CHAPTER 1

SURVEY OF NUMERICAL METHODS

J. Y. MURTHY

School of Mechanical Engineering
Purdue University
West Lafayette, Indiana, USA

W. J. MINKOWYCZ

Department of Mechanical Engineering
The University of Illinois at Chicago
Chicago, Illinois, USA

E. M. SPARROW

Department of Mechanical Engineering
University of Minnesota
Minneapolis, Minnesota, USA

S. R. MATHUR

Fluent Inc.
10 Cavendish Ct
Lebanon, New Hampshire, USA

1.1	INTRODUCTION	4
1.2	GOVERNING EQUATIONS	5
1.2.1	Continuity Equation	5
1.2.2	Momentum Equation	5
1.2.3	Energy Equation	5
1.2.4	Species Transport	6
1.2.5	General Scalar Transport Equation	6
1.3	ANATOMY OF A NUMERICAL SOLUTION	7
1.3.1	Domain Discretization	7
1.3.2	Discretization of Governing Equation	10
1.3.3	Solution of Linear Equations	15
1.3.4	Nonlinearity and Coupling	17
1.3.5	Properties of Numerical Solution Procedure	17
1.3.6	Summary and Discussion	18

1.4	COMPUTATIONAL TECHNIQUES FOR UNSTRUCTURED MESHES	18
1.4.1	Discretization of Convection–Diffusion Equation	18
1.4.2	Gradient Calculation	22
1.4.3	Summary and Discussion	25
1.5	HIGHER-ORDER SCHEMES FOR CONVECTION OPERATORS	25
1.5.1	Upwind-weighted Higher-order Schemes	26
1.5.2	Control of Spatial Oscillations	28
1.5.3	Summary and Discussion	31
1.6	LINEAR SOLVERS	31
1.6.1	Line Gauss-Seidel Method	32
1.6.2	Multigrid Methods	33
1.6.3	Gradient-search Techniques	37
1.7	COMPUTATION OF FLUID FLOW	38
1.7.1	Storage of Pressure and Velocity	38
1.7.2	Solution Methods	43
1.7.3	Density-based Schemes	45
1.8	CLOSURE	47

1.1 INTRODUCTION

During the last three decades, numerical simulation has come to play an increasingly important role in the analysis and design of engineering products and processes. A variety of techniques have been developed, and several have reached sufficient maturity to warrant routine use. It is not uncommon today for the industrial thermal analyst to use computational fluid dynamics (CFD) and computational heat transfer (CHT) techniques to do preliminary design in applications as diverse as electronics cooling, underhood automotive cooling, glass processing, as well as food, pharmaceutical, and chemical processing, to name a few. Large-scale simulations involving tens of millions of unknowns are now routinely performed in many industries using both serial and parallel processing. Nevertheless, though the solutions to many problems, especially those involving single-phase nonreacting Newtonian flows, are now within reach, a variety of industrial thermal and fluid flow problems remain intractable. These include gas–solid and gas–liquid flows, phase change, reacting flows, flows of viscoelastic fluids and other fluids with complex rheologies, and complex turbulent flows, among others. The challenges in solving these flows are related to deficiencies in existing numerical methods, insufficient computational power, and an incomplete understanding of the underlying physical processes.

The objective of this chapter is to survey the state of the art in computational fluid dynamics and heat transfer to arrive at an understanding of the types of numerical methods commonly used and the range of application of these techniques. The chapter is divided into two parts. It starts with a description of the typical governing equations for flow, heat, and mass transfer. An overview of the basic numerical solution process is then presented, including mesh generation, discretization of a typical governing equation, the solution of linear algebraic sets of equations, and the handling of nonlinearity and interequation coupling. The second half of the chapter addresses more advanced issues. An overview of unstructured-mesh methods is presented. Higher-order discretization methods are reviewed for both structured and unstructured meshes, as well as issues associated with the solution of linear algebraic equation sets for unstructured meshes. Finally, different approaches to the solution of compressible and incompressible

flows are reviewed. The chapter aims to give a broad overview of the central ideas, and subsequent chapters in the book amplify and expand on these ideas.

1.2 GOVERNING EQUATIONS

Industrial CFD simulations typically involve the solution of flows with heat transfer, species transport, and chemical reactions. These types of flows are described by the equations of mass, momentum, and energy conservation. For turbulent flows, it is common to use the Reynolds-averaged form of the governing equations in conjunction with a suitable turbulence model. Additional equations, such as those for radiative transport or for specialized combustion models, are also used. Typical Reynolds-averaged governing equations for turbulent flow and heat and mass transfer are presented in the following sections.

1.2.1 Continuity Equation

The Reynolds-averaged mixture continuity equation for the gas phase is

$$\frac{\partial}{\partial t}(\rho) + \nabla \cdot (\rho \mathbf{V}) = S_m \quad (1.1)$$

Here t is time, ρ is the Reynolds-averaged mixture density, $\mathbf{V}$ is the Reynolds-averaged velocity vector, and S_m represents external mass sources. Typically, these would result from mass-transfer interactions from a dispersed phase such as spray droplets or coal particles.

1.2.2 Momentum Equation

The Reynolds-averaged gas-phase momentum equation is

$$\frac{\partial}{\partial t}(\rho \mathbf{V}) + \nabla \cdot (\rho \mathbf{V} \mathbf{V}) + \nabla p = \nabla \cdot [(\mu + \mu_t) \nabla \mathbf{V}] + \mathbf{F} \quad (1.2)$$

Here, p is pressure, μ is the molecular viscosity, and μ_t is the turbulent viscosity, obtained from a turbulence model. $\mathbf{F}$ contains those parts of the stress term not shown explicitly, as well as other momentum sources, such as drag from the dispersed phase.

1.2.3 Energy Equation

Heat transfer is governed by the energy conservation equation

$$\frac{\partial}{\partial t}(\rho E) + \nabla \cdot (\rho \mathbf{V} E) = \nabla \cdot [(k + k_t) \nabla T] + \nabla \cdot (\boldsymbol{\tau} \cdot \mathbf{V}) - \nabla \cdot (p \mathbf{V}) + S_r + S_h \quad (1.3)$$

Here, k is the thermal conductivity and k_t is the turbulent thermal conductivity resulting from the turbulence model. $\boldsymbol{\tau}$ is the stress tensor, p is the pressure, and E is the total energy per unit mass defined as

$$E = e(T) + \frac{\mathbf{V} \cdot \mathbf{V}}{2} \quad (1.4)$$

and e is the internal energy per unit mass. The terms on the LHS of Eq. (1.3) describe the temporal evolution and the convective transfer of total energy. The first three terms on the RHS represent the conductive transfer, viscous dissipation and pressure work, respectively. S_r is the volumetric source term due to radiative heat transfer. In the present form of the energy equation, reaction source terms are included in S_h , which also contains all other volumetric heat sources, including those due to the presence of a dispersed phase.

1.2.4 Species Transport

Under the dilute approximation, the Reynolds-averaged conservation equation for the mass fraction, m_l , of specie l can be written as

$$\frac{\partial}{\partial t}(\rho m_l) + \nabla \cdot (\rho \mathbf{V} m_l) = \nabla \cdot \left(\left(\rho D + \frac{\mu_t}{\sigma_m} \right) \nabla m_l \right) + R_l \quad (1.5)$$

Here, D is the diffusion coefficient of specie l in the mixture, σ_m is the turbulent Schmidt number, and R_l is the volumetric source of the specie l due to chemical reactions.

1.2.5 General Scalar Transport Equation

The equations governing the transport of mass, momentum, energy, and chemical species may be cast into the form of a generic scalar transport equation [1] as

$$\frac{\partial (\rho \phi)}{\partial t} + \nabla \cdot (\rho \mathbf{V} \phi) = \nabla \cdot (\Gamma \nabla \phi) + S_\phi \quad (1.6)$$

Here, ϕ is the transport variable, Γ is the diffusion coefficient, and S_ϕ is the source term. Each governing equation represents a different choice of ϕ , Γ , and S_ϕ . Table 1.1 shows the values of ϕ , Γ , and S_ϕ corresponding to the governing equations shown in the previous sections. Different choices for these values may be made in the case of the energy equation. Here, the convective terms suggest a choice of $\phi = E$; however, the diffusion term is most naturally written in terms of the temperature and suggests $\phi = T$. For an incompressible substance or a perfect gas at low speeds, the equation of state $de = C_v dT$ may be invoked to obtain the choices listed in Table 1.1. A detailed discussion of alternative choices may be found in [2]. Once the governing equations are cast into the form of Eq. (1.6), a single numerical method may be devised to solve them.

TABLE 1.1 Choice of ϕ , Γ , and S_ϕ for Governing Equations

Equation	ϕ	Γ	S_ϕ
<i>Continuity</i>	1	0	S_m
<i>X momentum</i>	u	$\mu + \mu_t$	F_x
<i>Energy</i>	E	$\frac{k + k_t}{C_v}$	$\nabla \cdot (\tau \cdot \mathbf{V})$ $-\nabla \cdot (p \mathbf{V}) + S_r + S_h$

It is important to note that Eq. (1.6) has been written in *conservative form*. In contrast, by using the continuity equation, we may write the *nonconservative* form of Eq. (1.6) as

$$\rho \frac{\partial \phi}{\partial t} + \rho \mathbf{V} \cdot \nabla \phi = \Gamma \nabla \cdot \nabla \phi + \nabla \Gamma \cdot \nabla \phi + S_\phi \quad (1.7)$$

Though Eq. (1.7) is mathematically equivalent to Eq. (1.6), the two forms can yield numerical schemes with substantially different properties. Numerical schemes that seek to preserve the conservation property in the discretization start with the conservative form, Eq. (1.6), as the basis.

1.3 ANATOMY OF A NUMERICAL SOLUTION

In this section, the basic components of typical numerical solution procedures used to discretize and solve the general scalar transport equation are described. These include domain discretization, discretization of one or more governing equations of interest, and, finally, the solution of the resulting discrete algebraic equations.

1.3.1 Domain Discretization

The physical domain is discretized by meshing it, i.e., by dividing the domain into smaller, usually polyhedral, volumes. Though many variants exist, for the purposes of this chapter, the terminology shown in Fig. 1.1 will be used to describe the meshes. The fundamental unit of the mesh is the *cell* (sometimes called the element). Associated with each cell is the *cell centroid*. A cell is surrounded by *faces*, which meet at *nodes* or *vertices*. In three dimensions, the face is a surface surrounded by *edges*. In two dimensions, faces and edges are the same. A variety of mesh types are encountered in practice. These are described next.

FIGURE 1.1 Mesh terminology.

Regular and Body-fitted Meshes In many cases, our interest lies in analyzing domains that are regular in shape: rectangles, cubes, cylinders, and spheres. These shapes can be meshed by regular grids, as shown in Fig. 1.2a. The grid lines are orthogonal to each other, and conform to the boundaries of the domain. These meshes are also sometimes called orthogonal meshes.

For many practical problems, however, the domains of interest are irregularly shaped and regular meshes may not suffice. An example is shown in Fig. 1.2b. Here, grid lines are not necessarily orthogonal to each other, and curve to conform to the irregular geometry. If regular grids are used in these geometries, *stair stepping* occurs at domain boundaries, as shown in Fig. 1.3a. When the physical phenomena at the boundary are important in determining the solution, e.g., in flows dominated by wall shear, such an approximation of the boundary may not be acceptable.

FIGURE 1.2 (a) Regular and (b) body-fitted meshes.

FIGURE 1.3 (a) Stair-stepped and (b) block-structured meshes.

Structured, Block-structured, and Unstructured Meshes The meshes shown in Fig. 1.2 are examples of *structured* meshes. Here, every interior vertex in the domain is connected to the same number of neighbor vertices. Figure 1.3b shows a *block-structured* mesh. Here, the mesh is divided into blocks, and the mesh within each block is structured. However, the arrangement of the blocks themselves is not necessarily structured. Figure 1.4a shows an *unstructured* mesh. Here, each vertex is connected to an arbitrary number of neighbor vertices. Unstructured meshes impose fewer topological restrictions on the user, and, as a result, make it easier to mesh very complex geometries.

Conformal and Nonconformal Meshes An example of a nonconformal mesh is shown in Fig. 1.4b. Here, the vertices of a cell or element may fall on the faces of neighboring cells or elements. In contrast, the meshes in Figs. 1.2, 1.3, and 1.4a are conformal meshes. Here, the vertices of cells sharing a common face are coincident.

Cell Shapes Many modern finite-volume and finite-element techniques support a variety of cell and element shapes. The most widely used are quadrilaterals and hexahedra. Methods for generating good-quality structured meshes for quadrilaterals and hexahedra have existed for some time now [3]. Though mesh structure imposes restrictions on the complexity of the geometries that can be meshed, structured quadrilaterals and hexahedra are well-suited for flows with a dominant direction, such as boundary-layer flows. More recently, unstructured meshes are becoming necessary to handle the complex geometries that characterize industrial problems. Here, triangles and tetrahedra are increasingly being used, and techniques for their generation are rapidly reaching maturity [3]. Another recent trend is the use of hybrid meshes. For example, prisms are used in boundary layers, transitioning to tetrahedra in the free stream.

Node-based and Cell-based Schemes Numerical methods that store their primary unknowns at the node or vertex locations are called *node-based* or *vertex-based* schemes. Those that store them at the cell centroid, or associate them with the cell, are called *cell-based* schemes. Finite-element methods are node-based [4]. Many finite-volume methods are cell-based [1, 5, 6], though node-based finite-volume schemes are also available [7].

FIGURE 1.4 (a) Unstructured and (b) nonconformal meshes.

1.3.2 Discretization of Governing Equation

The most commonly used approaches to discretize the general scalar transport equation are the finite-difference, finite-volume, and finite-element techniques. These methods discretize the governing equations directly, using a variety of local profile assumptions or approximations, reducing the original partial differential equation into a set of coupled algebraic equations, which must then be solved. In contrast, the boundary-element technique [8], which has been used in a variety of heat conduction problems [9], invokes Green's identities to convert the original differential equation into an integral equation involving only surface quantities, which is then discretized and solved. A detailed description of the technique may be found in Chapter 4. Here attention is directed to techniques that directly discretize the governing equations.

To illustrate the similarities and differences between finite-difference, finite-element, and finite-volume techniques, consider a one-dimensional scalar transport equation with a constant diffusion coefficient and no unsteady or convective terms

$$\frac{d}{dx} \left(\Gamma \frac{d\phi}{dx} \right) + S_\phi = 0 \quad (1.8)$$

with boundary conditions $\phi(0) = \phi_0$ and $\phi(L) = \phi_L$. Equation (1.8), which has been written in conservation form, will be discretized using each of the three methods.

