodes off this line are transferred to the RHS to supplement the source term. According to the sequence of

scanning through the ξ-lines some of these contributions come from nodes which have already been updated, others not; for example:

$$\phi_p - a_{p-\xi}\phi_{p-\xi} - a_{p+\xi}\phi_{p+\xi} = b_p + \underbrace{(a_{p-\eta}\phi^*_{p-\eta} + a_{p-\zeta}\phi^*_{p-\zeta})}_{\text{already updated}} + \underbrace{(a_{p+\eta}\phi^*_{p+\eta} + a_{p+\zeta}\phi^*_{p+\zeta})}_{\text{not yet updated}} \quad (3.109)$$

A better estimate of $\phi^*_{p+\eta}$ is obtained by adding (implicitly) a fraction r of the change in ϕ_p. Thus

$$\begin{aligned} \phi^{(new)}_{p+\eta} &= \phi^{(old)}_{p+\eta} + (\phi^{(new)}_{p+\eta} - \phi^{(old)}_{p+\eta}) \\ &\approx \phi^{(old)}_{p+\eta} + r(\phi^{(new)}_{p} - \phi^{(old)}_{p}) \end{aligned} \quad (3.110)$$

with a similar expression for $\phi^*_{p+\zeta}$. The result is the less diagonally dominant system

$$\begin{aligned} [1 - r(a_{p+\eta} + a_{p+\zeta})]\phi_p - a_{p-\xi}\phi_{p-\xi} - a_{p+\xi}\phi_{p+\xi} &= [b_p - r(a_{p+\eta} + a_{p+\zeta})\phi^*_p] \\ &\quad + a_{p-\eta}\phi^*_{p-\eta} + a_{p+\eta}\phi^*_{p+\eta} + a_{p-\zeta}\phi^*_{p-\zeta} + a_{p+\zeta}\phi^*_{p+\zeta} \end{aligned} \quad (3.111)$$

The implementation of the procedure introduces few overheads, requiring only minor modifications to the tridiagonal coefficients. Van Doormaal and Raithby recommend $r \approx 0.85$ although we have found this to lead to divergence on occasion and have preferred the more robust value of 0.80.

Experience to date indicates that both modifications are more effective for the pressure-correction than for the scalar-transport equations. Computational advantages depend on the test case but computations of two-dimensional, laminar separated flow over a fence with either acceleration method indicated reduction in CPU time of greater than 50%. The greatest benefits accrue when convergence of the continuity equation would otherwise be the limiting factor and a more efficient procedure in future may be to "switch in" the acceleration methods when the pressure-correction is the equation with the largest residuals.

3.2.8 Handling of Equation Coupling

The features of the full Navier-Stokes equations which make analytic solutions scarce and numerical solution procedures necessarily iterative are the non-linearity and strong coupling between flow variables.

The non-linearity of the advection terms in particular means that the matrix coefficients a_{p+d} change as the velocity field changes and must therefore be updated after each iteration.

For primitive-variable solvers the strongest coupling is usually between velocity and pressure, but other coupling exists - for example, through turbulent stresses or buoyancy forces. The main alternatives for handling coupled equations are:

- *block simultaneous* - solve all equations together;
- *sequential* - one equation at a time.

In addition schemes may be either

- *time-factored*;
- *iterative.*

In this classification SIMPLE (Patankar, 1980) and its derivatives are sequential and iterative. PISO (Issa, 1985) is a time-factored sequential scheme. Block simultaneous schemes are considerably more complex than sequential schemes, both in their use of block solvers and in the rearrangement necessary when new variables are introduced.

For cartesian and mildly distorted grids the original SIMPLE scheme has been superseded by a family of more efficient descendants, of which three - SIMPLEC, SIMPLEX and SIMPLER - have been encoded in SWIFT. Although these differ in detail they share a common principle with their ancestor: solve the momentum equations, formulate an integral mass-conservation equation over a scalar control volume and then use an approximate relationship between velocity and pressure changes to solve a pressure-correction equation and steer the solution towards continuity.

Discretisation of the momentum equation gives rise to a linear relationship between each velocity component and the difference between adjacent pressure values (equation (3.79)):

$$U_p^{\alpha} - \sum_d a_{p+d}^{(\alpha)} U_{p+d}^{\alpha} = b_p^{(\alpha)} + d_p^{(\alpha)} (P_{p-\alpha} - P_p) \tag{3.112}$$

Let $\vec{U}^*$ and P^* be velocity and pressure fields from the previous iteration, consistent with (3.112). These will not necessarily satisfy continuity. If

$$\begin{aligned} \vec{U} &= \vec{U}^* + \vec{U}' \\ P &= P^* + P' \end{aligned} \tag{3.113}$$

is the mass-consistent solution of the Navier-Stokes equations (linearised in the form (3.112)) then the correction fields must satisfy

$$U_p^{\alpha\prime} = \sum_d a_{p+d}^{(\alpha)} U_{p+d}^{\alpha\ \prime} + d_p^{(\alpha)} (P'_{p-\alpha} - P'_p) \tag{3.114}$$

Equation (3.114) is exact ... but not very helpful. Let us assume that there is some approximate relationship between a correction-velocity component and the corresponding local pressure gradient:

$$U_p^{\alpha\prime} = \delta_p^{(\alpha)} (P'_{p-\alpha} - P'_p) \tag{3.115}$$

Forming the divergence of $\vec{U} = \vec{U}^* + \vec{U}' = \vec{U}^* + \delta P'$ we have, for an incompressible fluid, that $D[\vec{U}] = 0$, or

$$D_n[\delta P'] = -D[\vec{U}^*] \tag{3.116}$$

(For mildly distorted grids we can neglect the off-diagonal curvilinear correction $D_t[\delta P']$.) Expanding the LHS this becomes

$$\sum_d a_{p+d}^{(P)} (P'_p - P'_{p+d}) = -D[\vec{U}^*] \tag{3.117}$$

with the matrix coefficients $a_{p+d}^{(P)}$ given by (3.89) (but with d_p replaced throughout by δ_p). The source for the P' field is (minus) the local divergence. Thus, in a rough, hand-waving sort of way, where there is a large net mass inflow into a cell (ie, negative divergence) P' is large and tends to "push" fluid out of the control volume. Solving (3.117) for P' the velocity and

pressure fields can be updated by (3.113) and (3.115).