Finite-difference Methods Finite-difference methods approximate the derivatives in the governing differential equation using truncated Taylor series expansions. First Eq. (1.8) is recast in nonconservation form:

$$\Gamma \frac{d^2\phi}{dx^2} + S_\phi = 0 \quad (1.9)$$

Next, the discretization of the diffusion term is carried out. Consider the one-dimensional mesh shown in Fig. 1.5. The unknown discrete values of ϕ are stored at the nodes shown. The Taylor series expansion for ϕ can be written as

$$\phi_1 = \phi_2 - \Delta x \left(\frac{d\phi}{dx} \right)_2 + \frac{\Delta x^2}{2} \left(\frac{d^2\phi}{dx^2} \right)_2 + O(\Delta x^3) \quad (1.10)$$

and

$$\phi_3 = \phi_2 + \Delta x \left(\frac{d\phi}{dx} \right)_2 + \frac{\Delta x^2}{2} \left(\frac{d^2\phi}{dx^2} \right)_2 + O(\Delta x^3) \quad (1.11)$$

FIGURE 1.5 One-dimensional mesh.

The term $O(\Delta x^3)$ indicates that the terms that follow have a dependence on Δx^n where $n \geq 3$. Subtracting Eq. (1.10) from Eq. (1.11) gives

$$\left(\frac{d\phi}{dx}\right)_2 = \frac{\phi_3 - \phi_1}{2\Delta x} + O(\Delta x^2) \quad (1.12)$$

The addition of the two equations yields

$$\left(\frac{d^2\phi}{dx^2}\right)_2 = \frac{\phi_1 + \phi_3 - 2\phi_2}{\Delta x^2} + O(\Delta x^2) \quad (1.13)$$

By including the diffusion coefficient and dropping terms of $O(\Delta x^2)$ or smaller, the following equation is obtained:

$$\Gamma \left(\frac{d^2\phi}{dx^2}\right)_2 = \Gamma \frac{\phi_1 + \phi_3 - 2\phi_2}{\Delta x^2} \quad (1.14)$$

The second derivative of ϕ has thus been evaluated to second order.

The source term S_ϕ is evaluated at the point 2 using

$$S_2 = S_\phi(\phi_2) \quad (1.15)$$

Substituting Eqs. (1.14) and (1.15) into Eq. (1.8) yields

$$\frac{2\Gamma}{\Delta x^2}\phi_2 = \frac{\Gamma}{\Delta x^2}\phi_1 + \frac{\Gamma}{\Delta x^2}\phi_3 + S_2 \quad (1.16)$$

This is a discrete form of Eq. (1.8). By deriving a similar equation like for every grid point in the mesh, a set of algebraic equations in the discrete values of ϕ are obtained. The value of ϕ at each node is directly influenced only by its nearest neighbors; the use of a truncated Taylor series leads to this type of *local* dependence. At the boundaries, the discrete values of ϕ may be obtained by discretizing the boundary conditions. The resulting equation set may be solved by a variety of methods, which are discussed later in this chapter.

Finite-difference methods do not explicitly enforce the conservation principle in deriving discrete equations. Thus, energy balance may not be exactly satisfied for coarse meshes, though finite-difference methods that have the consistency property [10] are guaranteed to approach perfect conservation as the mesh is refined. As we will see, finite-difference methods yield discrete equations that look similar to finite-volume and finite-element methods for simple cases; however, this similarity not guaranteed in more complicated cases.

Finite-element Methods To develop the finite-element method, the one-dimensional diffusion equation Eq. (1.8), is reconsidered. There are different kinds of finite-element methods of which the method of weighted residuals is one. Here, a popular variant of the method of weighted residuals, called the Galerkin finite-element method, is considered. More detailed information about this class of numerical techniques may be found in [4, 11, 12].

The starting point is, again, the nonconservative form of the governing equation (1.9). The computational domain is divided into $N - 1$ elements corresponding to N nodes; a typical

element i is shown in Fig. 1.6. Let $\bar{\phi}$ be an approximation to ϕ . Since $\bar{\phi}$ is only an approximation, it does not satisfy Eq. (1.9) exactly, so that there is a residual R

$$R = \Gamma \frac{d^2 \bar{\phi}}{dx^2} + S_\phi \quad (1.17)$$

We wish to find $\bar{\phi}$ such that

$$\int_{\text{domain}} W R dx = 0 \quad (1.18)$$

where W is a *weight function*. Equation (1.18) requires that the residual R become zero in a weighted sense. To generate a set of discrete equations, a family of weight functions W_j , $j = 1, 2, \dots, N$ is used. Thus,

$$\int_{\text{domain}} W_j R dx = 0 \quad j = 1, 2, \dots, N \quad (1.19)$$

The weight functions $W_j(x)$ are typically local in that they are nonzero in the vicinity of node j , but are zero everywhere else in the domain. Further, a *shape function* $N_j(x)$ is assumed for $\bar{\phi}$, which specifies how $\bar{\phi}$ varies between nodes. Thus,

$$\bar{\phi}(x) = \sum_{i=1}^N N_i(x) \phi_i \quad (1.20)$$

The Galerkin finite-element method requires that the weight and shape functions be the same, i.e., $W_i = N_i$. Typically, the shape function variation is also defined locally, as shown for the case of a linear shape function in Fig. 1.6. Here, for $x_i \leq x \leq x_{i+1}$,

$$N_i(x) = \frac{x_{i+1} - x}{x_{i+1} - x_i}$$

$$N_{i+1}(x) = \frac{x - x_i}{x_{i+1} - x_i}$$

FIGURE 1.6 Linear shape functions on element i and corresponding variation of ϕ .

Furthermore, the source term S_ϕ is also interpolated on the domain from

$$S(x) = \sum_{i=1}^N N_i(x) S_{\phi,i} \quad (1.21)$$

Thus, under the Galerkin finite-element formulation, Eq. (1.9) becomes

$$\int_{x_0}^{x_L} \Gamma N_j(x) \frac{d^2 \bar{\phi}}{dx^2} dx + \int_{x_0}^{x_L} N_j(x) S dx = 0 \quad j = 1, 2, \dots, N \quad (1.22)$$

The next step is to integrate the first term in Eq. (1.22) by parts. This procedure yields

$$\Gamma N_j(x) \left. \frac{d\bar{\phi}}{dx} \right|_{x_0}^{x_L} - \int_{x_0}^{x_L} \Gamma \frac{dN_j}{dx} \frac{d\bar{\phi}}{dx} dx + \int_{x_0}^{x_L} N_j(x) S dx = 0 \quad j = 1, 2, \dots, N \quad (1.23)$$

Furthermore, Eq. (1.20) may be differentiated to yield

$$\frac{d\bar{\phi}}{dx} = \sum_{i=1}^N \frac{dN_i}{dx} \phi_i \quad (1.24)$$

The first term in Eq. (1.23) is

$$N_j(x_0)q_1 + N_j(x_L)q_N \quad (1.25)$$

Here q_1 and q_N are the heat fluxes into the domain at the boundaries. Here $N_1(x_0) = N_N(x_L) = 1$ by definition. If the shape function $N_j(x)$ is local, it is nonzero only in the vicinity of the node j . Thus, Eq. (1.25) becomes

$$\begin{aligned} N_j(x_0)q_1 + N_j(x_L)q_N &= q_1 & \text{if } j &= 1 \\ &= 0 & \text{if } j &= 2, \dots, N-1 \\ &= q_N & \text{if } j &= N \end{aligned} \quad (1.26)$$

Thus, the overall equation may be written as

$$\begin{aligned} \int_{x_0}^{x_L} \Gamma \frac{dN_j}{dx} \sum_{i=1}^N \frac{dN_i}{dx} \phi_i dx - \int_{x_0}^{x_L} N_j(x) \sum_{i=1}^N N_i(x) S_i dx &= N_j(x_0)q_1 + N_j(x_L)q_N \\ & \quad j = 1, 2, \dots, N \end{aligned} \quad (1.27)$$

The discrete equation for a node j may thus be written as

$$\begin{aligned} \sum_{i=1}^N K_{ij} \phi_i + \mathcal{S}_j &= q_1 & \text{if } j = 1 \\ &= 0 & \text{if } j = 2, \dots, N-1 \\ &= q_N & \text{if } j = N \end{aligned} \quad (1.28)$$

Here

$$\begin{aligned} K_{ij} &= \int_{x_0}^{x_L} \frac{dN_i}{dx} \frac{dN_j}{dx} dx \\ \mathcal{S}_j &= - \int_{x_0}^{x_L} N_j(x) \sum_{i=1}^N N_i(x) S_i dx \end{aligned} \quad (1.29)$$

In the above equations, when ϕ_0 and ϕ_L are given, the equations at nodes $j = 1$ and $j = N$ may be used to evaluate the fluxes q_1 and q_N . On the other hand, when q_1 and q_N are specified, the same equations are used to find ϕ_0 and ϕ_L . By choosing specific shape functions $N_i(x)$, a coupled algebraic equation set may be derived for the nodal values ϕ_i . Since N_i is local, the matrix K_{ij} is sparse. It is important to note that because the Galerkin finite-element method requires the residual to be zero only in a weighted sense, it does not enforce the conservation principle in its original form; like the finite-difference method, conservation is satisfied in the limit of a fine-enough mesh. Next, attention is turned to a method that employs conservation as a tool for developing discrete equations.

Finite-volume Methods The finite-volume method (sometimes called the control-volume method) divides the domain in to a finite number of nonoverlapping cells or control volumes over which conservation of ϕ is enforced in a discrete sense. The starting point is the conservative form of the scalar transport equation (1.8). Consider a one-dimensional mesh, with cells as shown in Fig. 1.7. Discrete values of ϕ are stored at cell centroids, which are denoted by W , P , and E . Far neighbor cells WW and EE are shown for later use. The cell faces of cell P are denoted by w and e . The face areas are assumed to be unity.

The focus is on the cell associated with P . Equation (1.8) is integrated over the cell P , yielding

$$\int_w^e \frac{d}{dx} \left(\Gamma \frac{d\phi}{dx} \right) dx + \int_w^e S dx = 0 \quad (1.30)$$

FIGURE 1.7 Arrangement of control volumes.

which can be integrated to give

$$\left(\Gamma \frac{d\phi}{dx}\right)_e - \left(\Gamma \frac{d\phi}{dx}\right)_w + \int_w^e S dx = 0 \quad (1.31)$$

This equation can also be obtained by writing a diffusion flux balance over the cell P from first principles. Thus far, no approximation has been made.

A profile assumption, i.e., an assumption about how ϕ varies between cell centroids, is now made. If it is assumed that ϕ varies linearly between cell centroids, we may write

$$\frac{\Gamma_e (\phi_E - \phi_P)}{\delta x_e} - \frac{\Gamma_w (\phi_P - \phi_W)}{\delta x_w} + \bar{S} \Delta x = 0 \quad (1.32)$$

Here $\bar{S}$ is the average value of S in the control volume. Note that the above equation is no longer exact because of the approximation in assuming that ϕ varies in a piecewise linear fashion between cell centroids.

Collecting terms, Eq. (1.32) becomes

$$a_P \phi_P = a_E \phi_E + a_W \phi_W + b \quad (1.33)$$

where

$$\begin{aligned} a_E &= \frac{\Gamma_e}{\delta x_e} \\ a_W &= \frac{\Gamma_w}{\delta x_w} \\ a_P &= a_E + a_W \\ b &= \bar{S} \Delta x \end{aligned} \quad (1.34)$$

Equations similar to Eq. (1.33) may be derived for all cells in the domain, yielding a set of algebraic equations, as before. These may be solved using a variety of direct or iterative methods.

Unlike finite-difference and finite-element methods, the finite volume discretization process starts with the statement of conservation over the cell. Cell values of ϕ that satisfy this conservation statement are then found. Thus, conservation is guaranteed for each cell, regardless of mesh size. Conservation does not guarantee accuracy, however; accuracy depends on the profile assumptions made. The solution for ϕ may be inaccurate, but will, nevertheless, be conservative.

1.3.3 Solution of Linear Equations

Regardless of what method is used, the process of discretization leads to a coupled algebraic set of equations in the discrete values of ϕ , such as Eq. (1.33). These equations may be linear (i.e., the coefficients are independent of ϕ) or they may be nonlinear (i.e., the coefficients are functions of ϕ). The techniques for solving these equations are independent of the discretization method and represent the *path to solution*. If the problem is well-posed and the discrete equation set is linear, it is guaranteed that only one solution exists, and all linear solvers that converge to a solution will lead to the same discrete solution. The accuracy of the solution depends only on the accuracy of the discretization technique.

Solution methods may be broadly classified as direct or iterative. Each class is considered in turn.

Direct Methods The discrete algebraic equations derived in the previous sections may be written as

$$\mathbf{A}\phi = \mathbf{B} \quad (1.35)$$

where $\mathbf{A}$ is the coefficient matrix, $\phi = [\phi_1, \phi_2, \dots]^T$ is a vector consisting of the discrete values of ϕ , and $\mathbf{B}$ is the vector resulting from the source terms.

Direct methods solve Eq. (1.35) using the standard methods of linear algebra. The simplest direct method is inversion, whereby ϕ is computed from

$$\phi = \mathbf{A}^{-1}\mathbf{B} \quad (1.36)$$

A solution for ϕ is guaranteed if $\mathbf{A}^{-1}$ can be found. However, the operation count for the inversion of an $N \times N$ matrix is $O(N^3)$. Consequently, inversion is almost never employed in practical problems. More efficient methods for linear systems are available. For the discretization methods of interest here, $\mathbf{A}$ is sparse, and for structured meshes it is banded. For certain types of equations, for example, for pure diffusion, the matrix is symmetric. Matrix manipulation can take into account the special structure of $\mathbf{A}$ in devising efficient solution techniques for Eq. (1.35). A number of standard textbooks describe direct solution techniques, Ref. [10], for example.

Iterative Methods Iterative methods are widely used in computational fluid dynamics. These methods employ a guess-and-correct philosophy, which progressively improves the guessed solution by repeated application of the discrete equations. Let us consider an extremely simple iterative method, the Gauss-Seidel method. Here, each grid point in the mesh is visited sequentially, and the value of ϕ is updated using

$$\phi_P = \frac{a_E\phi_E + a_W\phi_W + b}{a_P} \quad (1.37)$$

The neighbor values, ϕ_E and ϕ_W , are required and are assumed known at prevailing values. Thus, points that have already been visited will have recently updated values of ϕ , and those that have not will have old values. The domain is swept over and over until convergence. A related technique, Jacobi iteration, employs only old values during the sweep, updating all grid points simultaneously at the end of the sweep.

Convergence of the process is guaranteed for linear problems if the *Scarborough criterion* is satisfied, which requires

$$\begin{aligned} \frac{|a_E| + |a_W|}{|a_P|} &\leq 1 && \text{for all grid points} \\ &< 1 && \text{for at least one grid point} \end{aligned} \quad (1.38)$$

Matrices that satisfy the Scarborough criterion are said to be *diagonally dominant*. Direct methods do not require the Scarborough criterion to be satisfied; a solution to the linear set of equations can always be obtained as long as the coefficient matrix is not singular.

The Gauss-Seidel scheme can be implemented with very little storage. All that is required is storage for the discrete values of ϕ at the grid points. The coefficients a_P , a_E , a_W , and b can be computed on the fly if desired, since the entire coefficient matrix for the domain is not required when updating the value of ϕ at any grid point. Also, the iterative nature of the scheme makes it particularly suitable for nonlinear problems. If the coefficients depend on ϕ ,

they may be updated using prevailing values of ϕ as the iterations proceed. Furthermore, the Gauss-Seidel technique can be applied to sparse matrices with arbitrary fill patterns and does not require a band structure. Nevertheless, it is rarely used in practice because of slow convergence; techniques to accelerate its convergence are discussed in Section 1.6.

An alternative to solving the linear system is use a time-advancement strategy. Here, even though the desire is to solve the steady-state problem, the problem is posed as unsteady, and the solution marched to steady state. If an explicit scheme is used [1], no linear solver is necessary; however, the time step is limited by the stability limits imposed by explicit schemes. If an implicit time-stepping strategy is used [1], linear solvers are again necessary.

1.3.4 Nonlinearity and Coupling

In many engineering applications it is necessary to solve a number of governing equations simultaneously over the computational domain. In solving natural convection problems, for example, the flow field and the energy equation must be solved simultaneously. The solution of the flow field itself requires the simultaneous solution of the continuity and momentum equations. In addition, the governing equations may be nonlinear.

The simplest approach to solving coupled sets of governing equations is the *sequential* approach [1]. Here each governing equation is discretized and solved in turn using the procedures described previously. Prevailing values of the other solution variables are used where necessary. Governing equations are iterated upon in this way until the solution is deemed converged. This approach has been used widely for the solution of incompressible flows using pressure-based algorithms.

When the coupling between governing equations becomes strong, this type of sequential solution procedure can become untenably slow, and may even lead to instability and divergence. When computer memory and cost are not a limitation, it is possible to discretize all the governing equations at each node or cell centroid, and solve the complete nonlinear system using the Newton-Raphson method or other techniques [10, 13].

1.3.5 Properties of Numerical Solution Procedure

The discretization and solution procedures described here may be characterized by their accuracy, consistency, stability, and convergence characteristics. A discussion of these four characteristics now follows.

Accuracy Errors in the computed solution may result from (1) *modeling* errors, i.e., errors engendered by incorrectly representing the physics in the governing equations, (2) a lack of convergence in the iterative solution procedure, or (3) the truncation error in the discretization procedure. As was seen in Section 1.3.2, $d^2\phi/dx^2$ may be represented as

$$\Gamma\left(\frac{d^2\phi}{dx^2}\right)_2 = \Gamma\frac{\phi_1 + \phi_3 - 2\phi_2}{\Delta x^2} \quad (1.39)$$

The truncation error for this representation is $O(\Delta x^2)$. The error decreases quadratically with Δx . A scheme whose truncation error is $O(\Delta x^n)$ is an n th-order scheme.

Consistency A discretization scheme is consistent if the error in the solution tends to zero as $\Delta x \rightarrow 0$. If the truncation error is of the form $O(\Delta x^n)$, consistency is guaranteed. A numerical scheme for unsteady problems that has a truncation error $O(\Delta x/\Delta t)$, for example, would not be consistent unless $\Delta x/\Delta t \rightarrow 0$. Consistency is an important property of the discretization since it ensures that refining the mesh (or the time step) will yield more accurate solutions.

Stability A stable numerical scheme for a steady problem is one that can converge to a solution to the discrete equation set. For an unsteady problem, the time-stepping scheme employed must result in a bounded solution in time if the physical problem being solved has a bounded solution.

Convergence The term *convergence* is used in two senses in the CFD literature. On the one hand, one may speak of convergence to a mesh-independent solution by repeated mesh refinement. On the other hand, the term may also imply convergence to a stable solution during iteration or time marching.

1.3.6 Summary and Discussion

Thus far, the most important components of the basic solution process have been examined. It has been shown that the equations governing heat, mass, and momentum transfer have a common form represented by the general scalar transport equation. The most commonly used methods to discretize this set of equations are the finite-difference, finite-element, and finite-volume methods. The first two methods are not conservative on coarse meshes. On the other hand, the finite-volume method is predicated upon the conservation principle and guarantees perfect conservation even for coarse meshes. All discretization methods result in coupled sets of algebraic equations, which may be solved by either direct or iterative methods; the latter have emerged as the preferred alternative over the last two decades. Nonlinearity and coupling may be handled either through a simple segregated approach or through a coupled approach using Newton-Raphson or other direct-solution techniques.

Attention is next turned to more advanced topics in CFD. In the next few sections, we examine in greater depth emerging unstructured-mesh techniques for addressing complex geometries, higher-order discretization schemes, solvers for linear algebraic equations on unstructured meshes, as well as issues related to pressure–velocity coupling for compressible and incompressible flows.

1.4 COMPUTATIONAL TECHNIQUES FOR UNSTRUCTURED MESHES

Over the last two decades, there has been a concerted effort to develop CFD techniques to address realistic industrial geometries. Early efforts [5, 14–16] concentrated on body-fitted meshes, either single block or multiblock, using either covariant or contravariant velocity formulations. Both staggered and colocated formulations were pursued [5, 17]. However, it quickly became apparent that these approaches, though useful, could not address very complex geometries, and that far more flexibility in mesh connectivity was required.

Two distinct approaches have emerged over the last two decades to address this need. Finite-element methods [4, 18] were extended and refined to address fluid flow computations. At the same time, a number of cell- and node-based finite-volume formulations were developed [6, 7, 19–22]. Here, a typical cell-based unstructured finite volume formulation is presented to illustrate the underlying principles and to highlight special issues that arise with respect to developing linear solvers and higher-order techniques. A detailed discussion of this method may be found in [6].

1.4.1 Discretization of Convection–Diffusion Equation

The starting point of the development is the general scalar transport equation in conservation form, given by Eq. (1.6). The domain is discretized into arbitrary unstructured convex polyhedral

FIGURE 1.8 Control volume in unstructured mesh.

cells, as shown in Fig. 1.8, and conservation is enforced on these cells. In cell-based schemes, all transport variables are stored at cell centroids. One advantage of this arrangement is that conservation can be ensured for arbitrary control volumes with nonconformal interfaces without special interpolation techniques. Consider the mesh shown in Fig. 1.8, for example. Cell C1 can be considered to have five faces, $a-b-c-d-e$, and no special treatment is required. Another advantage is that on triangular and tetrahedral meshes, the ratio of the number of cells to nodes is between three and five. As a result, cell-based storage enjoys better resolution than node-based storage for roughly the same amount of work, which is typically proportional to the number of cell faces.

The basic development parallels the one-dimensional finite-volume example presented in Section 1.3.2 but special attention must be paid to mesh nonorthogonality and the lack of structure. Integrating Eq. (1.6) about the control volume C0 yields

$$\left[\frac{\partial}{\partial t} (\rho\phi) \right]_0 \Delta\mathcal{V}_0 + \sum_f F_f \phi_f = \sum_f D_f + (S_\phi \Delta\mathcal{V})_0 \quad (1.40)$$

where F_f is the mass flow rate out of C0 at the face f , $\Delta\mathcal{V}_0$ is the volume of the cell C0, D_f is the transport due to diffusion through the face f , and the summations are over the faces of the control volume. For the purposes of scalar transport, the mass flow rate F_f is assumed to be known. To obtain a set of algebraic equations, all other face quantities as well as volume integrals in Eq. (1.40) must be written in terms of the unknowns, i.e., values of ϕ at cell and boundary face centroids.

Diffusion Term The diffusion term across a face is given by

$$D_f = \Gamma_f \nabla \phi \cdot \mathbf{A} \quad (1.41)$$

$\mathbf{A}$ is the area vector associated with the face f . Since the line joining the centroids (associated with the vector $\hat{\mathbf{e}}_s$ in Fig. 1.8) is not perpendicular to face f , the gradient of ϕ normal to the face,

i.e., $\nabla\phi \cdot \mathbf{A}$, cannot be written purely in terms of a gradient in the $\hat{\mathbf{e}}_s$ direction. Decomposing the gradient in directions parallel to $\hat{\mathbf{e}}_s$ and tangent to $\mathbf{A}$, and using consistent approximations for derivatives, it is possible to write the diffusion term D_f as [6]

$$D_f = \Gamma_f \frac{\phi_1 - \phi_0}{ds} \frac{\mathbf{A} \cdot \mathbf{A}}{\mathbf{A} \cdot \hat{\mathbf{e}}_s} + S_f \quad (1.42)$$

where

$$S_f = \Gamma_f \left(\overline{\nabla\phi} \cdot \mathbf{A} - \overline{\nabla\phi} \cdot \hat{\mathbf{e}}_s \frac{\mathbf{A} \cdot \mathbf{A}}{\mathbf{A} \cdot \hat{\mathbf{e}}_s} \right) \quad (1.43)$$

Here $\overline{\nabla\phi}$ at the face is taken to be the average of the derivatives at the two adjacent cells, determined as discussed in Section 1.4.2. Thus, D_f is seen to consist of a *primary* diffusion term consisting of the first term in Eq. (1.42), and a *secondary diffusion* term, S_f . For orthogonal meshes, $\mathbf{A} \cdot \hat{\mathbf{e}}_s = 0$ and S_f is therefore zero. The primary component is expressed in terms of the difference of ϕ values in the two cells adjacent to face f (i.e., ϕ_0 and ϕ_1). The primary component is treated implicitly in the discrete equation for the two cells.

Convection Term On a structured mesh, a first-order approximation for the value of ϕ on the face e in Fig. 1.7 may be obtained using an *upwind* scheme as

$$\begin{aligned} \phi_e &= \phi_P & \text{if } F_e \geq 0 \\ &= \phi_E & \text{if } F_e < 0 \end{aligned} \quad (1.44)$$

Here F_e is the flow rate on the east face e , and is positive if the flow is in the positive x direction. A second-order central-difference approximation for ϕ_e may be written on a uniform mesh as

$$\phi_e = \frac{\phi_P + \phi_E}{2} \quad (1.45)$$

Similar schemes can be devised on unstructured meshes. For example, a first-order upwind approximation for ϕ at the face f can be taken to be the value at the upwind cell in Fig. 1.8:

$$\phi_f = \phi_{\text{upwind}} \quad (1.46)$$

Similarly for a uniform mesh, a central-difference approximate to ϕ_f can be written as

$$\phi_f = \frac{\phi_0 + \phi_1}{2} \quad (1.47)$$

Though higher-order schemes are generally preferred over first-order schemes in CFD, higher-order convection operators frequently result in a loss of boundedness unless specific steps are taken to limit spatial oscillations. A more complete discussion of interpolation schemes for convective operators may be found in Section 1.5.

Unsteady Term In the present numerical scheme, the unsteady term is discretized using backward differences. A first-order approximation is

$$\left[\frac{\partial}{\partial t} (\rho\phi) \right]_0 = \frac{(\rho\phi)_0^{n+1} - (\rho\phi)_0^n}{\Delta t} \quad (1.48)$$

higher-order representations of the unsteady term can be written using more levels of storage.

For unsteady problems, the discretization of the convection, diffusion, and source terms may be carried out at the previous time level n , resulting in an *explicit* scheme. Alternatively, discretizing these terms at the time level $n + 1$ results in an *implicit* scheme. Schemes such as the Crank-Nicholson scheme employ averages of both time levels [1].

Source Term The source term S_ϕ is first written in linearized form as

$$S_\phi = S_C + S_P \phi_0 \quad (1.49)$$

The forms of S_C and S_P are chosen from stability considerations [1]. As seen earlier, iterative linear solvers require diagonal dominance to converge. If such solvers are to be used, it is prudent to require S_P to be negative to improve diagonal dominance of the coefficient matrix resulting from the discretization process [1]. In most engineering problems, negative values of S_P arise naturally from the physical nature of the source term itself. A useful linearization process is to expand the source term in a truncated Taylor series about the current iterate, denoted by starred values:

$$(S_\phi)_0 = (S_\phi^*)_0 + \left(\frac{\partial S}{\partial \phi} \right)_0^* (\phi_0 - \phi_0^*) \quad (1.50)$$

By comparing Eqs. (1.49) and (1.50), S_C and S_P may be written as

$$\begin{aligned} S_C &= (S_\phi^*)_0 - \left(\frac{\partial S}{\partial \phi} \right)_0^* \phi_0^* \\ S_P &= \left(\frac{\partial S}{\partial \phi} \right)_0^* \end{aligned} \quad (1.51)$$

We note that at convergence, $\phi = \phi^*$, and the true value of S_ϕ is recovered. Thus, the linearization procedure changes the path to solution, but not the final solution itself. The linearized source term is used in Eq. (1.40).

Discrete Equation Set Collection of all the terms results in a discrete equation for each cell involving face neighbors of the cell. Using the first-order upwind approximation with an implicit time-stepping scheme, the overall discrete equation may be written as

$$a_P \phi_P = \sum_{nb} a_{nb} \phi_{nb} + b \quad (1.52)$$

where

$$\begin{aligned}
 a_{nb} &= \frac{\Gamma_f}{ds} \frac{\mathbf{A} \cdot \mathbf{A}}{\mathbf{A} \cdot \mathbf{e}_s} + \text{Max}[-F_f, 0] \\
 a_P &= \sum_{nb} a_{nb} - S_P \Delta \mathcal{V}_0 + \frac{\rho_0^n \Delta \mathcal{V}_0}{\Delta t} + \left(\sum_f F_f + \frac{\rho_0 - \rho_0^n}{\Delta t} \Delta \mathcal{V}_0 \right) \\
 b &= S_C \Delta \mathcal{V}_0 + \sum_{nb} (S_f)_{nb} + \frac{\rho_0^n \Delta \mathcal{V}_0}{\Delta t} \phi_0^n
 \end{aligned} \tag{1.53}$$

Here nb denotes the cell-centroid values associated with the face neighbor cells. Since only face neighbors appear directly in the discrete equation, the resulting coefficient matrix is sparse. Other neighbor values appear indirectly in S_f in the computation of the gradient of ϕ , but do not appear in the coefficient matrix. The superscript $n + 1$ has been dropped for clarity. Thus, the unsuperscripted terms are to be understood as being evaluated at time level $n + 1$.