In the original SIMPLE scheme (Patankar, 1980), the first term on the RHS of (3.114) is neglected, so that one has simply

$$\delta_p^{(\alpha)} \equiv d_p^{(\alpha)} \tag{3.118}$$

With this scheme convergence tends to be slow: severe under-relaxation is needed for both momentum and pressure equations.

One of the problems with the above approximation is that the terms neglected are often of the same magnitude as those retained. In the SIMPLEC scheme (Van Doormaal and Raithby, 1984) (3.114) is rewritten as

$$(1-\sum_d a_{p+d}^{(\alpha)})U_p^{\alpha\prime} = \sum_d a_{p+d}^{(\alpha)}(U_{p+d}^{\alpha\prime}-U_p^{\alpha\prime}) + d_p^{(\alpha)}(P_{p-\alpha}^{\prime}-P_p^{\prime}) \tag{3.119}$$

Now the first term on the RHS can be neglected legitimately if we assume that the local change in $\vec{U}'$ is small. Then the pressure correction coefficients become

$$\delta_p^{(\alpha)} \equiv \frac{d_p^{(\alpha)}}{1-\Sigma a_{p+d}^{(\alpha)}} \tag{3.120}$$

Equation (3.120) becomes exact in source-dominated flows.

The SIMPLEX scheme (Raithby and Schneider, 1988) attempts to solve for the pressure correction coefficients $\delta_p^{(\alpha)}$. Substituting (3.115) in (3.114) we have

$$-\delta_p^{(\alpha)}\Delta_p^{(\alpha)}P' = -\sum_d a_{p+d}^{(\alpha)}\delta_{p+d}^{(\alpha)}\Delta_{p+d}^{(\alpha)}P' - d_p^{(\alpha)}\Delta_p^{(\alpha)}P' \tag{3.121}$$

where $\Delta_p^{(\alpha)}P' \equiv P_p^{\prime}-P_{p+\alpha}^{\prime}$. If we now assume that $\Delta_{p+d}^{(\alpha)}P' = \Delta_p^{(\alpha)}P'$ then we have a matrix equation for the pressure-correction coefficients:

$$\delta_p^{(\alpha)} = \sum_d a_{p+d}^{(\alpha)} \delta_{p+d}^{(\alpha)} + d_p^{(\alpha)} \qquad (3.122)$$

Since the matrix coefficients $a_{p+d}^{(\alpha)}$ have already been set up for the corresponding momentum equations, little extra effort is required.

SIMPLER (Patankar, 1980) adopts the same correction coefficients as SIMPLE, but proceeds on the premise that the correction fields are good for updating the velocity (since a mass-consistent flow field is produced) but not the pressure. Instead the pressure field is updated from the *exact* pressure equation (3.88) prior to solving the momentum equations.

Although no detailed comparison of the algorithms for handling pressure and velocity coupling has been undertaken, experimentation with the test cases of Chapter 6 showed that convergence rates were considerably better with SIMPLEX and SIMPLER than SIMPLE and SIMPLEC. No definitive preference could be established between SIMPLEX and SIMPLER although the former is claimed by its authors to exhibit less degradation of performance as the grid is refined. In practice, SIMPLEX was used for large three-dimensional calculations simply because each iteration required smaller computational time and it was more quickly established whether computations were converging or diverging for a particular set of relaxation parameters.

The sequence of calculations for one SIMPLEX iteration is shown in Figure 3.7. Convergence is accepted when the normalised equation residuals fall to less than some prescribed tolerance (generally 10^{-4}). Since the coefficients change at each iteration there is little point in solving the pressure correction equation exactly at each stage and the practice of Van Doormaal and Raithby (1984) was adopted, with the number of iterations such as to reduce the sum of matrix residuals to some fixed fraction (10%) of its initial values. For the transport equations a single iteration of the matrix solver is adequate in each SIMPLEX loop.

3.2.9 Topographic Coordinate Transformation in SWIFT

The calculation of flow around curved boundaries using structured grids may be accomplished either by laying a rectangular grid over the domain and applying special treatment to the boundary-spanning cells or by using a curvilinear mesh which conforms to the curved surface. The latter course is adopted in SWIFT because it permits a more efficient resolution of the boundary layer. However, as we have seen, additional terms arise in the discretisation of advective and diffusive fluxes.

In SWIFT a simple vertical stretching conforms a structured grid to arbitrary smooth topography (Figure 3.8):

$$\begin{aligned} \xi &= x \\ \eta &= y \\ \zeta &= \frac{z - z_s}{1 - z_s/D} \end{aligned} \tag{3.123}$$

where $z_s(x,y)$ is the local height of topography and D is the height of the computational domain. The simplicity of the transformation is reflected in the comparatively small number of extra terms to be calculated. However, to permit future extension to more complex geometries, our policy has been to code for a completely general coordinate transformation and "comment out" redundant terms for the current application.

The main changes to the scalar transport equation necessitated by a curvilinear mesh are:

(i) inclusion of the volume dilation factor $\sqrt{g}$;

(ii) calculation of contravariant velocity and diffusivity components in advection and diffusion terms respectively;

(iii) transfer of pressure and scalar derivatives parallel to cell faces to the source term.

This requires additional geometric variables G_{ij}, A^i_j, which are given for the simple transformation (3.123) by:

$$
\begin{aligned}
G^{ij} &\equiv \sqrt{g}\, g^{ij} = \begin{pmatrix} \gamma & 0 & -c_x \\ 0 & \gamma & -c_y \\ -c_x & -c_y & \dfrac{1+c_x^2+c_y^2}{\gamma} \end{pmatrix} \\
A^i{}_j &\equiv \sqrt{g}\,\frac{\partial \xi^i}{\partial x^j} = \begin{pmatrix} \gamma & 0 & 0 \\ 0 & \gamma & 0 \\ -c_x & -c_y & 1 \end{pmatrix} \\
J &\equiv \sqrt{g} = \gamma
\end{aligned}
\tag{3.124}
$$

where

$$
\gamma = 1 - z_s/D\ , \qquad c_x = (1-\zeta/D)\frac{\partial z_s}{\partial x}\ , \qquad c_y = (1-\zeta/D)\frac{\partial z_s}{\partial y} \tag{3.125}
$$

Contravariant velocity components and (isotropic) diffusivities are then given by

$$
\begin{aligned}
\sqrt{g}\tilde{U}^i &= A^i{}_j U^j \\
\sqrt{g}\tilde{\Gamma}^{ij} &= G^{ij}\Gamma
\end{aligned}
\tag{3.126}
$$

3.2.10 Bluff-Body Blockages in SWIFT

Although this research is primarily intended to provide a numerical model for flow over topography, a capacity to compute flow over bluff bodies in a rectilinear geometry has been maintained. In the context of environmental flows this may find application in the computation of flow and dispersion around large buildings.

Bluff-body blockages are readily incorporated by tagging the relevant scalar control volumes as solid (Figure 3.9) and modifying equation coefficients for control volumes around the boundary so that wall fluxes appear as a source term. The staggered-mesh configuration automatically ensures that the normal velocity is stored on the blockage boundary. Velocity components inside and on the blockage boundary can be forced to zero by modifying the

source terms:

$$\begin{aligned} B_P &\rightarrow 0 \\ S_P &\rightarrow -\gamma \end{aligned} \tag{3.127}$$

where γ is a large number: say 10^{20}.

3.2.11 Boundary Conditions

The structure of the SWIFT code enables the ready implementation of three types of boundary conditions:

- Dirichlet boundary conditions (ϕ specified);
- Neumann boundary conditions ($\partial\phi/\partial n$ specified);
- wall-function treatment of near-wall turbulent flow.

A brief (and unsuccessful) foray has also been made into radiation boundary conditions (Chapter 2) in an attempt to prevent internal-wave reflection contaminating the flow field in simulations of unbounded stably stratified flow. This will not be described further here.

In the majority of two- and three-dimensional applications for which this code was developed, variables are specified at inflow (Dirichlet boundary conditions) and zero longitudinal gradient assumed at outflow (homogeneous Neumann boundary conditions).

In general, boundary values at the edges of the computational domain are updated iteratively as the computation proceeds, whilst internal boundaries (for example, around internal blockages) are accommodated by amending the source term in the discretised equations. In the latter case, for scalars and tangential velocity components the diffusive flux through the cell face abutting a solid boundary (Figure 3.10) is specified directly through the source term; thus

$$\begin{aligned} A_{p+n} &\rightarrow 0 \\ S &\rightarrow S - F_n \end{aligned} \tag{3.128}$$

where A_{p+n} is the matrix coefficient connecting the near wall node with that in the normal

direction n, S is the (integrated) source term and F_n the (outward) flux. The component parts of this flux may be included in explicit or implicit parts of the source term as appropriate. For the normal velocity component the value is stored on an internal blockage boundary (Figure 3.10) and the values inside and on the boundary may be forced to zero by modifying the source term in the conventional manner:

$$\begin{aligned} B_p &\rightarrow 0 \\ S_p &\rightarrow -\gamma \end{aligned} \tag{3.129}$$

where γ is a large number.

The handling of near-wall turbulence using wall functions will be discussed further in Chapter 4. However, the near-wall values of tangential velocity and turbulent kinetic energy can be used to deduce the wall shear stress or an *effective* eddy viscosity. Since the eddy viscosity is stored on the boundaries of the computational domain, wall-function treatments are invoked here simply by amending the effective value of ν_t. At internal boundaries, however, the surface stress (=momentum diffusion) must be transferred to the source term according to the prescription above. The profiles defined by the wall-function treatment yield *cell-averaged* production and dissipation terms in the turbulent kinetic energy equation, whilst the value of the dissipation rate ε at the near-wall node is forced to take its equilibrium value $\varepsilon_p = C_\mu^{3/4} k^{3/2} / \kappa l_n$ (l_n the normal distance) using the conventional source-term linearisation technique.

3.2.12 Imbedded Analytical Domain Method For Isolated Releases

The transport equation for concentration C is formally similar to that for the non-passive scalar θ, with some important qualifications. Firstly (and conveniently) C does not affect the flow and so its equation can be solved separately after the flow has been computed. Secondly, concentrations are derived from a "point-source" release, small in relation to the computational grid. (Actually, since plume dimensions are initially small in relation to the diffusing eddies this raises some fundamental questions about the validity of the flux-gradient transfer hypothesis for concentration, but we shall not pursue this here: see Pasquill and

Smith, 1983, for a discussion.) Since the finite-volume method assumes the source term to be uniformly distributed over a control volume this implies undue initial smearing of the concentration distribution, particularly where the grid has been refined to resolve the flow rather than plume dispersion.

To overcome this we have done two things. Firstly, after computation of the flow field we have interpolated velocity components and turbulence onto a second grid refined near the source location. Secondly we have used an "imbedded analytical domain" method to specify concentrations directly near the source.

Interpolation of flow variables onto a second grid for the concentration calculation is legitimate in so far as the concentration does not affect the flow dynamics. However, one point which has been made earlier but which tends to be studiously avoided in the literature is that *after interpolation the velocity field cannot be assumed to satisfy mass continuity* and some care must be taken to assure global conservation of material.

The imbedded analytic domain method allows a much coarser computational mesh by employing the *exact solution of an advection-diffusion equation with constant coefficients* to specify concentrations *analytically* in a local domain D surrounding the source (Figure 3.11). In principle, the extent of the analytical region should be such that the velocity (and diffusivity) do not vary significantly throughout D, whilst the plume on the boundary ∂D should have spread sufficiently to be resolved by the computational grid. Actually, these criteria are easier to describe in words than to implement in practice, since the grid lines may not be aligned with the local flow and, for a curvilinear grid, are not necessarily straight. Procedures to determine D will be described below.

Within the code, externally specified concentrations may be forced at grid points using the conventional source term linearisation technique (Patankar, 1980):

$$S^{(\Phi)} = B_p + S_p \phi \tag{3.130}$$

with

$$\begin{aligned} B_p &= \gamma C_a \\ S_p &= -\gamma \end{aligned} \tag{3.131}$$

C_a is the analytically specified concentration value and γ is a large number (eg, 10^{20}).