1.4.2 Gradient Calculation

Accurate computation of ϕ gradients is an important part of any unstructured mesh technique. Computation of secondary gradient terms requires the knowledge of gradients of ϕ at the cell centroids. Gradients are also required for the construction of higher-order convection operators (see Section 1.5) as well as in many physical models. For example, velocity derivatives are required to compute the production term in turbulence models or to compute the strain rate for non-Newtonian viscosity models. Unlike for structured grids, these cannot be obtained by simple finite differences. Classical finite-element methods and control-volume finite-element methods [7] address this by analytically differentiating the underlying shape functions. Cell-based finite-volume methods have typically employed two different approaches to gradient calculation, which are now presented.

Gradient Theorem Approach One approach is suggested by the gradient theorem which states that for any closed volume $\Delta \mathcal{V}_0$ enclosed by surface A

$$\int_{\Delta \mathcal{V}_0} \nabla \phi d\mathcal{V} = \int_A \phi d\mathbf{A} \tag{1.54}$$

where $d\mathbf{A}$ is the outward-pointing incremental area vector. A discrete version of Eq. (1.54) may be written as

$$\nabla \phi = \frac{1}{\Delta \mathcal{V}_0} \sum_f \phi_f \mathbf{A}_f \tag{1.55}$$

where $\mathbf{A}_f$ is the outward-pointing face area vector for face f . As a first approximation, the face value ϕ_f may be computed as the average of the two cells values sharing the face so that

$$\phi_f = \frac{\phi_0 + \phi_1}{2} \tag{1.56}$$

FIGURE 1.9 Arrangement of cells in unstructured mesh.

Once the derivative has been obtained by using Eqs. (1.55) and (1.56), the initial approximation of the face average value of ϕ may be successively improved by *reconstructing* it from the cell value. Thus, from Fig. 1.9, ϕ_f may be written as

$$\phi_f = \frac{(\phi_0 + \nabla\phi_0 \cdot \Delta\mathbf{r}_0) + (\phi_1 + \nabla\phi_1 \cdot \Delta\mathbf{r}_1)}{2} \quad (1.57)$$

By iteratively applying Eq. (1.57) to the gradient calculation in Eq. (1.55), the accuracy of the computed gradient may be improved. Iteration increases the effective stencil of ϕ values appearing in the discrete equation and can lead to oscillatory results. In practice, the gradients used to reconstruct face values are *limited* to the bounds dictated by neighbor ϕ values so as to avoid undershoots and overshoots in the solution. The concept of *limiting* is discussed in Section 1.5.2.

Least-squares Approach The least-squares approach computes the gradient at a cell such that it reconstructs the solution in the neighborhood of the cell in a least-squares sense. For example, consider cell C0. It would be desirable to have the value of ϕ computed at the centroid of neighbor cell Cj in Fig. 1.10 be equal to ϕ_j . By assuming a locally linear variation of ϕ , one may write

$$\phi_0 + \nabla\phi_0 \cdot \Delta\mathbf{r}_j = \phi_j \quad (1.58)$$

Here $\Delta\mathbf{r}_j$ is the vector from the centroid of cell C0 to the centroid of cell Cj. Substituting for $\Delta\mathbf{r}_j$ in Eq. (1.58) yields

$$\Delta x_j \left. \frac{\partial\phi}{\partial x} \right|_0 + \Delta y_j \left. \frac{\partial\phi}{\partial y} \right|_0 = \phi_j - \phi_0 \quad (1.59)$$

for all cells Cj, $j = 1, \dots, J$ surrounding C0.

It is convenient to assemble all the equations in matrix form as follows:

$$\mathbf{M}\mathbf{d} = \Delta\phi \quad (1.60)$$

FIGURE 1.10 Nodal locations and vectors used in least-squares calculation of cell gradient.

Here $\mathbf{M}$ is the $J \times 2$ matrix

$$\mathbf{M} = \begin{bmatrix} \Delta x_1 & \Delta y_1 \\ \Delta x_2 & \Delta y_2 \\ \vdots & \vdots \\ \Delta x_J & \Delta y_J \end{bmatrix} \quad (1.61)$$

and $\mathbf{d}$ is the vector of the components of gradients of ϕ at cell C0

$$\mathbf{d} = \begin{bmatrix} \left. \frac{\partial \phi}{\partial x} \right|_0 \\ \left. \frac{\partial \phi}{\partial y} \right|_0 \end{bmatrix} \quad (1.62)$$

and $\Delta\phi$ is the vector of the differences of ϕ

$$\Delta\phi = \begin{bmatrix} \phi_1 - \phi_0 \\ \phi_2 - \phi_0 \\ \vdots \\ \phi_J - \phi_0 \end{bmatrix} \quad (1.63)$$

Equation (1.60) represents J equations in the two unknowns $(\partial\phi/\partial x)|_0$ and $(\partial\phi/\partial y)|_0$. Since, in general, J is larger than two, Eq. (1.60) is an overdetermined system. Physically,

this means that a linear profile cannot be assumed for ϕ in the vicinity of cell C0 that exactly reconstructs the known solution at all of its neighbors. One can only hope to find a solution that fits the data in the best possible way, i.e., a solution for which the root mean square (rms) value of the difference between the neighboring cell values and the reconstructed values is minimized. From Eq. (1.59), the difference in the reconstructed value and the cell value for cell Cj is

$$R_j = \Delta x_j \left. \frac{\partial \phi}{\partial x} \right|_0 + \Delta y_j \left. \frac{\partial \phi}{\partial y} \right|_0 - (\phi_j - \phi_0) \quad (1.64)$$

The sum of the squares of the errors over all the neighboring cells is

$$R = \sum_j R_j^2 \quad (1.65)$$

The objective is to find $(\partial \phi / \partial x)|_0$ and $(\partial \phi / \partial y)|_0$ such that R is minimized. By differentiating R with respect to $(\partial \phi / \partial x)|_0$ and $(\partial \phi / \partial y)|_0$ and equating to zero, we obtain

$$\mathbf{M}^T \mathbf{M} \mathbf{d} = \mathbf{M}^T \Delta \phi \quad (1.66)$$

$\mathbf{M}^T \mathbf{M}$ is a 2×2 matrix that can easily be inverted analytically to yield the required gradient $\nabla \phi$. The least-squares approach is easily extended to three dimensions. This method places no restrictions on cell shape, and does not require a structured mesh.

1.4.3 Summary and Discussion

In this section, an overview of typical unstructured, cell-based, finite-volume techniques has been presented. These techniques involve conservation, albeit over arbitrary polyhedral cells. To obtain diagonal dominance in the linear system, the diffusion term in unstructured formulations is decomposed into primary and secondary terms, with the primary term being implicitly included in the coefficient matrix. First-order accurate convective operators are easily incorporated, but higher-order convective operators are more challenging to formulate, and remain an open research area; this aspect is discussed further in Section 1.5. The computation of gradients, again, is substantially more complicated than for structured meshes. The nominally linear algebraic equation set resulting from the discretization is sparse but not banded. Solution techniques to address this type of problem are discussed in Section 1.6.

1.5 HIGHER-ORDER SCHEMES FOR CONVECTION OPERATORS

Over the last two decades, a great deal of effort has been devoted to improving the accuracy of convective operators for both structured and unstructured meshes. The first-order upwind and second-order central-difference schemes described in Section 1.4.1 are usually not suitable for practical use on moderate-sized meshes. Consider convection of a scalar ϕ over the square domain shown in Fig. 1.11. The left and bottom boundaries are held at $\phi = 0$ and $\phi = 1$, respectively. The flow field in the domain is given by $\mathbf{V} = 1.0\mathbf{i} + 1.0\mathbf{j}$, so that the velocity vector is aligned with the diagonal, as shown. The objective is to compute the distribution of ϕ in the domain using the upwind and central-difference schemes for the case when the flow Peclet number $Pe = \rho|\mathbf{V}|L/\Gamma \rightarrow \infty$. For this case, the solution is $\phi = 1$ below the diagonal and $\phi = 0$

FIGURE 1.11 Schematic of scalar transport in a square domain and computed variation of ϕ along the vertical centerline.

above the diagonal. Figure 1.11 shows the predicted ϕ values along the vertical center-line of the domain ($x = 0.5$) using 13×16 quadrilateral cells. The first-order upwind scheme smears the ϕ profile so that there is a diffusion layer even when there is no physical diffusion. The central-difference scheme, on the other hand, shows unphysical oscillations in the value of ϕ . Though this example employs a Peclet number of infinity, similar problems manifest themselves for high Reynolds number flows on moderate-sized meshes in many practical simulations.

Over the last two decades, a number of improvements to the discretization of the convective operator have been made. Two broad approaches have been taken. One approach has been to develop schemes that are substantially more accurate than the first- and second-order schemes described thus far. For applications such as direct numerical simulation (DNS) of turbulence and for computational aero-acoustics (CAA), ultra-high-accuracy methods using compact finite differences and spectral/spectral-element schemes have been developed [23–25]. The second approach has addressed more conventional applications. Here the focus has been on constructing higher-order upwind-weighted schemes by truncating Taylor series expansions to second order or higher. Methods have been developed to control spatial oscillations in these schemes while retaining formal higher-order accuracy. This latter class of schemes is now described.

1.5.1 Upwind-weighted Higher-order Schemes

The upwind scheme may be interpreted as a truncation to $O(\Delta x)$ of a Taylor series expansion for ϕ . If face e in Fig. 1.7 is considered for the case $F_e > 0$, such an expansion in the neighborhood of the *upwind* point P may be written, assuming a uniform mesh of size Δx , as

$$\phi(x) = \phi_P + (x - x_P) \frac{\partial \phi}{\partial x} + \frac{(x - x_P)^2}{2!} \frac{\partial^2 \phi}{\partial x^2} + O(\Delta x)^3 \quad (1.67)$$

By retaining more terms in the Taylor series, a family of upwind-weighted higher-order schemes may be developed.

Second-order Upwind Schemes A second-order upwind scheme may be derived by retaining the first two terms of the expansion in Eq. (1.67). Evaluating Eq. (1.67) at $x_e = x_P + (\Delta x)/2$, we obtain

$$\phi_e = \phi_P + \frac{\Delta x}{2} \frac{\partial \phi}{\partial x} \quad (1.68)$$

This approximation has a truncation error of $O(\Delta x)^2$. To write ϕ_e in terms of cell centroid values, $\partial \phi / \partial x$ must be written in terms of cell centroid values. On a one-dimensional grid, the derivative at P may be written using either a forward, backward, or central difference formula to give three alternative second-order schemes. For example, if $\partial \phi / \partial x$ is written using

$$\frac{\partial \phi}{\partial x} = \frac{\phi_P - \phi_W}{\Delta x} \quad (1.69)$$

we obtain

$$\phi_e = \phi_P + \frac{\phi_P - \phi_W}{2} \quad (1.70)$$

This is the basis of the Beam-Warming scheme [26].

Third-order Upwind Schemes Third-order accurate schemes may be derived by retaining the second derivative in the Taylor series expansion as

$$\phi(x) = \phi_P + (x - x_P) \frac{\partial \phi}{\partial x} + \frac{(x - x_P)^2}{2!} \frac{\partial^2 \phi}{\partial x^2} \quad (1.71)$$

Using cell-centroid values to write the derivatives $\partial \phi / \partial x$ and $\partial^2 \phi / \partial x^2$ we obtain

$$\frac{\partial \phi}{\partial x} = \frac{\phi_E - \phi_W}{2\Delta x} + O(\Delta x^2) \quad (1.72)$$

and

$$\frac{\partial^2 \phi}{\partial x^2} = \frac{\phi_E + \phi_W - 2\phi_P}{(\Delta x)^2} + O(\Delta x^2) \quad (1.73)$$

Inserting Eqs. (1.72) and (1.73) into Eq. (1.71) and rearranging yields

$$\phi_e = \frac{\phi_E + \phi_P}{2} - \frac{\phi_E + \phi_W - 2\phi_P}{8} \quad (1.74)$$

This scheme is called the QUICK scheme (quadratic upwind interpolation for convective kinetics) [27]. These schemes are not truly multidimensional in that upwinding occurs along grid lines. Also, line structure is required in these schemes, making them unsuitable for use on unstructured meshes.

Extension to Unstructured Meshes Formulation of higher-order schemes for unstructured meshes is an area of active research and new ideas continue to emerge. A second-order accurate unstructured mesh scheme based on the ideas in the previous section is now presented. The starting point is the multidimensional equivalent of Eq. (1.68). Referring to Fig. 1.9, if $F_f > 0$, ϕ may be written using a Taylor series expansion about the *upwind* cell centroid as

$$\phi(x, y) = \phi_0 + (\nabla\phi)_0 \cdot \Delta\mathbf{r} + O(|\Delta\mathbf{r}|^2) \quad (1.75)$$

where $\Delta\mathbf{r}$ is

$$\Delta\mathbf{r} = (x - x_0)\mathbf{i} + (y - y_0)\mathbf{j} \quad (1.76)$$

To find the face value ϕ_f , Eq. (1.75) is evaluated at $\Delta\mathbf{r} = \Delta\mathbf{r}_0$, as shown in Fig. 1.9, to give

$$\phi_f = \phi_0 + (\nabla\phi)_0 \cdot \Delta\mathbf{r}_0 + O(|\Delta\mathbf{r}_0|^2) \quad (1.77)$$

As with structured meshes, $(\nabla\phi)_0$ must be evaluated. This can be done using either of the techniques described in Section 1.4.2.

1.5.2 Control of Spatial Oscillations

The schemes described in the previous section give higher-order accuracy but can still produce spatial oscillations in steady problems. If used in conjunction with the Euler explicit scheme [26] for time-stepping in unsteady problems, these schemes are unconditionally unstable. A number of research efforts have tried to remedy these problems, two of which are described below.