The advection-diffusion equation with constant coefficients

$$U\frac{\partial C}{\partial x} - \Gamma\nabla^2 C = Q\delta(\vec{x}) \tag{3.132}$$

has a solution corresponding to a point source Q at the origin,

$$C = \begin{cases} \dfrac{Q}{4\pi\Gamma}\dfrac{e^{-\frac{U(r-x)}{2\Gamma}}}{r} & 3 \text{ dimensions} \\ \dfrac{Q}{2\pi\Gamma}e^{\frac{Ux}{2\Gamma}}\mathrm{K}_0\left(\frac{Ur}{2\Gamma}\right) & 2 \text{ dimensions} \end{cases} \tag{3.133}$$

where x is the coordinate in the direction of the mean wind, $r=(x^2+y^2+z^2)^{½}$ and K_0 is a modified Bessel function. Using the wind speed (and direction) and diffusivity at the source location this expression is used to specify C directly within a box D enclosing the source. To conserve material the flux across the bounding surface of D into the adjacent control volumes must also be specified directly. The flux across (finite) face area $\vec{A}$ is

$$\vec{F} = \vec{f}.\vec{A} \tag{3.134}$$

where $\vec{f}$ is the (advective+diffusive) flux density evaluated from (3.133) as

$$\vec{f} \equiv \vec{U}C - \Gamma\nabla C = \begin{cases} C\left[\frac{1}{2}\vec{U} + (\frac{1}{2}U + \frac{\Gamma}{r})\vec{e}_r\right] & 3 \text{ dimensions} \\ C\left(\frac{1}{2}\vec{U} + \frac{1}{2}U\dfrac{\mathrm{K}_1(\frac{Ur}{2\Gamma})}{\mathrm{K}_0(\frac{Ur}{2\Gamma})}\vec{e}_r\right) & 2 \text{ dimensions} \end{cases} \tag{3.135}$$

at the centre of the control volume face. This flux is effectively a source term for the neighbouring cell. Because of the discretisation involved (the flux density actually varies over the cell face) the total outward flux of material across ∂D is made equal to Q by scaling the

analytically-derived fluxes by $Q/\Sigma F$.

Outside the analytic region D the plume must be resolvable by the computational mesh and we shall require that the grid spacing be less than the plume width. At small distances from the plume centreline the argument of the exponential in (3.133) is

$$-\frac{U}{2\Gamma}(r-x) \;=\; -\frac{Ux}{2\Gamma}\left[(1+\frac{y^2+z^2}{x^2})^{1/2}-1\right] \;\approx\; -\frac{1}{2}\frac{(y^2+z^2)}{\sigma^2} \tag{3.136}$$

(using the binomial expansion) where $\sigma=\sqrt{2\Gamma x/U}$ is the plume standard deviation. Thus, at the downwind end of the analytical region we require that $\sigma\approx\gamma\Delta$, where Δ is the mesh size and γ is a constant of order unity. Substituting for σ and rearranging this gives for n, the downwind number of control volumes (assuming a uniform grid),

$$n \;\equiv\; \frac{x}{\Delta} \;=\; \frac{\gamma^2}{2}\frac{U\Delta}{\Gamma} \tag{3.137}$$

Thus, fixing the downstream end of the analytical domain at L, the length scale over which the velocity (and, in principle, the diffusivity) vary significantly, the mesh size around the source is required to satisfy

$$n\Delta \;\le\; L \tag{3.138}$$

or,

$$\Delta \;\le\; \frac{1}{\gamma}\left(\frac{2\Gamma L}{U}\right)^{1/2} \tag{3.139}$$

In practice, for rapidly varying three-dimensional flows, even allowing for grid expansion, this minimum mesh size in the vicinity of the source can turn out to be unrealistically small. In these circumstances the minimum mesh size Δ is set *a priori* and the following compromise algorithm adopted to determine the dimensions of the analytical domain D.

(i) Calculate the maximum value of n required to satisfy the constraint (3.137), with $U=|\vec{U}_s|$ the mean wind speed and $\Gamma=\Gamma_s$ the (eddy+molecular) scalar diffusivity (obtained by interpolation) and $\Delta=(\Delta x\Delta y\Delta z)^{1/3}$ the geometric mean mesh spacing at the

source location. Constant γ is taken as 1.0.

(ii) Consider a successively expanding box around the source control volume, with equal numbers of cells in each direction. Allow this box to expand until either n is reached or the relative velocity change $|\vec{U}-\vec{U}_s|/|\vec{U}_s|$ exceeds an appreciable fraction; (here 0.2).

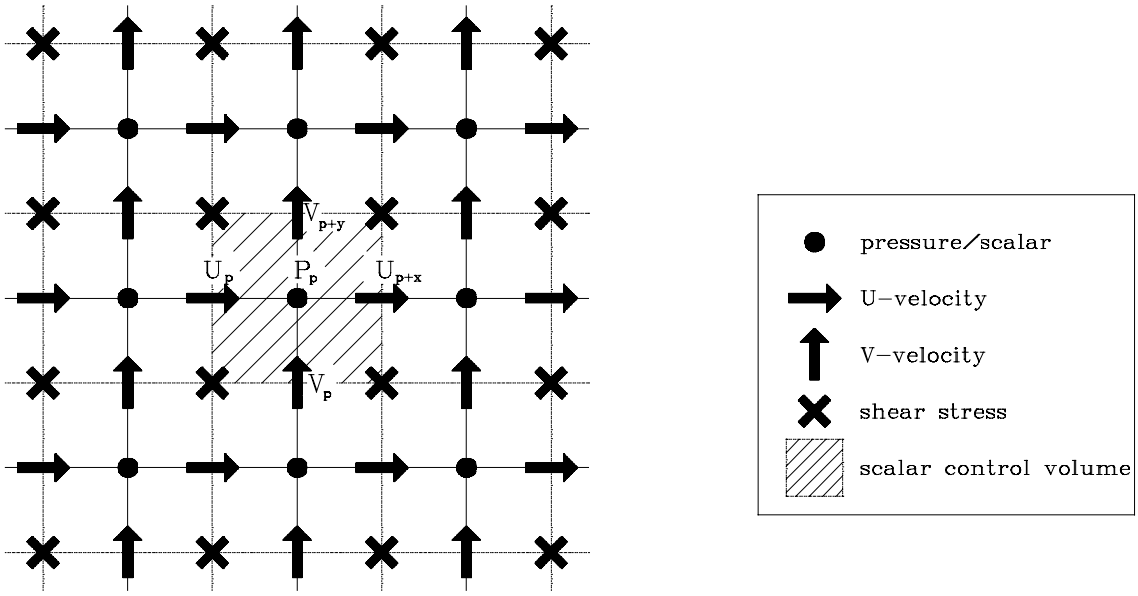

Figure 3.1: Staggered grid arrangement and indexing convention.

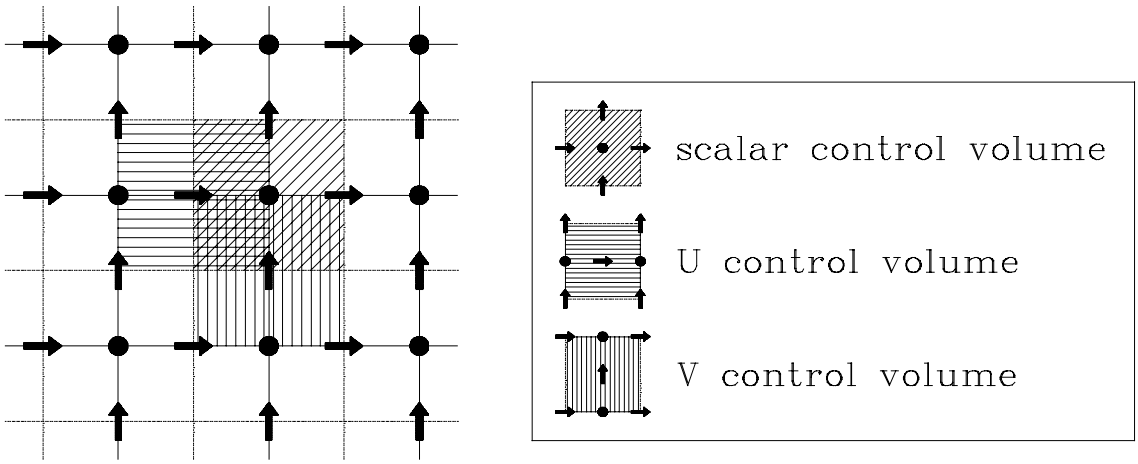

Figure 3.2: Control volume geometry.

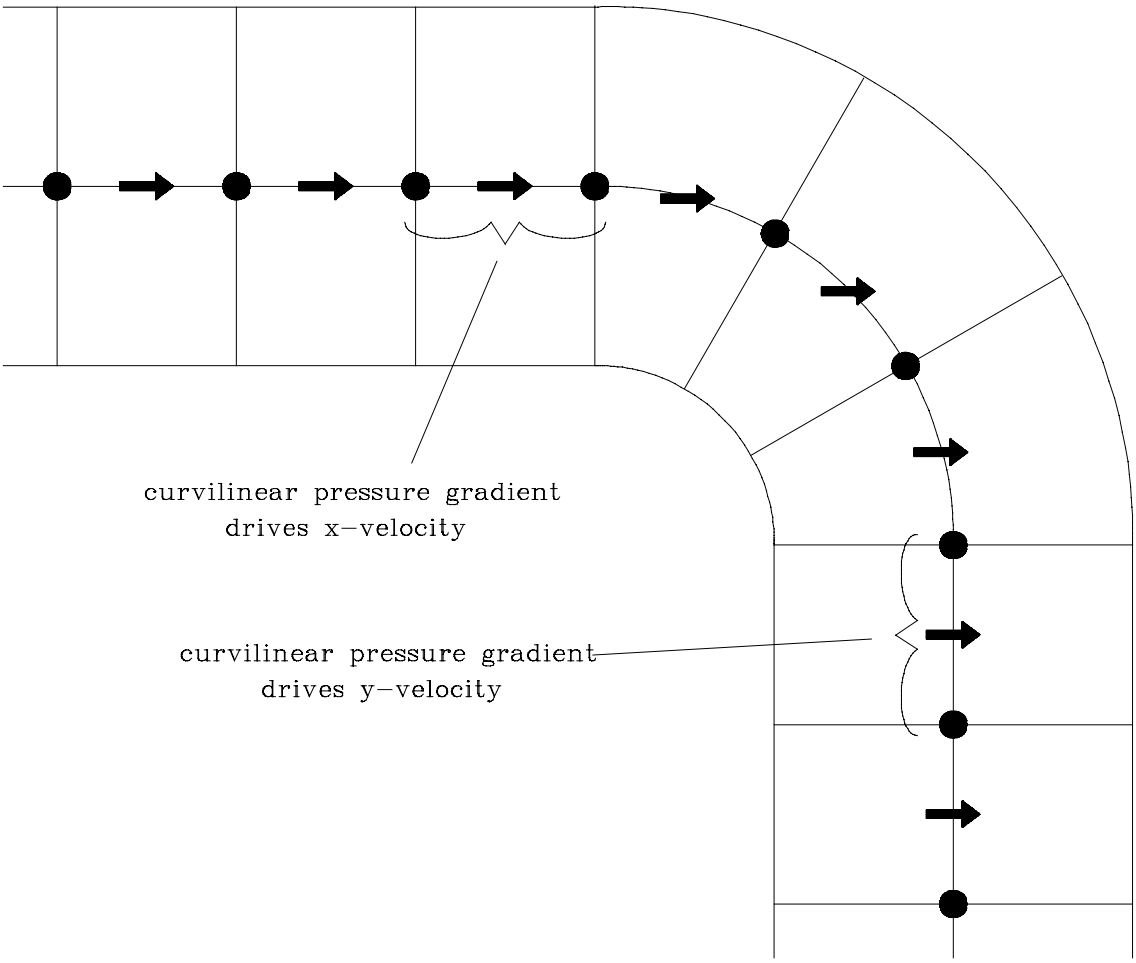

Figure 3.3: Cartesian velocity coordinates misaligned with coordinate lines in curved geometry.