Added Dissipation Schemes One technique to eliminate spatial oscillations is to use one of the higher-order schemes developed in the previous sections, but to damp out the oscillations through the explicit use of an *artificial viscosity* tailored to maintain the desired formal accuracy of the scheme [28]. In the case of the central-difference scheme, a dissipation term involving a discrete fourth derivative is used. Thus, referring to Fig. 1.7, ϕ_e can be expressed as

$$\phi_e = \frac{\phi_P + \phi_E}{2} + \varepsilon_e^{(4)} (\phi_{EE} - 3\phi_E + 3\phi_P - \phi_W) \quad (1.78)$$

This amounts to adding a term of the type $\Delta x^3 (\partial^4 \phi / \partial x^4)$ to the governing equation. Since the additional term is $O(\Delta x^3)$, it does not change the formal second-order accuracy of the central-difference scheme. Near discontinuities in ϕ , it is necessary to add a stronger dissipation and a second-order term is also introduced, which reduces the formal accuracy of the scheme to first order [28]. The resulting expression for ϕ_e is

$$\phi_e = \frac{\phi_P + \phi_E}{2} - \varepsilon_e^{(2)} (\phi_E - \phi_P) + \varepsilon_e^{(4)} (\phi_{EE} - 3\phi_E + 3\phi_P - \phi_W) \quad (1.79)$$

To use this type of idea successfully, it is necessary to choose the coefficients $\varepsilon_e^{(2)}$ and $\varepsilon_e^{(4)}$, and also to detect discontinuities and shocks, so that $\varepsilon_e^{(2)}$ can be made small in the bulk of the flow.

Flux Limiters The use of Eq. (1.68) does not guarantee that ϕ_e is bounded by ϕ_P and ϕ_E , or by any other stencil in the neighborhood of face e , leading to spatial oscillations in the computed values of ϕ . Schemes employing flux limiters seek to overcome this problem by *limiting* the contribution of the gradient term using

$$\phi_e = \phi_P + \Psi_e(r_e) \frac{\Delta x}{2} \frac{\partial \phi}{\partial x} \Big|_P \quad (1.80)$$

Here Ψ is limiter function chosen to assure the boundedness of ϕ . The gradient $(\partial \phi / \partial x)|_P$ depends on the scheme being implemented. For the Beam-Warming scheme, for example,

$$\phi_e = \phi_P + \Psi_e(r_e) \frac{\Delta x}{2} \frac{\phi_P - \phi_W}{\Delta x} \quad (1.81)$$

The limiter Ψ is a function of the variable r_e , which is itself a function of differences of ϕ :

$$r_e = \frac{\phi_E - \phi_P}{\phi_P - \phi_W} \quad (1.82)$$

A variety of limiter functions have been used in the literature, including the minmod, superbee, van Leer, and van Albada limiters [29, 30]. The corresponding functional variation is shown in Fig. 1.12. The advantage of using a limiter becomes readily apparent when considering the problem of linear advection of a square wave form with a uniform velocity u . Since there is no diffusion, the numerical scheme must preserve the shape of the wave form during its translation. Figure 1.13 shows the prediction of the Beam-Warming scheme with and without limiters. In the absence of limiters, oscillations in the shape of the wave begin to develop and grow with time, and are particularly evident at corners. These oscillations disappear when limiters are used. For a more detailed discussion of these methods, see [29, 30].

FIGURE 1.12 Limiter functions.

FIGURE 1.13 Linear advection of a square wave using Beam-Warming scheme (a) without limiter, (b) with minmod limiter, and (c) with superbee limiter.

FIGURE 1.13 (continued).

1.5.3 Summary and Discussion

In this section, we have reviewed widely used higher-order schemes for the convection operator. Schemes based on higher-order truncations of the Taylor series do yield more accurate schemes, but require special manipulation using either artificial dissipation or limiting to control spatial oscillations. As is clear from the development, the schemes described in this section require line structure. The extension to unstructured meshes remains an area of active research.

1.6 LINEAR SOLVERS

Attention is now turned to another important aspect of the numerical process, namely the solution of linear algebraic equation sets. As discussed in Section 1.3.3, regardless of what discretization process is used, the result is a coupled algebraic set of equations in the discrete values of ϕ . The resulting coefficient matrices have two important characteristics. First, they are sparse, and in the case of structured meshes, they are banded. Second, the coefficient matrices are usually approximate; for nonlinear problems, for example, the coefficient matrix is updated repeatedly as a part of outer iteration loop to resolve nonlinearities. Over the last three decades, iterative methods have emerged as the preferred approach in CFD. They are naturally suited for handling nonlinearities since the coefficient matrix can be updated during the iterative process. In addition, operation counts as well as storage typically scale as $O(N)$, where N is the number of unknowns. The specific solution techniques depend on whether the underlying mesh is structured or not. Special algorithms taking advantage of band structure are used for structured meshes. For unstructured meshes, matrix sparseness is exploited. Of course, linear solvers for unstructured meshes can be used for structured meshes as well. Typical solution techniques will now be considered.

1.6.1 Line Gauss-Seidel Method

The line Gauss-Seidel technique (LGS) is widely used with structured meshes. The central component of LGS is a direct solver for tridiagonal systems called the tridiagonal matrix algorithm (TDMA), which is applied iteratively along lines in the structured mesh. The procedure is also sometimes called the line-by-line TDMA. The TDMA is essentially a Gaussian-elimination procedure which takes advantage of the tridiagonal structure of the matrix.

Tridiagonal Matrix Algorithm Consider the equation system

$$a_i \phi_i = b_i \phi_{i+1} + c_i \phi_{i-1} + d_i \quad (1.83)$$

This type of equation results from the discretization of a 1D convection–diffusion equation using the techniques described previously. An equation of this type maybe written for each grid point i . For the first grid point, $i = 1$, $c_i = 0$, and for the last grid point, $i = N$, b_i is zero. Thus, for point $i = 1$

$$\phi_1 = P_1 \phi_2 + Q_1 \quad (1.84)$$

Equation (1.84) may now be used to eliminate ϕ_1 in favor of ϕ_2 in the equation for $i = 2$, resulting in

$$\phi_2 = P_2 \phi_3 + Q_2 \quad (1.85)$$

In general,

$$\phi_i = P_i \phi_{i+1} + Q_i \quad (1.86)$$

Here,

$$\begin{aligned} P_i &= \frac{b_i}{a_i - c_i P_{i-1}} \\ Q_i &= \frac{d_i + c_i Q_{i-1}}{a_i - c_i P_{i-1}} \end{aligned} \quad (1.87)$$

It should be noted that $P_1 = b_1/a_1$, $Q_1 = d_1/a_1$, and $P_N = 0$. The equation for the last point, $i = N$, yields

$$\phi_N = Q_N \quad (1.88)$$

The implementation of the algorithm is done in two parts. In the *forward* step, the coefficients P_i and Q_i , $i = 1, 2, \dots, N$ are found using Eq. (1.87) recursively, and ϕ_N is calculated. In the *backward* sweep, Eq. (1.86) is used recursively for $i = N - 1, N - 2, \dots, 1$ to recover ϕ_i .

Line-by-line Algorithm For two- and three-dimensional structured meshes, the equation system is banded, but is not tridiagonal. In these cases, the TDMA is applied iteratively along

FIGURE 1.14 Two-dimensional Cartesian mesh.

lines. For two-dimensional structured meshes (see Fig. 1.14), the discrete equation for a point P may be written as

$$a_P \phi_P = a_E \phi_E + a_W \phi_W + a_N \phi_N + a_S \phi_S + b \quad (1.89)$$

Here each grid point P is connected to its four neighbor points E, W, N, S . A tridiagonal system may be created along each line by assuming values on the neighbor lines to be temporarily known so that

$$a_P \phi_P = a_N \phi_N + a_S \phi_S + b^* \quad (1.90)$$

where $b^* = (b + a_E \phi_E^* + a_W \phi_W^*)$, and the starred values are prevailing values of ϕ .

The procedure starts with a guess of all grid point values of ϕ . Starting with a vertical grid line $I = 1$, Eq. (1.90) is solved, with b^* being evaluated from the current guess of ϕ . The TDMA is used along $I = 1$ to obtain ϕ values along the line. These are, of course, provisional since b^* is based on guessed or prevailing values. The calculation now shifts to line $I = 2$, and the procedure is repeated; the most recently computed values on $I = 1$ are used to construct b^* . All I lines are visited in this fashion. The same procedure is then applied in the J direction. Several such iterations may be done to obtain a converged solution for ϕ . For three dimensions, grid planes are visited sequentially and iteratively, applying the LGS on each plane until overall convergence is obtained. Other iterative techniques for structured meshes include the alternating direction implicit (ADI) technique, which uses the TDMA in conjunction with a time-stepping scheme [31], incomplete lower-upper (ILU) decomposition [31], and the strongly implicit procedure (SIP) [32].

1.6.2 Multigrid Methods

The LGS technique cannot be used for unstructured meshes since there are no easily identifiable lines in the domain. It may be recalled from Section 1.3.3 that the Gauss-Seidel technique does

not require line structure, and can be applied to sparse matrices with diagonal dominance, making it ideal for solving the sparse systems resulting from unstructured discretizations. However, the rate of convergence is too slow for practical use. Multigrid techniques may be used to accelerate Gauss-Seidel iteration, although techniques other than Gauss-Seidel may be used as the core solver in multigrid techniques as well.

Convergence Behavior of Jacobi and Gauss-Seidel Techniques Although the Jacobi and Gauss-Seidel methods are easy to implement and are applicable for matrices with arbitrary fill patterns, their usefulness is limited by their slow convergence characteristics. The usual observation is that residuals drop quickly during the first few iterations but afterward the iterations “stall.” This behavior is especially pronounced for large matrices.

To demonstrate this behavior, a one-dimensional 1D Laplace equation over a domain of length L is considered, that is,

$$\frac{\partial^2 \phi}{\partial x^2} = 0 \quad (1.91)$$

Dirichlet boundary conditions are applied, so that $\phi(0) = \phi(L) = 0$. The exact solution to this problem is simply $\phi(x) = 0$. The behavior of iterative schemes may be studied by starting with an arbitrary initial guess. The error at any iteration is then simply the current value of the variable ϕ . To distinguish the convergence characteristics for different error profiles, the current problem is solved with an initial guess given by

$$\phi_i = \sin\left(\frac{k\pi x_i}{L}\right) \quad (1.92)$$

Equation (1.92) represents *Fourier* modes and k is the wave number. Equation (1.91) is discretized using the techniques described previously. Starting with Eq. (1.92) as the initial guess, the Gauss-Seidel method is applied for 50 iterations on a grid with $N = 64$. The maximum error in the solution is shown in Fig. 1.15a. With an initial guess corresponding to $k = 1$, the maximum error has reduced by less than 20% after 50 iterations. On the other hand, with a guess of $k = 16$, the error reduces by over 99% after merely 10 iterations. An arbitrary initial guess would contain more than one Fourier mode. To see what the scheme does in such cases, an initial guess consisting of modes corresponding to $k = 2, 8$, and 16 is used. For this situation

$$\phi_i = \frac{1}{3} \left[\sin\left(\frac{2\pi x_i}{L}\right) + \sin\left(\frac{8\pi x_i}{L}\right) + \sin\left(\frac{16\pi x_i}{L}\right) \right] \quad (1.93)$$

From Fig. 1.15b it can be seen that the error drops rapidly at first but then decreases much more slowly. The Gauss-Seidel scheme is very effective at reducing high wave-number errors. This accounts for the rapid drop in residuals at the beginning. Once the high-wave-number components are removed, only the smooth error profiles remain for which the scheme is not very effective and thus convergence stalls.

Using this sample problem, another commonly encountered shortcoming of the Gauss-Seidel iterative scheme can be observed. It is found that convergence deteriorates as the grid is refined. Retaining the same form of initial guess and using $k = 2$, the previous problem is solved on three different grids, $N = 32, 64$, and 128. The resulting convergence plot, shown in Fig. 1.16, indicates that the rate of convergence becomes worse as the mesh is refined. On a finer grid,

FIGURE 1.15 Convergence of Gauss-Seidel method on $N = 64$ grid for (a) initial guesses consisting of single wave numbers and (b) initial guess consisting of multiple modes.

it is possible to resolve more modes. The higher modes converge quickly but the lower modes appear more “smooth” and hence converge more slowly. The initial error profile behaves like a high-wave-number profile on a coarser grid, but like a low-wave-number profile in a finer grid. A quantitative analysis of these behaviors may be found in [33].

The multigrid method seeks to accelerate the convergence rate of iterative linear solvers by involving coarser grids. It is necessary that the accuracy of the final solution be determined only by the finest grid that is employed. This means that the coarse grids can provide only corrections or guesses to the fine-grid solution. As the fine-grid solution approaches the exact answer, the influence of any coarse levels should approach zero. Thus, it is enough to solve

FIGURE 1.16 Convergence of Gauss-Seidel method on different-sized grids for initial guess corresponding to $k = 2$ mode.

only an approximate problem at the coarse levels since their solution will not govern the final accuracy that is achieved.

Two different multigrid approaches are available in the literature. The first is the *geometric* multigrid or full approximation storage (FAS) procedure ([34, 35], for example). Here, a sequence is created of coarse multigrid meshes that are not necessarily nested. (A nested multigrid mesh is one in which each face of the coarse mesh is composed of the faces of original fine mesh.) The governing equations are discretized on each coarse level independently, and the solution errors at each level are used to accelerate the solution on finer levels. An alternative is the *algebraic* multigrid method [36, 37], which is now described.

Algebraic Multigrid Method The algebraic multigrid (AMG) method is well-suited for unstructured meshes since it does not involve discretization of the governing equations on coarser grids. Instead, a hierarchy of coarse equation sets is constructed by grouping a number of fine-level discrete equations. Residuals from a fine-level relaxation sweep are “restricted” to form the source terms for the coarser-level correction equations. The solution from the coarser equations is in turn “prolongated” to provide corrections at the finer level. The use of different grid sizes permits the reduction of errors at all wavelengths using relatively simple smoothing operators.

It is useful to represent the discrete equation at point i at a grid level l as

$$\sum_j M_{ij}^l \phi_j + S_i = 0 \quad (1.94)$$

where j is the index of a neighbor cell.