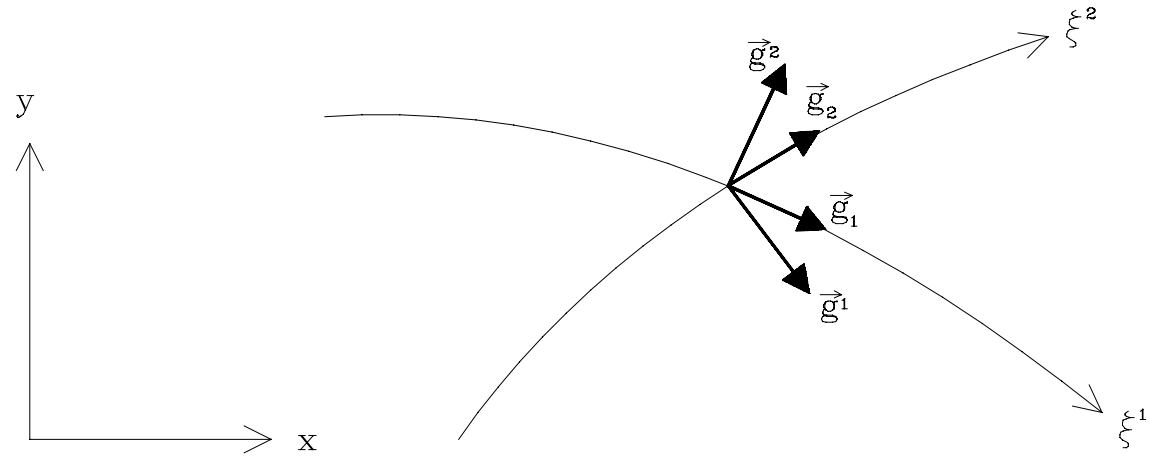

Figure 3.4: Covariant and contravariant basis vectors in a curvilinear coordinate system.

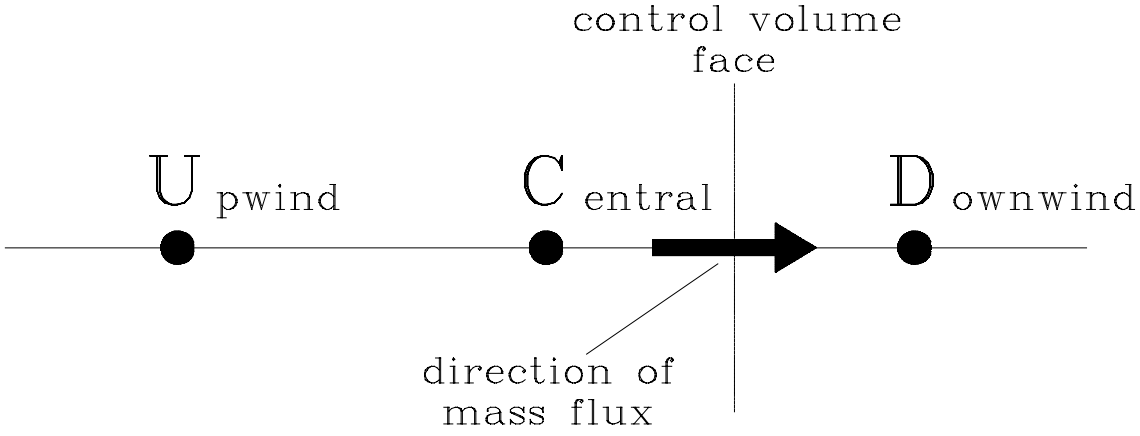

Figure 3.5: Definition of **U**pwind, **C**entral and **D**ownwind nodes in three-point upwind-biased advection schemes.

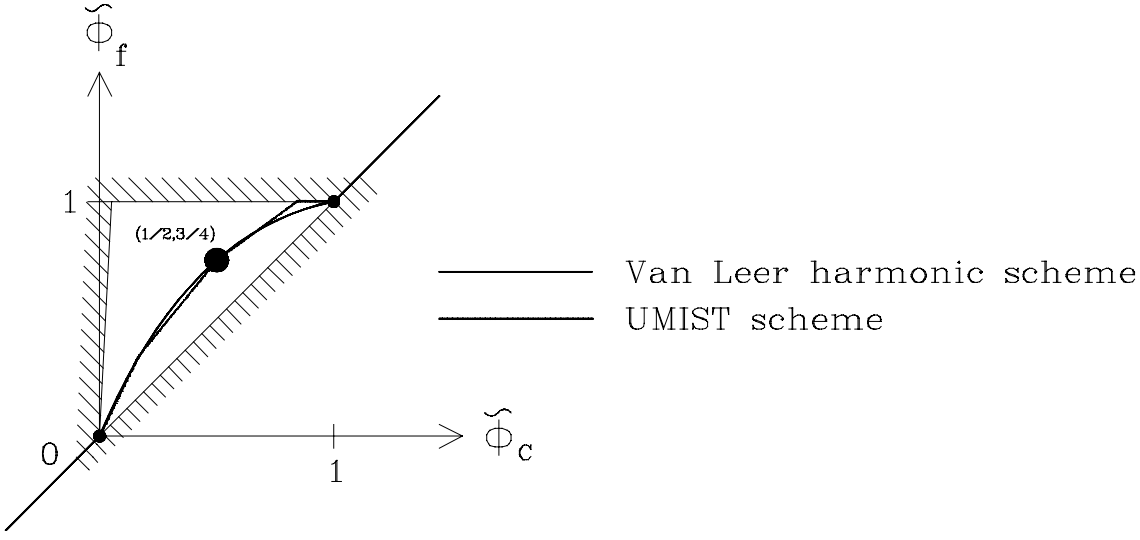

Figure 3.6: Normalised variable diagram: the hatching delimits the constraints in the monotonic region.

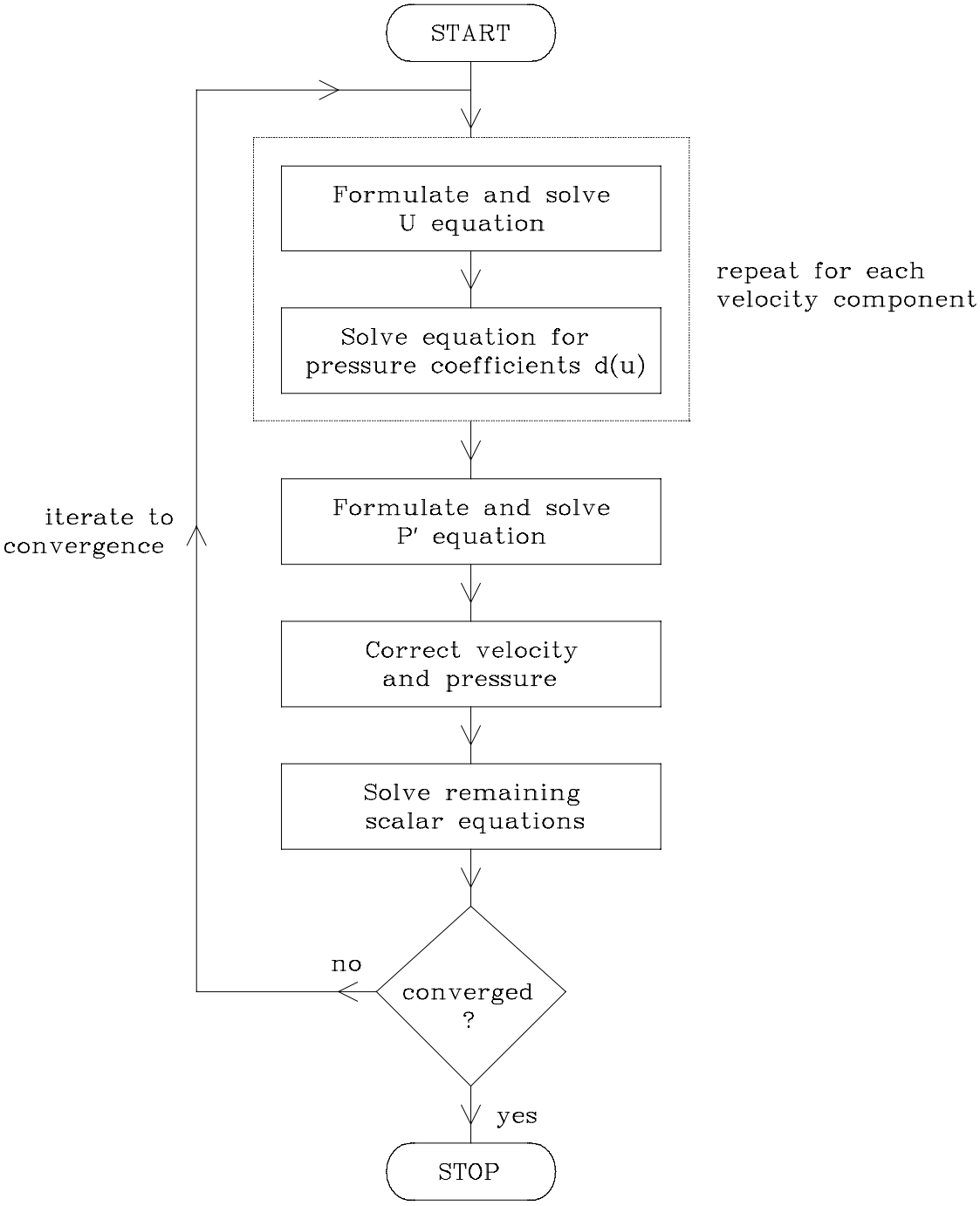

Figure 3.7: Sequence of operations in one SIMPLEX iteration.

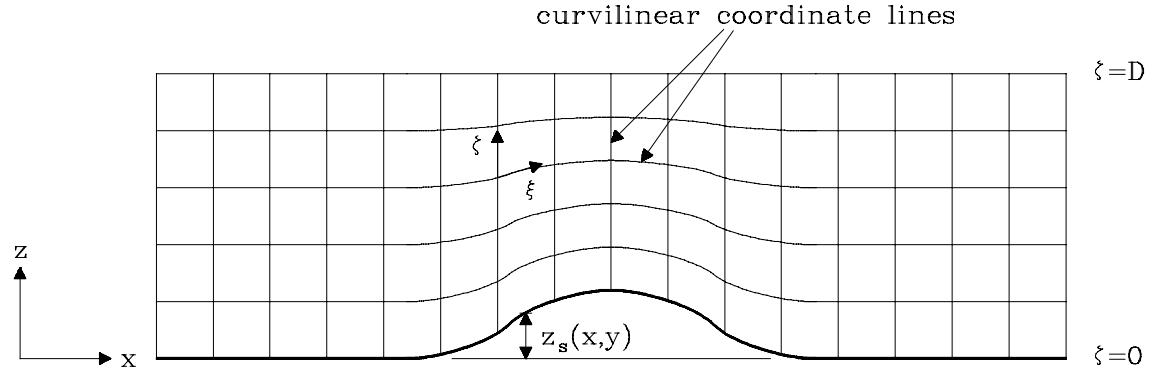

Figure 3.8: Terrain-following curvilinear coordinate system employed in SWIFT.

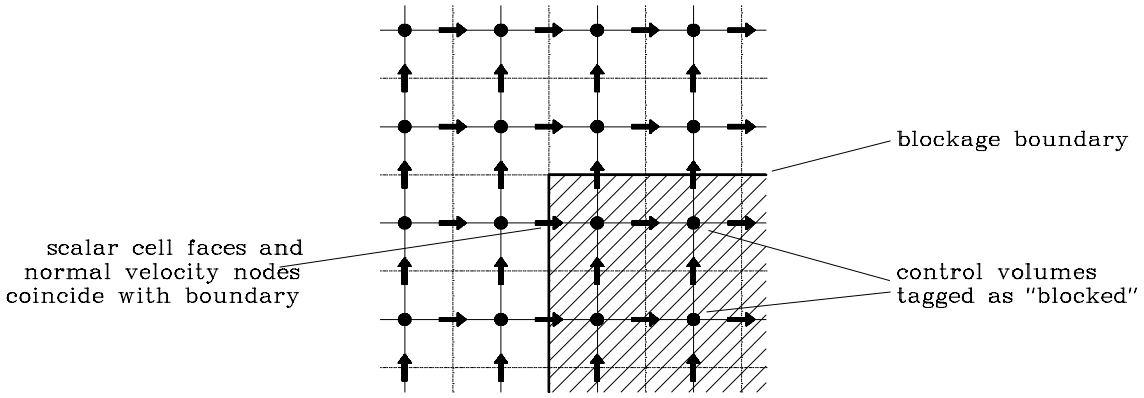

Figure 3.9: Arrangement of control volumes and tagged cells in bluff-body calculations.