The algebraic multigrid method visits each ungrouped fine-level cell and groups it with n of its neighboring ungrouped cells for which the coefficient M_{ij} is the largest [37]. The AMG

performs best when the group size, n , is 2. The coefficients for the coarse-level equations are obtained by summing the coefficients of the fine-level equations:

$$M_{IJ}^{l+1} = \sum_{i \in G_I} \sum_{j \in G_J} M_{ij}^l \quad (1.95)$$

where the superscripts denote the grid level and G_I is the set of fine-level cells that belong to the coarse-level group I . This results in a system of equations of the same form as the fine level (i.e., Eq. (1.52)), with $1/n$ th the number of unknowns

$$M_{IJ}^{l+1} \phi_J^{l+1} - \sum_{i \in G_I} R_i^l = 0 \quad (1.96)$$

where R_i^l is the residual in the fine-level equation at the current iteration

$$R_i^l = M_{ij}^l \phi_j^{*l} + S_i \quad (1.97)$$

The value ϕ_j^{*l} is the current iterate. The process is repeated recursively until no further coarsening is possible. A variety of strategies, such as the V, W, and Brandt cycles [38] may be used to cycle between the grid levels. The solution at any level is obtained by a Gauss-Seidel iterative scheme and is used to correct the current iterate at the next finer level. Thus, for all $i \in G_I$:

$$\phi_i^l = \phi_i^{*l} + \phi_I^{l+1} \quad (1.98)$$

Intelligent mesh agglomeration strategies for creating coarse-level meshes are critical for obtaining significant convergence acceleration using multigrid schemes. Lonsdale [37] employed an agglomeration strategy that grouped together cells connected by the largest coefficients. This strategy has proven effective for a variety of problems involving high thermal conductivity ratios, large domain aspect ratios, and disparate grid sizes. For a more in-depth discussion of mesh agglomeration strategies, see [39].

Algebraic multigrid methods used with sequential solution procedures have the advantage that the agglomeration strategy can be equation-specific; the discrete coefficients for the specific governing equation can be used to create coarse mesh levels. Since the coarsening is based on the coefficients of the linearized equations, it also changes appropriately as the solution evolves. This is especially useful for nonlinear and/or transient problems. In some heat transfer applications, however, the mutual coupling between the governing equations is the main cause of convergence degradation, and sequential solution procedures do not perform well. Typical examples include flows with large body forces such as high-Rayleigh number buoyant flows, or flows with large swirl numbers. Geometric or full-approximation storage multigrid methods that solve the coupled problem on a sequence of coarser meshes may offer better performance in such cases.

1.6.3 Gradient-search Techniques

Gradient-search techniques have recently found increased use in CFD because of their ability to solve the equation sets resulting from unstructured discretizations. For symmetric positive-definite matrices, the original problem $\mathbf{A}\phi = \mathbf{B}$ can be shown to be equivalent to the minimization of a functional F defined as

$$F = \frac{1}{2} \phi^T \mathbf{A} \phi - \phi^T \mathbf{B} \quad (1.99)$$

The method of steepest descent essentially finds the minimum of F by using search directions opposite to ∇F . This search process is usually too slow for practical use. In contrast, conjugate gradient methods [40] employ search directions that are conjugate to all previous search directions; preconditioning may be used to improve the speed of conjugate gradient techniques. Few of the linear systems resulting from CFD problems are either symmetric or positive-definite. Extensions of the method to address asymmetric matrices include biconjugate gradients [31], CGSTAB and BI-CGSTAB [41, 42], and GMRES [43].

1.7 COMPUTATION OF FLUID FLOW

The class of problems considered thus far involve convection and diffusion of a scalar in the presence of a *known* flow field. Even though the continuity and momentum equations have the same form as the general scalar transport equation, Eq. (1.6), a number of additional factors must be considered in the computation of the flow field. In three dimensions, the unknowns to be computed are the three velocity components and the pressure. The equations available for their computation are the three momentum equations and the continuity equation. A number of issues arise in the storage and computation of pressure and velocity, which are now discussed.

1.7.1 Storage of Pressure and Velocity

For simplicity, we consider the uniform structured two-dimensional mesh shown in Fig. 1.14. The pressure p and the velocity vector $\mathbf{V}$ are assumed to be stored at the cell centroid. Following the practices outlined in previous sections, the discrete u - and v -momentum equations may be written as

$$\begin{aligned} a_P u_P &= \sum_{nb} a_{nb} u_{nb} + (p_w - p_e) \Delta y + b_u \\ a_P v_P &= \sum_{nb} a_{nb} v_{nb} + (p_s - p_n) \Delta x + b_v \end{aligned} \quad (1.100)$$

The summation over nb denotes a summation over the neighbors E , W , N , and S in Fig. 1.14. Here, the pressure gradient is written in terms of the values of pressure on the control volume faces. Since the pressure is stored at the cell centroids and not at the face, interpolation is necessary. For a uniform grid, p_e may be found by linear interpolation between cell centroids from

$$p_e = \frac{p_E + p_P}{2} \quad (1.101)$$

The other face pressures may be similarly interpolated. Incorporating this assumption into the discrete momentum equations yields

$$\begin{aligned} a_P u_P &= \sum_{nb} a_{nb} u_{nb} + (p_W - p_E) \frac{\Delta y}{2} + b_u \\ a_P v_P &= \sum_{nb} a_{nb} v_{nb} + (p_S - p_N) \frac{\Delta x}{2} + b_v \end{aligned} \quad (1.102)$$

The pressure terms occurring in the momentum equations are seen to involve *alternate* pressures rather than *adjacent* pressures; the value p_P does not appear in the equations at all.

Next, attention is turned to the continuity equation. Discretizing the continuity equation, we obtain

$$(\rho u)_e \Delta y - (\rho u)_w \Delta y + (\rho v)_n \Delta x - (\rho v)_s \Delta x = 0 \quad (1.103)$$

The face velocities are not available directly but must be interpolated from the cell centroid values to the face. For a uniform grid, for example, $(\rho u)_e$ may be found by linear interpolation as

$$(\rho u)_e = \frac{(\rho u)_P + (\rho u)_E}{2} \quad (1.104)$$

The other terms in Eq. (1.103) may similarly be interpolated. Gathering terms, the discrete continuity equation for the cell P is

$$(\rho u)_E \Delta y - (\rho u)_W \Delta y + (\rho v)_N \Delta x - (\rho v)_S \Delta x = 0 \quad (1.105)$$

An examination of the discrete continuity equation for cell P reveals that it does not contain the velocity for cell P . Consequently, a checkerboarded velocity pattern of the type shown in Fig. 1.17 can be sustained by the continuity equation. If the momentum equations can sustain this pattern, the checkerboarding would persist in the final solution. Since the pressure gradient is not known a priori, but is computed as a part of the solution, it is possible to create pressure fields whose gradients exactly compensate the checkerboarding of momentum transport implied by the checkerboarded velocity field. Under these circumstances, the final pressure and velocity fields would exhibit checkerboarding, even though the discrete momentum and continuity equations are perfectly satisfied.

In practice, perfect checkerboarding is rarely encountered because of irregularities in the mesh, boundary conditions, and physical properties. Instead, the tendency toward checkerboarding manifests itself in unphysical wiggles in the velocity and pressure fields. It should be emphasized that these wiggles are a property of the spatial discretization and would be obtained regardless of the method used to solve the discrete equations.

A number of different remedies have emerged to address the checkerboarding problem, some of which are described below.

Staggered Storage of Pressure and Velocity A popular remedy for checkerboarding on structured meshes, either regular or body-fitted, is the use of a staggered mesh [1]. A typical staggered mesh arrangement is shown in Fig. 1.18. We distinguish between the *main* cell or control volume and the *staggered* cell or control volume. The pressure is stored at centroids of the *main* cells. The velocity components are stored on the faces of the main cells as shown,

100 ○	200 ○	100 ○	200 ○
----------	----------	----------	----------

FIGURE 1.17 Checkerboarded velocity field.

FIGURE 1.18 Staggered mesh.

and are associated with the *staggered* cells. The u velocity is stored on the e and w faces and the v velocity is stored on the n and s faces. Scalars such as enthalpy or species mass fraction are stored at the centroids of the cell P . All properties, such as density and Γ , are stored at the main grid points.

The cell P is used to discretize the continuity equation as

$$(\rho u)_e \Delta y - (\rho u)_w \Delta y + (\rho v)_n \Delta x - (\rho v)_s \Delta x = 0 \quad (1.106)$$

However, no further interpolation of velocity is necessary since discrete velocities are available directly where required. Thus, the possibility of velocity checkerboarding is eliminated.

For the momentum equations, the staggered control volumes are used to write momentum balances. The procedure is the same as that described previously, except that the pressure gradient term may be written directly in terms of the pressures on the faces of the momentum control volumes, without interpolating as in Eq. (1.101). Thus, for the discrete momentum equation for the velocity u_e , the pressure term is

$$(p_P - p_E) \Delta y \quad (1.107)$$

Similarly, for the velocity v_n , the pressure term is

$$(p_P - p_N) \Delta x \quad (1.108)$$

Thus, with the use of Eqs. (1.107) and (1.108), there is no longer a dependence on alternate pressure values; adjacent pressure values appear in the balance and do not support pressure checkerboarding. It may be noted that the mesh for the u -momentum equation consists of nonoverlapping cells that fill the domain completely. This is also true for the v -momentum equation and the continuity equation. The control volumes for u , v , and p overlap each other, but this is of no consequence. Furthermore, since the velocities are available on the main cell

faces, face flow rates can easily be computed where they are needed for the discretization of the convective terms in the scalar transport equation.

For body-fitted meshes, components of either the covariant or the contravariant velocity vector are stored on the faces [5, 15]. In all other respects, the basic idea is the same as that described here.

Unequal Order Schemes For unstructured meshes in either the finite-volume or the finite-element context, pressure-velocity staggering is difficult to implement because of geometric complexity. As a result, control-volume finite-element methods (CVFEM) as well as conventional finite-element methods have used unequal-order interpolation of pressure and velocity [7, 44, 45]. Here, pressure is effectively interpolated to lower order than velocity. In CVFEM, this is accomplished by resolving the pressure on a *macroelement*, whereas the velocity is resolved on a *subelement*, which is formed by dividing the macroelement into smaller elements. Alternatively, a lower-order interpolation function may be used for pressure vis-à-vis velocity [44, 45].

Colocated Schemes Both mesh staggering and unequal-order interpolation require book-keeping and storage of extra geometric information. As a result, research has been directed to the development of *colocated* or *equal-order* interpolation schemes. Here, pressure and Cartesian velocity components are both stored at the cell centroid. However, the interpolation of the face velocity from cell-centered velocities is modified so as to remove checkerboarded pressure modes [17, 46, 47]. The modified interpolation is equivalent to an added dissipation that damps spatial wiggles in pressure and velocity; consequently, these schemes are sometimes referred to as *added-dissipation* schemes. Formulations for regular, body-fitted, and unstructured meshes have appeared in the literature [6, 17, 46]. A formulation for an equal-order CVFEM has been published in [47]. In the finite-element context, formulations interpolating velocity and pressure to equal order have been published [18, 48].

In the discussion that follows, an orthogonal, one-dimensional, uniform mesh is used for clarity. The mesh and associated nomenclature are shown in Fig. 1.7. Adopting a linear interpolation of pressure between cell centroids, the discrete u -momentum equations for cells P and E may be written as

$$\begin{aligned} a_P u_P &= \sum_{nb} a_{nb} u_{nb} + b_P'' + \frac{p_W - p_E}{2} \\ a_E u_E &= \sum_{nb} a_{nb} u_{nb} + b_E'' + \frac{p_P - p_{EE}}{2} \end{aligned} \quad (1.109)$$

For convenience, Eqs. (1.109) are recast as

$$\begin{aligned} u_P &= \hat{u}_P + d_P \frac{p_W - p_E}{2} \\ u_E &= \hat{u}_E + d_E \frac{p_P - p_{EE}}{2} \end{aligned} \quad (1.110)$$

where $d_P = 1/a_P$ and $d_E = 1/a_E$, and

$$\begin{aligned} \hat{u}_P &= \frac{\sum_{nb} a_{nb} u_{nb} + b_P''}{a_P} \\ \hat{u}_E &= \frac{\sum_{nb} a_{nb} u_{nb} + b_E''}{a_E} \end{aligned} \quad (1.111)$$

The continuity equation for point P is discretized to yield

$$\rho_e u_e - \rho_w u_w = 0 \quad (1.112)$$

Collocated formulations prevent checkerboarding by devising interpolation procedures that express the face velocities u_e and u_w in terms of *adjacent* pressure values rather than alternate pressure values. The face velocity u_e is not defined purely as a linear interpolant of the two adjacent cell values; an additional term, called the *added dissipation* prevents velocity checkerboarding. If u_e is linearly interpolated

$$u_e = \frac{u_P + u_E}{2} = \frac{\hat{u}_P + \hat{u}_E}{2} + d_P \left(\frac{p_W - p_E}{4} \right) + d_E \left(\frac{p_P - p_{EE}}{4} \right) \quad (1.113)$$

Instead, the following interpolation is used:

$$\begin{aligned} u_e &= \frac{u_P + u_E}{2} - d_P \left(\frac{p_W - p_E}{4} \right) - d_E \left(\frac{p_P - p_{EE}}{4} \right) + d_e (p_P - p_E) \\ &= \frac{\hat{u}_P + \hat{u}_E}{2} + d_e (p_P - p_E) \end{aligned} \quad (1.114)$$

Here $d_e = (d_P + d_E)/2$. A similar expression may be written for u_w .

In obtaining Eq. (1.114), the pressure difference term resulting from the linear interpolation of velocities (which involves the pressures p_W , p_E , p_P , and p_{EE}) was removed. In its place, a new pressure difference term $(p_P - p_E)$ was added instead. Another way of looking at this is to say that in writing u_e , the $\hat{u}$ component is linearly interpolated between P and E , but the pressure difference term is written directly in terms of the *adjacent* cell-centroid pressures p_P and p_E .

This type of interpolation is sometimes referred to as *momentum interpolation* in the literature. It was proposed, with small variations, by different researchers in the early 1980s [16, 17]. Similar concepts have also been used in the compressible flow community with density-based solvers. It is possible to show that the interpolation procedure is equivalent to adding a dissipation proportional to $(\partial^4 p / \partial x^4) \Delta x^3$ to the continuity equation. Consequently, it is also sometimes referred to as an *added dissipation* scheme. Since the added dissipation is a third-order term, its addition preserves the formal accuracy of second-order schemes. Another useful way to think about momentum interpolation is to consider the face velocity u_e to be a sort of staggered velocity, which is obtained by interpolation. By not using an actual staggered cell, momentum interpolation avoids the creation of staggered cell geometry and makes it possible to use the idea for unstructured meshes.

An important point to be noted is that the discrete continuity equation is written in terms of the face velocities u_e and u_w . It is not written directly in terms of the cell-centered velocities u_P and u_E . Thus, at convergence, it is the *face* velocities that directly satisfy the discrete continuity equation, not the cell-centered velocities. On the other hand, the cell-centered velocities directly satisfy the discrete momentum equations. They satisfy the continuity equations only indirectly through their role in defining u_e and u_w . This curious distinction between cell-centered and face velocities is an inherent property of collocated schemes. Another important issue with momentum interpolation is that the final solution obtained is dependent on the underrelaxation factor, and for unsteady problems, it is time-step dependent. A strategy for avoiding this dependence may be found in [46].