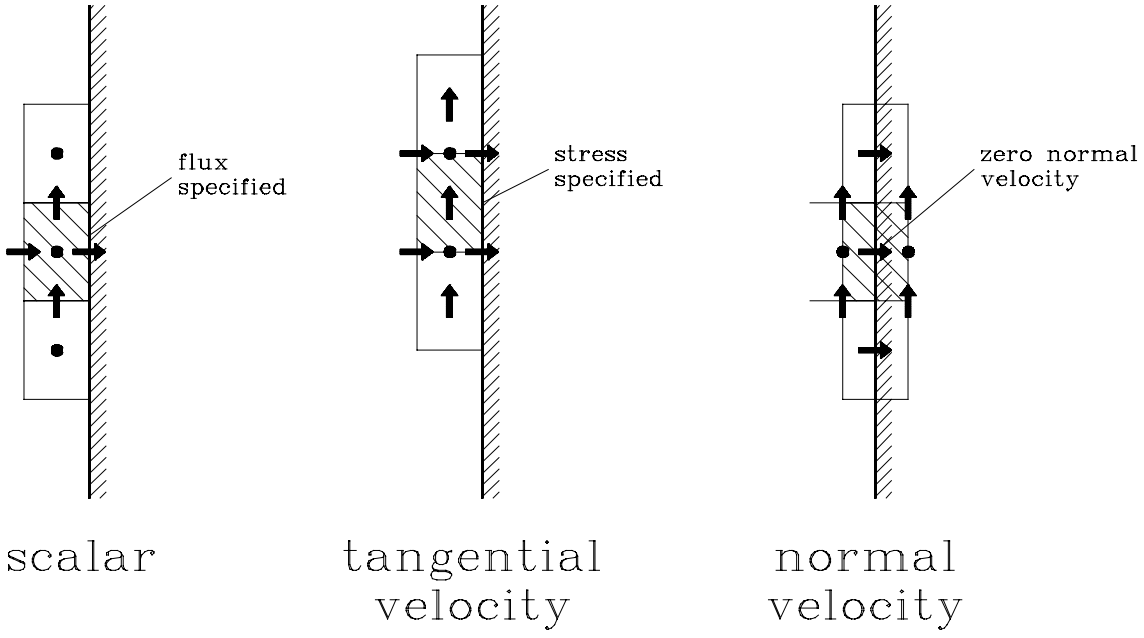

Figure 3.10: Disposition and special treatment of control volumes abutting internal boundary.

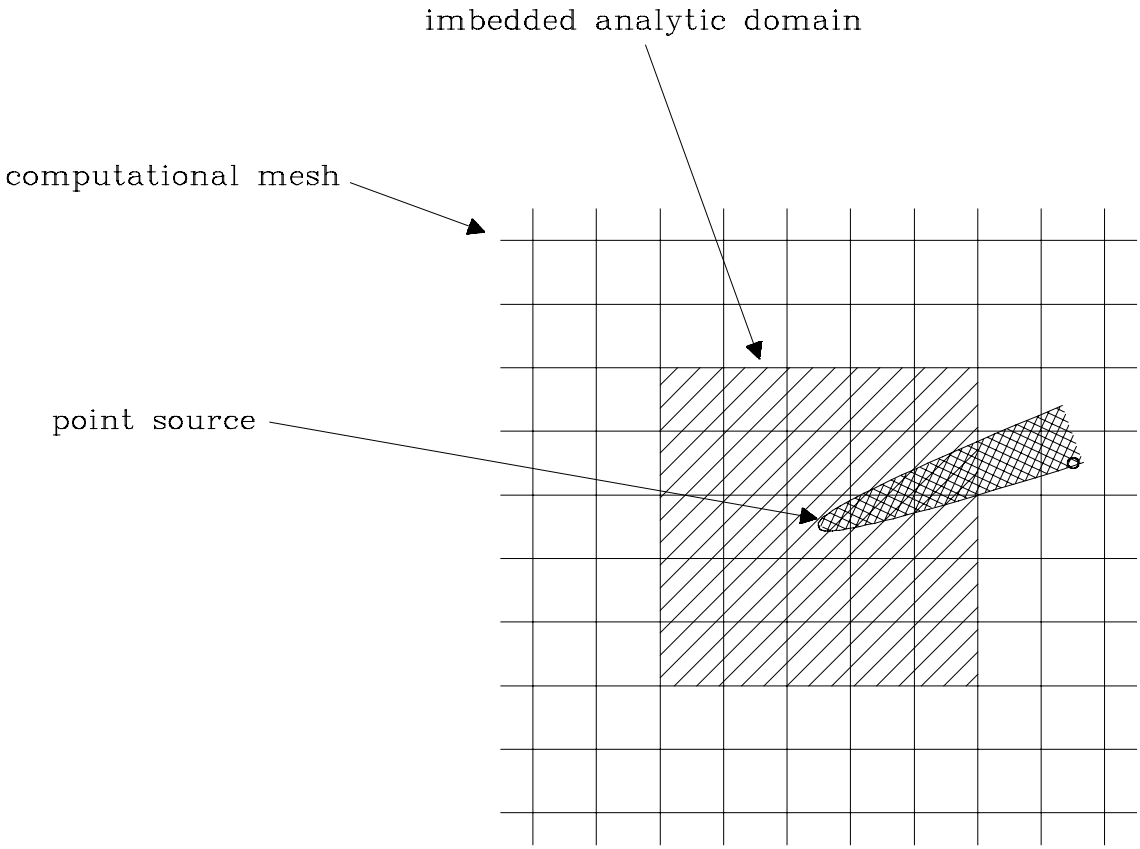

Figure 3.11: Imbedded analytical region for point source calculations.