1.7.2 Solution Methods

Thus far, issues related to *discretization* of the continuity and momentum equations have been examined. Attention is now turned to the solution of these equations.

One alternative is to employ a *direct* solution technique. The discrete continuity and momentum equations over the entire domain may be assembled into a large algebraic system of the form

$$\mathbf{M}\phi = \mathbf{b} \quad (1.115)$$

where $\mathbf{M}$ is a matrix of size $N \times N \times 4$, and N is the number of grid points. For a collocated formulation, the unknowns consist of the three velocity components and pressure at the cell centroids of all N cells. This approach has not thus far been tenable for most practical industrial problems with present-day computational power. However, the emergence of efficient multifrontal solvers [13] have made this approach viable for specialized applications, and the technique may find greater use in the future as computational power increases.

For practical CFD problems, sequential iterative solution procedures are frequently adopted because of low storage requirements and reasonable convergence rate. However there is a difficulty associated with the sequential solution of the continuity and momentum equations for incompressible flows. To solve a set of discrete equations iteratively, it is necessary to associate the discrete set with a particular variable. For example, the discrete energy equation is used to solve for the temperature. Similarly, the discrete u -momentum equation is used to solve for the u velocity. If the continuity equation were to be used to solve for pressure, a problem would arise for incompressible flows because the pressure does not appear in the continuity equation directly. On the other hand, the density does appear in the continuity equation, but for incompressible flows, the density is unrelated to the pressure and cannot be used instead. Thus, if sequential, iterative methods are to be used, it is necessary to find a way to introduce the pressure into the continuity equation. Methods that use pressure as the solution variable are called *pressure-based* methods. They are very popular in the incompressible flow community.

A number of methods in the literature [29, 30] use density as a primary variable rather than pressure. This practice is especially popular in the compressible flow community. For compressible flows, pressure and density are related through an equation of state. It is possible to find the density using the continuity equation, and to deduce the pressure from it using an equation of state. Such methods are called *density-based* methods.

For incompressible flows, a class of methods called the *artificial compressibility* methods have been developed, which seek to ascribe a small (but finite) compressibility to incompressible flows to facilitate numerical solution through density-based methods [49]. Conversely, pressure-based methods have also been developed, which may be used for compressible flows [5, 50, 51].

It is important to realize that the necessity for pressure- and density-based schemes is directly tied to the decision to solve the governing equations sequentially and iteratively. It is this choice that forces us to associate each governing differential equation with a solution variable. Other methods, such the point-coupled technique of Vanka [34], require no such association.

Pressure-based Techniques: SIMPLE Algorithm For simplicity, the implementation of pressure-based techniques is discussed in the context of a staggered storage scheme for pressure and velocity. A detailed description for collocated storage on unstructured meshes may be found in [6]. The starting point is the semi-implicit method for pressure-linked equations (SIMPLE) [52], though a number of variants have been published over the years, such as SIMPLER [1] and SIMPLEC [53].

The primary idea behind SIMPLE is to create a discrete equation for pressure (or alternatively, a related quantity called the pressure correction) from the discrete continuity equation. Since the continuity equation contains discrete face velocities, it is necessary to relate these discrete velocities to the discrete pressure field. The SIMPLE algorithm uses the discrete momentum equations to derive this connection.

Consider the staggered mesh arrangement shown in Fig. 1.18. Let u^* and v^* be the discrete u and v fields resulting from a solution of the discrete momentum equations. Let p^* represent the discrete pressure field, which is used in the solution of the momentum equations. Thus, u_e^* and v_n^* satisfy

$$\begin{aligned} a_e u_e^* &= \sum_{nb} a_{nb}^u u_{nb}^* + \Delta y (p_P^* - p_E^*) + b_e \\ a_n v_n^* &= \sum_{nb} a_{nb}^v v_{nb}^* + \Delta x (p_P^* - p_N^*) + b_n \end{aligned} \quad (1.116)$$

Similar expressions may be written for u_w^* and v_s^* . If the pressure field p^* is only a guess or a prevailing iterate, the discrete u^* and v^* obtained by solving the momentum equations will not, in general, satisfy the discrete continuity equation, Eq. (1.106). A correction to the starred velocity field is proposed such that the corrected values satisfy Eq. (1.106). Correspondingly, a pressure correction is also proposed. Thus,

$$\begin{aligned} u &= u^* + u' \\ v &= v^* + v' \\ p &= p^* + p' \end{aligned} \quad (1.117)$$

An approximate correction equation modeled on Eqs. (1.116) is now proposed:

$$\begin{aligned} a_e u_e' &\approx \Delta y (p_P' - p_E') \\ a_n v_n' &\approx \Delta x (p_P' - p_N') \end{aligned} \quad (1.118)$$

The corrected velocities, given by Eq. (1.117), are required to satisfy the discrete continuity equation, Eq. (1.106). Substituting in the discrete continuity equation, the pressure-correction equation may be derived as

$$a_P p_P' = \sum_{nb} a_{nb} p_{nb}' + b \quad (1.119)$$

where

$$\begin{aligned} a_E &= \frac{\rho_e \Delta y^2}{a_e} \\ a_W &= \frac{\rho_w \Delta y^2}{a_w} \\ a_N &= \frac{\rho_n \Delta x^2}{a_n} \end{aligned}$$

$$\begin{aligned}
a_S &= \frac{\rho_s \Delta x^2}{a_s} \\
a_P &= \sum_{nb} a_{nb} \\
b &= \rho_w u_w^* \Delta y - \rho_e u_e^* \Delta y + \rho_s v_s^* \Delta x - \rho_n v_n^* \Delta x
\end{aligned} \tag{1.120}$$

The source term in the pressure-correction equation is the *mass imbalance* for the cell P . If the face flow rates satisfy the discrete continuity equation (i.e., b is zero), $p' = \text{constant}$ satisfies Eq. (1.120), and differences of p' are zero. Thus, the velocity field is corrected only as long as the velocity fields produced by the momentum equations do not satisfy continuity.

The overall procedure for the SIMPLE algorithm employs a sequential solution of the momentum- and pressure-correction equations, correcting the velocities produced by the momentum equations to satisfy the discrete continuity equation. A detailed description of the procedure may be found in [1]. Even though pressure-based schemes were originally intended to address incompressible flows, pressure as a variable is well-suited for the solution of compressible flows as well. Segregated pressure-based algorithms for all-speed flows have been published in [5, 50, 51].

Coupled Techniques Though segregated techniques have had a great deal of success in solving industrial problems, they can be slow to converge and may even diverge when interequation coupling is strong. A number of researchers have developed more strongly coupled approaches [34, 35, 54–56]. Among the earliest of these attempts is [34]. Here, a symmetric coupled Gauss-Seidel iteration forms the relaxation sweep of a FAS multigrid scheme. A staggered grid approach is used. The Gauss-Seidel sweep visits each main control volume, solving for the four face velocities (in 2D) and the cell-centered pressure simultaneously, assuming neighbor values to be temporarily known. Coarse-level meshes are created by grouping fine-level cells together while retaining a structured Cartesian mesh. In an alternative approach, one of the SIMPLE family of algorithms is used as the relaxation sweep in a FAS multigrid scheme in [35, 55]. Nonorthogonal body-fitted meshes in a colocated formulation are addressed in [55], whereas unstructured meshes are used in [35]; here, the coarsest mesh is generated first, and progressively fine meshes are created by subdivision. An algebraic coupled multigrid scheme is presented in [54]. Here, only the finest mesh is created, and coarse-level equations are found from the algebraic manipulation of fine-level discrete equations. Substantial increases in the rate of convergence have been reported in these papers, though with a storage cost. Despite their promise, the use of coupled schemes still lags that of segregated solvers for more complex physics, and the area remains ripe for future research.

1.7.3 Density-based Schemes

In incompressible flows, pressure is not a thermodynamic quantity and, as noted earlier, it does not appear directly in the continuity equation. However, in compressible flows, it is directly related to density and temperature through the equation of state. This fact has important consequences for numerical solution procedures. The basic differences between incompressible and compressible flow methods are best illustrated by considering the treatment of the convective terms. Consequently, attention will be focused only on the inviscid part of the governing

equations. The analysis begins by writing the equations for mass, momentum, and energy conservation collectively in the following form

$$\frac{\partial}{\partial t}(\mathcal{Q}) + \nabla \cdot (\mathcal{E}) = 0 \quad (1.121)$$

where $\mathcal{Q}$ is the vector of conserved variables and $\mathcal{E}$ represents the inviscid fluxes

$$\mathcal{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix}, \quad \mathcal{E} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho vu \\ \rho wu \\ uH \end{bmatrix} \mathbf{i} + \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho wv \\ vH \end{bmatrix} \mathbf{j} + \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ wH \end{bmatrix} \mathbf{k} \quad (1.122)$$

This equation set, which is also referred to as the Euler equations, is closed by the equation of state, and thermodynamic relations that relate the energy (E), enthalpy (H), density (ρ), and pressure (p).

Equation (1.121) has the same form as the general transport equation, Eq. (1.6), except that the transported variable is a vector and not a scalar. In principle, the same discretization procedures described above for the scalar transport equation can be used. For example, a finite-volume discretization of Eq. (1.121) can be written in a manner similar to Eq. (1.40) as

$$\left[\frac{\partial}{\partial t}(\mathcal{Q}) \right]_0 \Delta \mathcal{V}_0 + \sum_f \mathcal{E}_f \cdot \mathbf{A} = 0 \quad (1.123)$$

The unsteady term can be written in the same manner as Eq. (1.48). The main difference from the scalar equation is in how the convective face fluxes, $\mathcal{E}_f$, are evaluated. Unlike a passive scalar, the face values for the conserved variable cannot be simply be computed using an upwind scheme based on the convecting velocity field. This is because the Euler equations have multiple characteristic directions along which different characteristic variables are propagated with different speeds. The speeds are defined by the eigenvalues of the Jacobian matrix, $\mathcal{A} \equiv \partial \mathcal{E} / \partial \mathcal{Q}$. For ideal gases, the eigenvalues are $V_n + c$, V_n (repeated thrice) and $V_n - c$, where V_n is the face normal velocity and c is the speed of sound. For supersonic flow, $V_n > c$, and all the five eigenvalues are positive. Therefore, the face flux can be written purely in terms of the upwind $\mathcal{Q}$. However, for subsonic flow, one of the eigenvalues is negative. This means that one piece of information at the face must come from the downwind cell and the other four from the upwind cell.

A variety of schemes have been derived for computing the numerical fluxes that preserve the physical properties of the Euler equations. All the schemes have a means of distinguishing the contribution to the face flux from the left and right cell states. In flux-splitting methods [57, 58], the fluxes from the left and right states are combined according to the eigenvalues of some averaged state. In another class of methods, referred to as flux-difference splitting [59], the flux is written as a sum of the averaged flux from both sides and a flux difference, which is typically evaluated from an approximate solution of the Riemann problem formed by the difference of the conserved variables. The ideas discussed in Section 1.5 for higher-order discretization of the convective fluxes can be incorporated into these methods as well.

Once a numerical flux method has been chosen, the discrete set of equations can be solved using either explicit or implicit schemes. For the former, a multistage Runge-Kutta type method is usually employed, with the residual of the fluxes evaluated using the previous stage values

of Q [28]. At each stage the conserved variables, Q , at each location are updated based on the residual; all the other variables can then be computed using the auxiliary equations. This time-marching process is also used for steady-state problems where the solution is updated till it becomes invariant in time. These methods have restrictions on the maximum time step that can be used because of the Courant-Friedrichs-Lewy (CFL) condition [26].

For implicit schemes, the residual is calculated in the same manner as for explicit schemes, but in addition the fluxes are linearized with respect to Q so that a linear system of the same form as Eq. (1.52) is obtained relating Q at any discrete location to the values of Q at neighboring locations. Typically, the unsteady term is retained even for steady-state solutions; it serves the same purpose as the underrelaxation that is used for the scalar transport equation. The resulting matrix has the same sparse pattern as that for the scalar equations but instead of a scalar coefficient, a dense block matrix of dimension 5×5 appears. Such a system can be solved using the block-matrix equivalents of the methods used for scalar systems. Because of the nonlinearity of the equations, iterations must be used even with an implicit coupled solution procedure.

Density-based techniques are very efficient when used for transonic or supersonic flow regimes. However, the pressure–velocity coupling problems discussed earlier for incompressible flows are also encountered in these techniques when used for low-speed applications, both in permitting checkerboarded pressure and velocity fields as well as in causing deterioration of convergence rates. As the Mach number is decreased, the maximum allowable time step decreases, making explicit time-marching methods impractical. The Jacobian A becomes singular in the incompressible limit. Many approaches have been formulated to overcome this problem. Early methods used an artificial compressibility idea to avoid the singularity [49]. Modern methods use a preconditioning approach wherein the unsteady term in Eq. (1.121) is premultiplied with a preconditioning matrix Γ such that the resulting system has well-behaved eigenvalues [60]. The preconditioning matrix can also be tailored to incorporate ideas equivalent to the momentum interpolation discussed in Section 1.7.1 to avoid the checkerboarding problem. In this process, the temporal behavior of the equations becomes nonphysical since the unsteady term is modified but the steady-state solution is a valid solution of the equations. When time accuracy is required, dual-time-stepping schemes [61] are used wherein an additional physical time derivative term is added and at each physical time step the system is iterated to convergence in the pseudo-time variable.

1.8 CLOSURE

In this chapter, a survey of numerical methods to solve the equations for fluid flow and heat transfer has been presented. The emphasis has been on widely used discretization techniques, such as the finite-difference, finite-element, and finite-volume techniques. All these methods are seen to have common elements, including mesh generation, discretization of the governing equations, and solution of linear algebraic equations. These approaches also resolve checkerboarding issues related to the storage of pressure vis-à-vis velocity with similar techniques. The solution of incompressible flows using segregated as well as coupled techniques has been reviewed, as have density-based techniques for compressible flows.

The chapters in the handbook expand on the material presented in this chapter. They are divided into two sections. The emphasis of the first half is on the fundamental algorithmic aspects. Detailed expositions on finite-difference and finite-element methods, boundary-element techniques, control-volume based techniques, high-resolution spectral-element techniques, and meshless methods are presented in Chapters 1–7.

The overview in the present chapter has dealt primarily with the solution of equations governing heat conduction and convection phenomena. Chapter 8 addresses the use of the Monte Carlo method for radiative heat transfer, a statistical technique employing particle-tracking ideas quite different from the Eulerian techniques described in the overview. Chapter 9 also addresses radiative transport, but the techniques described employ many of the discretization ideas discussed in the present chapter for the scalar transport equation. Chapter 10 expands on the introduction to pressure-based techniques for fluid flow given here and Chapters 11 and 12 address modeling and simulation issues related to the computation of turbulent flows. Chapters 13–16 address a variety of enabling technologies related to CFD, including high-performance computing, validation of CFD solvers and models, mesh generation, as well as the emerging area of symbolic–numerical computations.

The emphasis of the second half of the handbook is on applications. In Chapters 17–27, a wide variety of areas are addressed, ranging from biomedical applications and the thermal management of data centers to materials processing applications. Two chapters, on microscale heat transfer and molecular dynamics, address numerical methods for the rapidly evolving field of micro- and nanotechnology. The handbook closes with an overview of available numerical methods and their use, information on the resources available to both the user and the developer of CFD software, as well as discussion of future research directions.

NOMENCLATURE

a_P, a_{nb}	coefficients of discrete equations, Eq. (1.52)
$\mathbf{A}$	area vector, Fig. 1.8
$\mathcal{A}$	Jacobian matrix
b	source term in discrete equation, Eq. (1.52)
c	speed of sound
C_v	specific heat at constant volume
D	diffusion coefficient of species l in mixture, Eq. (1.5)
D_f	diffusion flux on face f , Eq. (1.41)
e, w, n, s	control-volume faces for structured mesh, Fig. 1.14
E, W, N, S	neighbor cell centroid locations, Fig. 1.14
E	total energy per unit mass, Eq. (1.3)
f	control-volume face, Fig. 1.8
F_f	mass flow rate on face f , Eq. (1.40)
F_x	body force in x -momentum equation, Table 1.1
k	thermal conductivity, Eq. (1.3); wave number, Eq. (1.92)
k_t	turbulent thermal conductivity, Eq. (1.3)
L	length of calculation domain
m_l	mass fraction of species l , Eq. (1.5)
N	number of grid points
N_i	shape function, Eq. (1.20)
p	static pressure, Eq. (1.2)
R	residual, Eq. (1.17)
R_l	volumetric source of specie l , Eq. (1.5)
P	centroid of control volume, Fig. (1.7)
S_C, S_P	components of linearized source term, Eq. (1.49)
S_h	energy source per unit volume, Eq. (1.3)
S_m	mass source per unit volume, Eq. (1.1)

S_r	radiation source per unit volume, Eq. (1.3)
S_ϕ	source of ϕ per unit volume, Eq. (1.6)
$\mathcal{S}$	secondary diffusion term, Eq. (1.42)
T	temperature, Eq. (1.3)
(u, v)	Cartesian velocity components
W_i	weight function, Eq. (1.18)
$\mathbf{V}$	velocity vector, Eq. (1.1)
(x, y)	coordinate directions
$\Delta\mathcal{V}$	volume of control volume, Eq. (1.40)

Greek Symbols

ϕ	transported scalar, Eq. (1.6)
Γ	diffusion coefficient, Eq. (1.6)
μ	molecular viscosity, Eq. (1.2)
μ_t	turbulent viscosity, Eq. (1.2)
Ψ	limiter function, Eq. (1.80)
ρ	density, Eq. (1.1)
σ_m	turbulent Schmidt number, Eq. (1.5)

REFERENCES

1. S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*, McGraw-Hill, New York, 1980.
2. J. Y. Murthy and S. R. Mathur, A Conservative Numerical Scheme for the Energy Equation, *J. Heat Transfer*, **120**, 1081–1085 (1998).
3. J. F. Thompson, B. K. Soni, and N. P. Weatherill (eds.), *Handbook of Grid Generation*, CRC Press, 1998.
4. O. C. Zienkiewicz, *The Finite Element Method*, Vols. 1–3, Butterworth-Heinemann, Oxford, UK, 2000.
5. K. C. Karki and S. V. Patankar, Pressure-based Calculation Procedure for Viscous Flows at All Speeds in Arbitrary Configurations, *AIAA J.*, **27**, 1167–1174 (1989).
6. S. R. Mathur and J. Y. Murthy, A Pressure-based Method for Unstructured Meshes, *Numer. Heat Transfer*, **31**, 195–216 (1997).
7. B. R. Baliga and S. V. Patankar, A Control-volume Finite Element Method for Two-dimensional Fluid Flow and Heat Transfer, *Numer. Heat Transfer*, **6**, 245–261, 1983.
8. C. A. Brebbia, J.C.F. Telles, and L. C. Wrobel, *Boundary Element Techniques*, Springer-Verlag, Berlin, 1984.
9. A. J. Kasab and L. C. Wrobel, Boundary Elements for Heat Conduction, in W. J. Minkowycz and E. M. Sparrow (eds.), *Advances in Numerical Heat Transfer*, Vol. 2, pp. 143–187, Taylor & Francis, New York, 2000.
10. J. D. Hoffman, *Numerical Methods for Engineers and Scientists*, Marcel Dekker, New York, 2001.
11. J. N. Reddy, *An Introduction to the Finite Element Method*, McGraw-Hill, New York, 1993.
12. K. H. Heubner and E. A. Thornton, *The Finite Element Method for Engineers*, Wiley, New York, 1982.
13. T. Davis, *UMFPACK Version 4.1*, University of Florida, Gainesville, Florida, April 2003.
14. W. Shyy, S. M. Correa, and M. E. Braaten, Computation of Flow in a Gas Turbine Combustor, *Combustion Sci. Technol.*, **58**, 97–117 (1988).
15. W. Shyy and T. G. Vu, On the Adaptation of Velocity Variable and Grid Systems for Fluid Flow in Curvilinear Co-ordinates, *J. Comput. Phys.*, **92**, 82–105 (1991).

16. M. Peric, *A Finite Volume Method for the Prediction of Three-Dimensional Fluid Flow in Complex Ducts*, Ph.D thesis, University of London, 1985.
17. C. M. Rhie and W. L. Chow, Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation, *AIAA J.*, **21**, 1523–1532 (1983).
18. T. E. Tezduyar, R. Shih, S. Mittal, and S. E. Ray, Incompressible Flow Computations with Stabilized Bilinear and Linear Equal-order Interpolation Velocity–Pressure Elements, Technical Report, *University of Minnesota Supercomputer Institute Research Report UMSI90/165*, 1990.
19. Y. Jiang and A. J. Przekwas, Implicit, Pressure-based Incompressible Navier-Stokes Equations Solver for Unstructured Meshes, *AIAA-94-0305*, 1994.
20. L. Davidson, A Pressure Correction Method for Unstructured Meshes with Arbitrary Control Volumes, *Int. J. Numer. Methods Fluids*, **22**, 265–281 (1996).
21. I. Demirdzic and S. Muzaferija, Numerical Method for Coupled Fluid Flow, Heat Transfer and Stress Analysis Using Unstructured Moving Meshes with Cells of Arbitrary Topology, *Comput. Methods Appl. Mech. Eng.*, **125**, 235–255 (1995).
22. C. Prakash, An Improved Control Volume Finite Element Method for Heat and Mass Transfer, *Numer. Heat Transfer*, **9**, 253–276 (1986).
23. S. K. Lele, Compact Finite Differences with Spectral-like Resolution, *J. Comput. Phys.*, **103**, 16–42 (1992).
24. C. H. Amon, Spectral Element Methods for Unsteady Fluid Flow and Heat Transfer in Complex Geometries: Methodology and Applications, in W. J. Minkowycz and E. M. Sparrow (eds.), *Advances in Numerical Heat Transfer*, Vol. 2, pp. 71–108, Taylor & Francis, New York 2000.
25. R. Peyret, *Spectral Methods for Incompressible Viscous Flow*, Springer-Verlag, New York, 2002.
26. J. C. Tannehill, D. Anderson, and R. H. Pletcher, *Computational Fluid Mechanics and Heat Transfer*, 2d ed., Series in Computational Methods and Physical Processes in Mechanics and Thermal Sciences. Taylor & Francis, Philadelphia, 1997.
27. B. P. Leonard, A Stable and Accurate Convective Modeling Procedure Based on Quadratic Upstream Interpolation. *Comput. Methods Appl. Mech. Eng.*, **19**, 59–98 (1979).
28. A. Jameson, W. Schmidt, and E. Turkel, Numerical Solution of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes, *AIAA 91-1259*, June 1981.
29. C. Hirsch, *Numerical Computation of Internal and External Flows: Computational Methods for Inviscid and Viscous Flows*, Wiley, New York, 1990.
30. C. B. Laney, *Computational Gas Dynamics*, Cambridge University Press, Cambridge, UK, 1998.
31. J. H. Ferziger and M. Peric, *Computational Methods for Fluid Dynamics*, Springer-Verlag, New York, 1996.
32. H. L. Stone, Iterative Solution of Implicit Approximations of Multidimensional Partial Differential Equations, *SIAM J. Numer. Anal.*, **5**, 530–558 (1968).
33. W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2d ed., SIAM, Philadelphia, 2000.
34. S. P. Vanka, A Calculation Procedure for Three-dimensional Steady Recirculating Flows Using Multigrid Methods, *Comput. Methods Appl. Mech. Eng.*, **55**, 321–338 (1986).
35. R. Jyotsna and S. P. Vanka, Multigrid Calculation of Steady, Viscous Flow in a Triangular Cavity, *J. Comput. Phys.*, **122**, 107–117 (1995).
36. B. R. Hutchinson and G. D. Raithby, A Multigrid Method Based on the Additive Correction Strategy, *Numer. Heat Transfer*, **9**, 511–537 (1986).
37. R. D. Lonsdale, An Algebraic Multigrid Scheme for Solving the Navier-Stokes Equations on Unstructured Meshes, in *Proceedings of the 7th International Conference on Numerical Methods in Laminar and Turbulent Flow*, pp. 1432–1442, Stanford University, Palo Alto, CA, 1991.
38. A. Brandt, Multi-level Adaptive Solutions to Boundary Value Problems, *Math. Comput.*, **31**, 333–390 (1977).

39. S. R. Mathur and J. Y. Murthy, Unstructured Finite Volume Methods for Multi-mode Heat Transfer, in W. J. Minkowycz and E. M. Sparrow (eds.), *Advances in Numerical Heat Transfer*, Vol. 2, pp. 37–67, Taylor & Francis, New York, 2000.
40. G. H. Golub and C. van Loan, *Matrix Computations*, Johns-Hopkins University Press, Baltimore, MD, 1990.
41. H. A. van den Vorst and P. Sonneveld, CGSTAB, a More Smoothly Converging Variant of CGS, Technical report, Delft University of Technology, Delft, The Netherlands, 1990.
42. H. A. van den Vorst, BI-CGSTAB: A Fast and Smoothly Converging Variant of BI-CG for the Solution of Non-symmetric Linear Systems, *SIAM J. Sci. Stat. Comput.*, **13**, 631–644 (1992).
43. Y. Saad and M. H. Schultz, GMRES: A Generalized Residual Algorithm for Solving Non-symmetric Linear Systems, *SIAM J. Sci. Stat. Comput.*, **7**, 856–869 (1986).
44. P. Hood and C. Taylor, Navier-Stokes Equations Using Mixed Interpolation, in J. T. Oden, O. C. Zienkiewicz, R. H. Gallagher, and C. Taylor (eds.), *Finite Element Methods in Flow Problems*, pp. 121–132, UAH Press, Huntsville, AL, 1974.
45. P. S. Huyakorn, C. Taylor, R. L. Lee, and P. M. Gresho, A Comparison of Various Mixed-interpolation Finite Elements in the Velocity–Pressure Formulation of the Navier-Stokes Equations. *Comput. Fluids*, **6**, 25–35 (1978).
46. S. Majumdar, Role of Underrelaxation in Momentum Interpolation for Calculation of Flow with Non-staggered Grids, *Numer. Heat Transfer*, **13**, 125–132 (1988).
47. C. Prakash and S. V. Patankar, A Control-volume Based Finite Element Method for Solving the Navier-Stokes Equations Using Equal-order Velocity–Pressure Interpolation, *Numer. Heat Transfer*, **8**, 259–280 (1985).
48. T. J. R. Hughes, L. P. Franca, and M. Becestra, A New Finite Element Formulation for Computational Fluid Dynamics, V: Circumventing the Babuska-Brezzi Condition: A Stable Petrov-Galerkin Formulation of the Stokes Problem Accommodating Equal-order Interpolations, *Comput. Methods Appl. Mech. Eng.*, **59**, 85–99 (1986).
49. A. J. Chorin, A Numerical Method for Solving Incompressible Viscous Flow Problems, *J. Comput. Phys.*, **2**, 12–26 (1967).
50. M. H. Kobayashi and J. C. F. Pereira, Characteristic-based Pressure Correction at All Speeds, *AIAA J.*, **34**, 272–280 (1996).
51. S. R. Mathur and J. Y. Murthy, All-speed Flows on Unstructured Meshes Using a Pressure-correction Approach, *AIAA 99-3365*, Norfolk, VA, 1999.
52. S. V. Patankar and D. B. Spalding, A Calculation Procedure for Heat, Mass and Momentum Transfer in Three-dimensional Parabolic Flows, *Int. J. Heat Mass Transfer*, **15**, 1787–1805 (1972).
53. J. P. Van Doormaal and G. D. Raithby, Enhancements of the SIMPLE Method for Predicting Incompressible Flow Problems, *Numer. Heat Transfer*, **7**, 147–163 (1984).
54. M. Raw, Robustness of Coupled Algebraic Multigrid for the Navier-Stokes Equations. *AIAA 96-0297*, 1996.
55. K. M. Smith, W. K. Cope, and S. P. Vanka, A Multigrid Procedure for Three-dimensional Flows on Non-orthogonal Collocated Grids, *Int. J. Numer. Methods Fluids*, **17**, 887–904 (1993).
56. R. Webster, An Algebraic Multigrid Solver for Navier-Stokes Problems, *Int. J. Numer. Methods Fluids*, **18**, 761–780 (1994).
57. B. van Leer, Toward the Ultimate Conservative Difference Scheme, *J. Comput. Phys.*, **32**, 101–136 (1979).
58. M. S. Liou and J. C. Steffen, A New Flux Splitting Scheme, *J. Comput. Phys.*, **107**, 23–39 (1993).
59. P. L. Roe, Approximate Riemann Solvers, Parameter Vectors and Difference Schemes, *J. Comput. Phys.*, **43**, 357–372 (1982).
60. Y. H. Choi and C. L. Merkle, Application of Preconditioning in Viscous Flows, *J. Comput. Phys.*, **105**, 207–223 (1993).
61. S. Venkateswaran and C. L. Merkle, Dual Time Stepping and Preconditioning for Unsteady Computations, *AIAA-95-0079*, 1995.